Wu J. Uncovering several useful structures of complex networks in computer science applications. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY, 2026. DOI: 10.1007/s11390-025-5946-2, CSTR: 32374.14.s11390-025-5946-2

Uncovering Several Useful Structures of Complex Networks in Computer Science Applications

Jie Wu, *Fellow, CCF, IEEE*

Cloud Computing Research Institute, China Telecom, Beijing 100191, China
Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122, U.S.A.

E-mail: jiewu@temple.edu

Received September 15, 2025; accepted November 25, 2025.

Abstract Graph theory originated in the 18th century when Euler worked on the Königsberg bridge problem. Since then, graph theory has been applied to many fields, ranging from biological networks to transportation networks. In this paper, we study complex networks and their applications in computer science, with a focus on computer system and network applications, including mobile and wireless networks. In a social society, many group activities can be represented as a complex network in which entities (vertices) are connected in pairs by lines (edges). Uncovering useful global structures of complex networks is important for understanding system behaviors and providing global guidance for application designs. We briefly review existing graph models, discuss several mechanisms used in traditional graph theory, distributed computing, and system communities, and point out their limitations. Throughout the paper, we focus on how to uncover useful structures in dynamic networks and summarize three promising approaches to uncover useful structures: trimming, layering, and remapping. We then study several distributed and local labeling and coding methods as a way of learning and their relationships to machine learning (ML), including graph neural networks (GNNs). Finally, we present challenges in algorithmic techniques, with a focus on distributed and localized ones, in labeling and coding in a dynamic network.

Keywords complex network, distributed and localized solution, dynamic and mobile network, graph theory, labeling and coding, learning, structural property

1 Introduction

Graph theory originated in the 18th century when Euler worked on the Königsberg bridge problem[①]. Since then, graph theory has been applied to many fields, ranging from biological networks to transportation networks. In computer science (CS), the classic four-color problem[②] finds applications in resource allocations, such as wireless channel allocations where adjacent areas should use different channels to avoid contentions and collisions. In this paper, we study complex networks with a focus on computer systems and networks, including mobile and wireless networks. We concentrate on three aspects: graph models, graph structures, and graph solutions using distributed and localized labeling and coding. The study focuses on work done using graph theory in applications in CS. We do not cover graph neural networks (GNNs)[1, 2] and their applications. Most of the work discussed in this paper appeared before GNNs. However, both the discovery of graph structures and the distributed and localized labeling and coding of graphs in this paper are related to graph learning.

1.1 Background

Many group activities can be represented as a complex network in which entities (vertices) are connected in pairs by lines (edges). This complex network is applicable in multiple different fields and can

Survey
Special Issue: Celebrating the 40th Anniversary of JCST
[①]https://en.wikipedia.org/wiki/Seven_Bridges_of_K%C3%B6nigsberg, Nov. 2025.
[②]https://en.wikipedia.org/wiki/Four_color_theorem, Nov. 2025.

affect everything from the Internet, the food web, and metabolic networks to social networks[3, 4]. The graph model is commonly used to model a complex network. However, the traditional graph model[5, 6] is not convenient for modeling complex dynamic and mobile networks in which the connections between nodes change over time.

There is no formal definition of a graph structure. In this paper, a structure can be "logical" like a special property associated with a network (e.g. a small-world network[7]) or "physical" like a special subnetwork (e.g. the backbone in the Internet). The structure considered in this paper is global and spans the entire network. Uncovering the useful global structure (or simply, the structure) of a complex network is important to understand the behavior of the system and provide global guidance for application designs. Structures can be determined either ahead of time in a static setting or on-the-fly in a dynamic setting. We focus on algorithmic techniques that deal with ways of discovering and/or representing structures.

Many structures are embedded and are influenced by various factors. Complex networks can consist of multiple layers[8] from application sessions and social relationships to physical network layers. Interactions and influences between layers may play an important role in the shaping of network structures. There are several success stories of using special structures or features to support various applications. In a small-world network[7] with six degrees of separation, if the connection of the node follows the inverse square distribution (that is, the probability that two nodes u and v are connected is proportional to $d(u,v)^{-2}$, where $d(\cdot)$ is the distance), there is a localized solution[9] in which each node knows only its own local connections and is capable of finding short paths with high probability.

This paper focuses on three areas of research related to structures: 1) determining appropriate graph models, 2) discovering useful structures for a given graph model, and 3) designing algorithmic techniques as a way of learning to uncover and/or represent structures using distributed and localized labeling and coding.

1.2 Scope

We will discuss the challenges we face dealing with the three areas and explore possible solutions. In a dynamic environment, the connections of the nodes (or contacts) are based on the notion of a "vicinity" between nodes in time and space. We will examine two special intersection graphs: unit disk graphs and interval graphs, and explore their limitations before looking at a more general time-evolving graph. In uncovering useful structures, we do not have simple solutions for different types of applications. Instead, we study three strategies that can be used for a range of applications: trimming, layering, and remapping. Each strategy is illustrated through several applications. We focus our attention on dynamic and mobile environments where the graph structure is dynamic and each node has a limited view of its surroundings.

Finally, each structure is discovered in either a distributed or a localized solution. Like a distributed solution, a localized solution[10] is also based on message passing of node status. However, such status does not propagate beyond a certain number of hops, which is usually a small number. We advocate distributed and localized labeling and coding schemes that use colors and labels to identify logical and physical structures. Distributed or localized labeling and coding is also a learning process consisting of an iterative process that propagates the node states until equilibrium. There is a simple decision process at each node at the end of the process, unlike GNNs, which use a neural network (NN) to produce an output. Our labeling and coding methods are applied directly to the network under study for a particular graph structure and application.

Throughout the paper, we describe approaches used in different communities, including graph theory, distributed computing (such as ACM PODC in the theoretical community), and distributed systems (such as IEEE ICDCS in the system community). The discussion will focus on the differences among different methods and possible extensions. Because the area under study is vast, this position paper covers only a subset of problems in CS, many of which come from research results of the author over the past 35 years; it does not intend to be a comprehensive survey. For example, research on social networks reveals some interesting metrics and properties, such as the centrality associated with nodes and the power law and heavy tail distributions in the node degree distribution. These metrics and properties are not considered in this paper unless they are related to the study topics. In particular, we will not cover subjects related to GNNs[1, 2] and social networks[11, 12], as

there are many books and surveys already. In addition, we will not cover the work on graph processing[13, 14]. Graph processing refers to computational techniques and algorithms used to analyze, manage, and manipulate graph structures. In fact, there is Graph 500[15], a benchmark for supercomputers based on large-scale graph analysis.

1.3 Overview

The remainder of this paper is organized as follows. Section 2 surveys the relevant graph models for complex networks with a focus on special intersection graphs and general graphs that evolve over time. Random graphs as special models are also discussed. Section 3 describes three ways to uncover structures as a way of learning. Section 4 discusses distributed and local labeling and coding as a way of learning to uncover the structure for a specific application. The way of achieving self-stabilization in a dynamic network is also discussed. Finally, Section 5 concludes the paper with a short discussion on relationships with GNNs.

2 Graph Model

Traditionally, a complex network can be represented as a graph $G = (V, E)$[6] with a set of vertices (simply called nodes here) V and a set of edges (simply called links here) E. In CS, networks include, but are not limited to, the Internet, peer-to-peer (P2P), mobile ad hoc (MANET)[16], sensor, vehicular ad hoc (VANET)③, social, and delay/disruption-tolerant (DTN)④ networks. These networks operate in special environments that pose unique challenges to network design.

In data center networks (DCNs), two types of nodes are used: servers (regular nodes on graphs as circles) and switches (squares in Fig.1(d)). One way is to view switches as hyperedges connecting multiple servers or switches. In general, there are three possible types of connection: server-server, server-switch, and switch-switch[17]. Most DCNs use server-switch and switch-switch connections as in the spine-leaf structure, which is basically a folded Clos network⑤ by placing all servers on one side. The remainder of the discussion focuses only on graphs with server-server, or direct, connections without using switches.

In mobile networks like DTNs and VANETs, network connection is highly dynamic and disruptive due to node mobility. In these networks, the links are also called contacts between two nodes with appropriate contact duration and inter-contact time. The traditional graph model cannot adequately capture the dynamic nature of the connections in the mobile networks.

- *Which Graph Model is Suitable for Representing a Complex Network?*

Unfortunately, there is no model that can fit all cases. We explore models for special cases and follow with a more general model that evolves over time. We also discuss controlled random graphs as a useful tool for graph construction for a particular purpose.

2.1 Intersection Graphs

There are several types of networks where node connections are based on their vicinities in time (such as online social networks) and in space (such as sensor networks, MANETs, and VANETs). Intersection graphs can be used for these cases. Intersection graphs[18] are formed from a family of sets $S_i, i = 1, 2, \ldots$, by creating a vertex v_i for each set S_i and connecting two vertices v_i and v_j by an edge whenever the corresponding two sets have a non-empty intersection:

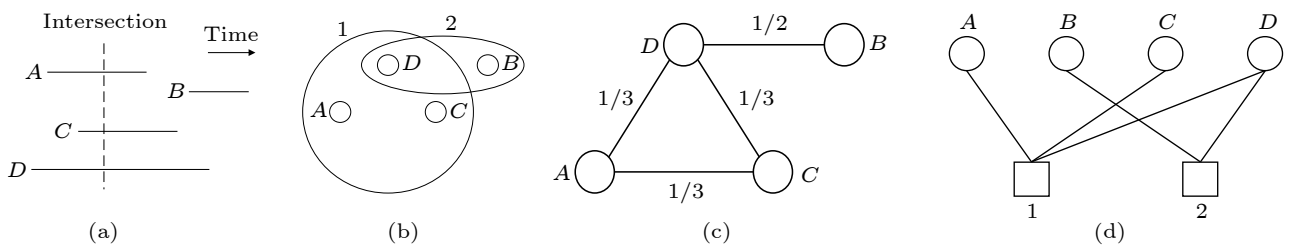


Fig.1. Illustration of interval graph with four different representations. (a) Line intervals along time. (b) Hypergraph. (c) Clique-expanded graph. (d) Star-expanded graph.

③https://en.wikipedia.org/wiki/Vehicular_ad_hoc_network, Nov. 2025.
④https://en.wikipedia.org/wiki/Delay-tolerant_networking, Nov. 2025.
⑤https://en.wikipedia.org/wiki/Clos_network, Nov. 2025.

$$E = \{(v_i, v_j) \mid S_i \cap S_j \neq \emptyset\}.$$

Among special intersection graphs, unit disk graphs[6] are a family of unit disks (for set S_i) in a 2-dimensional (2-D) space. Each unit disk is a vertex centered at v_i with a radius r. An edge exists between two nodes if both lie within the radius (r) of each other. Unit disk graphs have been extensively studied for wireless applications in sensor networks, MANETs, and VANETs, because most wireless devices have a transmission radius. Note that not all graphs are unit disk graphs. An example is a star graph with one center node and six or more leaves. As a special graph, unit disk graphs have some unique properties that general graphs do not have. For example, a constant approximation algorithm exists to solve the Traveling Salesman Problem (TSP) in unit disk graphs.

An interval graph[7] is the intersection graph of a family of intervals on the real line/time (for the set S_i). Each line interval represents a vertex. An edge exists if the corresponding intervals intersect. If a line interval represents a time period, an interval graph can be used to represent an online social network (see Fig.1(a)). Not all graphs are interval graphs. In fact, if G is an interval graph, it must be a chordal graph or a perfect graph. A chordal graph is one in which all cycles of four or more vertices have a chord, which is an edge that is not part of the cycle but connects two vertices of the cycle. The impossibility of a large chordless cycle is that time is linear, not circular. Interval graphs can be recognized in linear time, and an optimal graph coloring or maximum clique in these graphs can be found in linear time. In an online social network, each user can be online multiple times, and then multiple-interval graphs (interval graphs where each vertex may have more than one interval associated with it) can be used. The following question arises: What are the unique properties that we can explore for multi-interval graphs? What type of distributions do multiple-interval graphs follow? More importantly, what types of properties of online social networks can be explored through the edge density distribution? Understanding the edge density distribution can also play an important role in understanding online social network behaviors such as social influencing and recommendation.

Vertices in interval graphs are not necessarily connected in pairs. In Fig.1(a), three nodes A, C, D are intersected at a certain time; this example is analogous to an online social network with three online users at the same time. A hyperedge, a generalized edge connecting more than two vertices, seems to be more appropriate in this case. A general hypergraph[19] can be represented as several circles (hyperedges), each of which contains several connecting nodes, as shown in Fig.1(b). There are two other ways of representing a hypergraph using a clique-expanded graph, where each edge is associated with a weight that is reversely proportional to the clique size (such as 3 and 2 in Fig.1(b)), as shown in Fig.1(c) and a star-expanded graph, where a bipartite graph is constructed between circles and nodes as shown in Fig.1(d). These representations can be useful in hypergraph learning[20] and hypergraph neural networks (HGNNs)[21].

2.2 Time-Evolving Graphs

Time-evolving graphs[22] try to present graphs in both time and space. Such graphs have also been called temporal graphs and time-varying graphs in different settings. We start with a brief overview of the graphs that evolve over time, the connectivity that is sensitive to time, and the different performance measures for a path.

Let $G = (V, E)$ be a graph. $G_0, G_1, \ldots, G_k$ is an ordered sequence of subgraphs of G with an increasing time sequence $t_0, t_1, \ldots, t_k$. $G_i = (V_i, E_i)$ is called a subgraph during the time period $[t_i, t_{i+1})$, where $V_i = V$ and $E_i \subseteq E$. The corresponding time-evolving graph, denoted as EG, is the collection of G_i. Figs.2(a) and 2(b) show two snapshots of a cyclic VANET that has three mobile nodes (B, C, and D) and two static nodes (A and E). Fig.2(c) is the corresponding time-evolving graph. Edge labels have cycles; for example, (B, D) and (C, D) have a cycle of 6, (A, D) has a cycle of 2, and (A, B) and (B, C) have a cycle of 3.

In EG, $u \xrightarrow{i} v$ denotes the edge label that indicates the existence of the edge (u, v) at time t_i for G_i (or simply i in Fig.2(c)). Message transmission over an edge (also called contact) is instantaneous with a short period of contact duration. The edge (A, E) between two static nodes A and E always exists, which can take any time label. A path $u \xrightarrow{*} v$ is an alterna-

[6] https://en.wikipedia.org/wiki/Unit_disk_graph, Nov. 2025.
[7] https://en.wikipedia.org/wiki/Interval_graph, Nov. 2025.

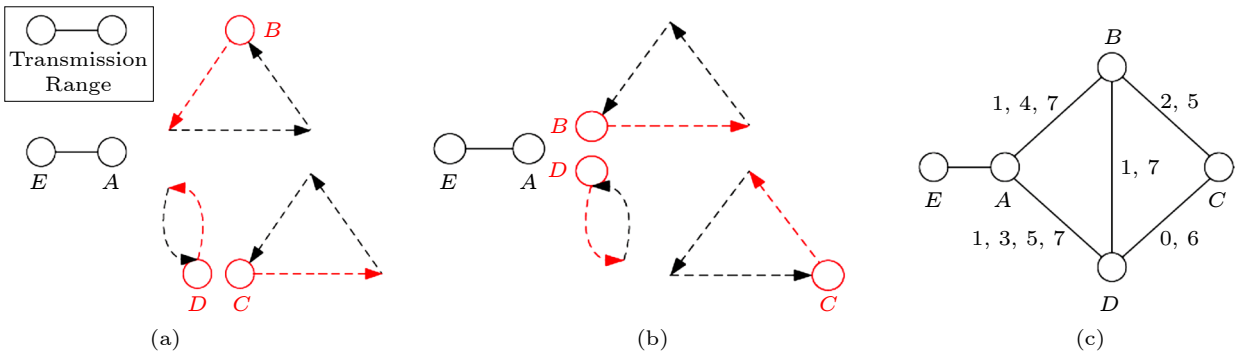


Fig.2. Cyclic VANET example with three mobile nodes, B, C, and D, with moving cycles 3, 3, and 2, respectively, and two static nodes A and E. (a) Snapshot at time unit 0. (b) Snapshot at time unit 1. (c) Corresponding time-evolving graph.

tive sequence of vertices and edges with non-decreasing edge labels. Here, $*$ stands for a wild card symbol for a sequence of labels. The vertex u is said to be connected to v at time unit i if a path $u \xrightarrow{*} v$ exists whose first edge label is larger than or equal to i. In Fig.2(c), path $A \xrightarrow{4} B \xrightarrow{5} C$ exists. Here, A is connected to C at time i for 0, 1, 2, 3, or 4, assuming that each vertex, including A, has sufficient storage capacity to store messages from previous contacts. In a particular time slot, two vertices may not be connected. In fact, A and C in Fig.2 are not connected in any particular time slot. Hence, the network is not connected at any given time. However, store-carry forwarding can still deliver messages. A weighted time-evolving graph has a definition similar to the graph that changes over time, except that each edge in the time unit i is associated with a weight w_i, which has different interpretations depending on the application. For example, a weight can be the bandwidth, transmission delay, or reliability of a link.

Using EG, the traditional graph terminology can be extended to a temporal one. For example, the path in the traditional graph is extended to journey (which is a path over time), the distance to temporal distance, and the diameter to dynamic diameter (which is the flood time). For example, in Fig.2(c), there are following journals from A to C via B: $A \xrightarrow{1} B \xrightarrow{2} C$, $A \xrightarrow{1} B \xrightarrow{5} C$, and $A \xrightarrow{4} B \xrightarrow{5} C$.

We can consider the following path optimization problems as extensions of the traditional shortest path problem, but still solvable using the variations of the classical Dijkstra's shortest path algorithm[23].

- *Earliest Completion Time Path*: find a path with the earliest completion time at the destination.
- *Minimum Hop Path*: find a path with the minimum number of hops to the destination.
- *Fastest Path*: find a path with the minimum span (i.e., elapsed time) between its first contact and its last contact.

In Fig.2(c), among the journals from A to C via B, $A \xrightarrow{1} B \xrightarrow{2} C$ the one has the earliest completion time. Both $A \xrightarrow{1} B \xrightarrow{2} C$ and $A \xrightarrow{4} B \xrightarrow{5} C$ are the fastest.

One challenge is how a graph model can capture the essence of complex networks while still being simple enough so that many optimization problems are tractable. In this spirit, we should exclude non-essential parameters. Another approach uses a macro-level model instead of micro-level time unit labels for each edge. In the system community, time unit labels are abstracted as contacts that follow a certain distribution based on a given mobility model[24]. Two measures are often used: contact duration distribution and inter-contact time distribution. The exponential distribution is frequently used because of the simplicity of its mathematics. However, a random waypoint mobility[⑧] without a boundary does not meet the exponential distribution for either contact duration or inter-contact time. In the theoretical community, the Markovian process[⑨] in which the network topology at time t depends only on its topology at time $t-1$ is commonly used and has a unique stationary distribution. For example, the elegant two-state edge-Markovian process is used to describe edge dynamics. If an edge exists at time i, at time $i+1$, it dies with probability p. If the edge does not exist at time i, it will appear at time $i+1$ with another probability q. This model has been used successfully to calculate the dynamic diameter[25]. However, Markovian models are still too simple for various mobility patterns. The

[⑧]https://en.wikipedia.org/wiki/Random_waypoint_model, Nov. 2025.
[⑨]https://en.wikipedia.org/wiki/Markov_decision_process, Nov. 2025.

question of the existence of other models, which are both mathematically elegant and better match practical mobility models remains.

In general, there is a trade-off between the expressiveness and the decision-power of a model. We should strive for the simplest model, e.g., a model focused on space or time only with just enough power to study the problem at hand. However, a more powerful model focused on both time and space potentially reveals more properties than a weaker one.

Fig.3 shows an application of time-evolving graph in modeling the spread of COVID-19 in two classrooms (i.e., Room 1 and Room 2) that involve four students A, B, C, and D. As seen in Fig.3(a), different sets of nodes occupy the same locations, at different times t_1 and t_2 (or simply labeled 1 and 2, respectively), in a spatio-temporal pattern. All individuals in the same room are assumed to be in proximity close enough to all the others in the room to spread the disease through respiratory droplets, as shown in solid lines in Fig.3(b). With aerosol transmission, infected nodes can shed viral particles while in the room, which may infect others in the room simultaneously at the moment or within three hours. Assuming $t_2 > t_1$ in Fig.3(c) with $t_2 - t_1$ no more than three hours, the unidirectional arrow in Fig.3(b) shows the additional aerosol transmissions. Note that each droplet spread implies an aerosol spread. Fig.3(c) shows a simplified graph by adding two nodes (Room 1 and Room 2) to reduce the number of transmission links. With this EG, we can easily evaluate various ways of virus spreading[26].

2.3 Random Graphs

Random graphs[11, 27, 28] are based on probability distributions in graphs. A random graph can be described by a probability or a random process which generates links that connect nodes. For example, in the Erdos-Rényi random graph $G(n, p)$[29], each edge of an n-node graph has a probability of p of its existence. Random graphs have been used for many CS applications in designing network structures in a controlled way. Here, we look at two applications: controlled small-world networks[30] for routing and approximate nearest neighbor (ANN) search[31] in database applications, and controlled random graphs with a small diameter (and hence a large spectral gap[32]).

Consider an $n \times n$ grid (2-D mesh) as in Fig.4. Each non-boundary node u has four neighbors (via

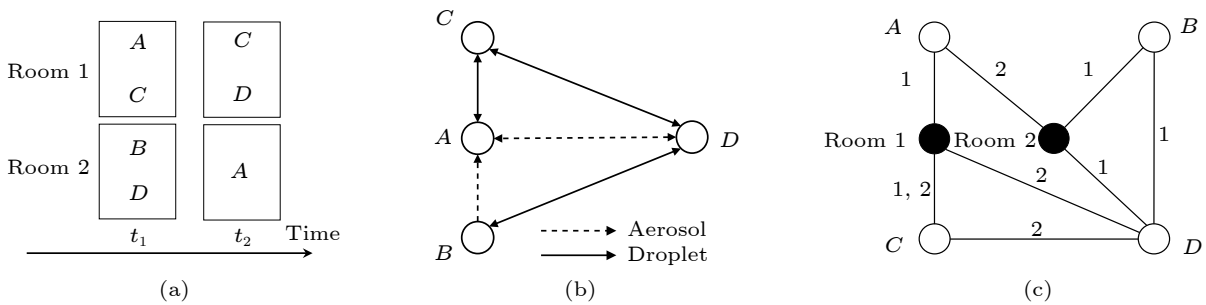


Fig.3. COVID-19 virus spreading through droplet and aerosol among four students in two classrooms: Class 1 and Class 2. (a) Student assignments at t_1 and t_2. (b) Aerosol and droplet spreading. (c) Time-evolving graph at t_1 and t_2.

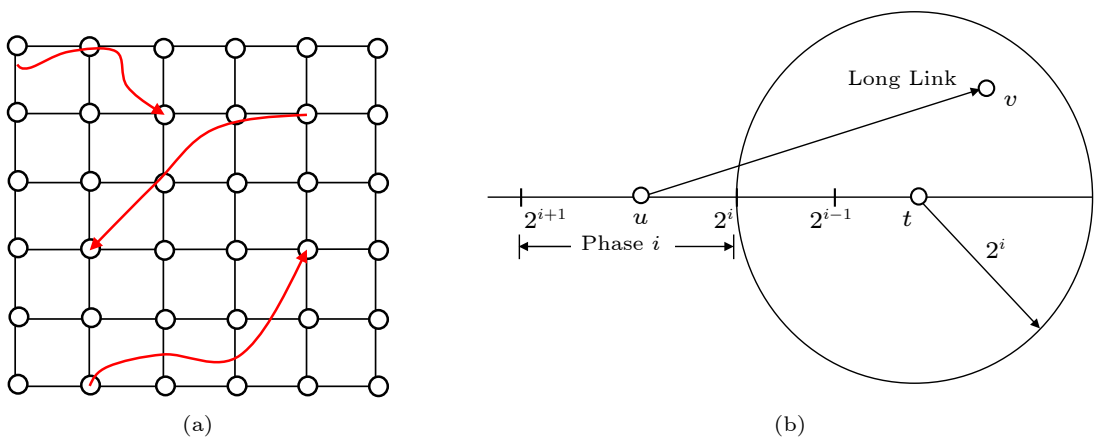


Fig.4. Controlled small-world network[33]. (a) Long links over a 2-D mesh. (b) Routing in phases from u to t.

short links). In addition, each node has a long link (u, v) with a probability of $cd(u, v)^{-2}$ with a constant c. Here, $d(u, v)$ measures the Hamming distance between u and v in the 2-D mesh. In total, node u has exactly l long links such that $\sum_{v \in G} cd(u, v)^{-2} = l$. In routing from u to t, as shown in Fig 4(b), it is operated in phases. The routing is in phase i if the current node u satisfies $2^i < d(u, t) \leq 2^{i+1}$ where t is the destination node. At node u in phase i, a long link (u, v) is used if $d(v, t) \leq 2^i$. Otherwise, a short link is used based on greedy routing (that is, a neighbor closer to t in the 2-D mesh). In phase i, it is shown that the expected time before the current node has a legitimate long link is bounded proportionally to $\log n$[30, 33]. Once v is reached, it becomes the current node and enters the next phase. As there are at most $\log n$ phases, the overall routing path length is $O(\log^2 n)$.

Another controlled random graph example is used to generate a small diameter for a graph of arbitrary sizes. Such a graph is suitable for various applications, including short delay for collection communication and fast convergence in distributed federal learning. Consider the following construction of n-node d-regular graphs, where $d = 2k$[34]. We use $k = 1$ as an example. First, we generate $d = 2$ rings in Fig.5(a). For each ring, assign n nodes based on the random number generator from (0, 1) as shown in Table 1. Then arrange these nodes according to these numbers in an order. Finally, superimpose k rings to form a $(d = 2k)$-degree regular graph as shown in Fig.5(b). It is assumed that each node in a ring has two different neighbors with all the other rings. It has been shown in [34] that such a graph has a small diameter.

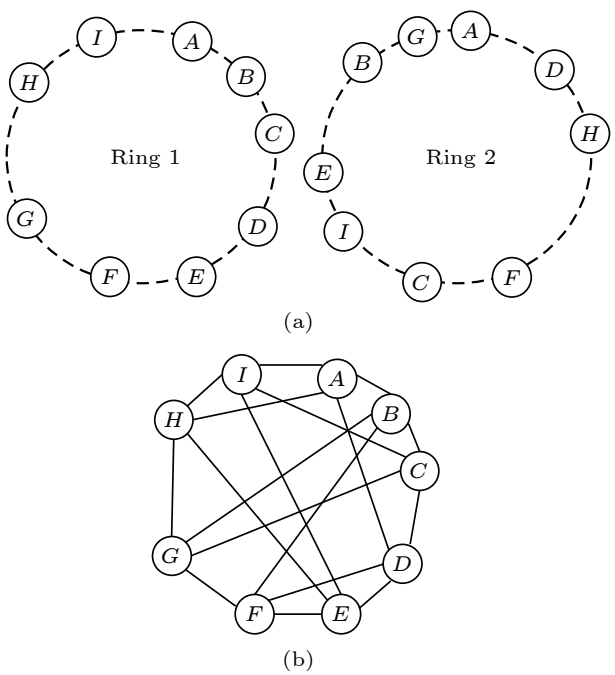


Fig.5. $(n = 9)$-node $(d = 2k)$-regular graph with $k = 2$ from Table 1. (a) Assigning nodes on two rings based on Table 1. (b) Combining two rings in (a).

Table 1. Lists Generated from Random Numbers in (0, 1)

Node ID	Ring 1	Ring 2
A	0.08	0.02
B	0.15	0.87
C	0.23	0.56
D	0.33	0.14
E	0.45	0.73
F	0.53	0.41
G	0.66	0.68
H	0.80	0.23
I	0.92	0.95

3 Uncovering Useful Structures

This section focuses on uncovering useful network structures for a given network. Quite a lot of work has been done in social networks in terms of centrality[35]. Among various forms of centrality: 1) node degree measures the number of neighbors of a node; 2) closeness is the average length of the shortest path between a node and all other nodes on the graph; 3) betweenness quantifies the number of times that a node acts as a bridge along the shortest path between two other nodes; 4) eigenvector centrality, including PageRank[36], measures a node score as the summation of the scores of its neighbors. Centrality primarily measures the importance of a single node. Here, we focus on structures or structural properties that span the entire network. In addition, such structures or structural properties are useful in supporting network-wide applications. Like centrality, a network structure that is optimal for one application is often suboptimal for a different application. We discuss three approaches that can be used to uncover such useful structures for various applications: 1) structure trimming, 2) structure layering, and 3) structure remapping. Note that these approaches can be applied jointly following a certain sequence. In [37], a special structure on top of a lightning network[38] can be constructed for a blockchain application through a trimming process followed by a clustering process (which is a special layering approach) or a clustering process followed by a trimming process.

3.1 Structural Trimming

Structural trimming deals with the removal of links and/or nodes to form a subgraph as a useful structure. Usually a subgraph maintains several of the global properties of the original graph. Basic properties include connectivity and inclusion of a minimum spanning tree or a shortest path tree. In some cases, a property is an approximation of a global measure. For example, subgraph distances closely resemble the distances in the original graph for designing approximation algorithms for graph problems[39]. There is also a vast literature on P2P networks[40] that add connections to form an overlay network. Here, we focus on structural trimming instead of overlay. The main purpose of trimming is to reduce the complexity of information dissemination or to reduce the complexity of network search without losing the desirable properties of the original network topology. In wireless networks, the sparsity of the topology also reduces the bandwidth contention that occurs during simultaneous wireless transmissions. Structural trimming can be performed by static and dynamic trimming.

3.1.1 *Static Trimming*[41]

Static trimming is usually performed by topology control[41]. Various localized trimming processes for unit disk graphs with known locations or with neighborhood connectivity information have been studied. Certain properties are more difficult to maintain than others. For example, maintaining a minimum shortest path tree using local location information is more involved than maintaining a minimum spanning tree.

As most existing work on trimming has been on the traditional graph, we use the time-evolving graph EG to illustrate the use of local information to trim "useless" or "redundant" nodes and links. Usually, local information (within k hops for a small k) does not cause excessive propagation delay. For example, for 2-hop information, each node maintains its own neighborhood information in EG and exchanges this information with its neighbors. When a node or link is removed, the network connectivity remains the same. More specifically, if the network is time-i-connected, it remains connected after using the following trimming rule: *node u can be trimmed if for any path $w \xrightarrow{i} u \xrightarrow{j} v$ such that $i \leq j$, there is another path, called a replacement path, $w \xrightarrow{i'} u_1 \rightarrow \ldots \rightarrow u_k \xrightarrow{j'} v$ such that $i \leq i'$ and $j' \leq j$*. Here, we only compare the edge labels of the first and last hops in these two paths. In Fig.2(c), any path $A \rightarrow D \rightarrow C$ can be replaced by a path $A \rightarrow B \rightarrow C$. For example, $A \xrightarrow{3} D \xrightarrow{6} C$ can be replaced by $A \xrightarrow{4} B \xrightarrow{5} C$. Therefore, A can ignore neighbor D. It is possible that two paths can replace each other. To avoid circular replacement, each node u is assigned a distinct priority $p(u)$. A node can be replaced only if its priority is lower than those of all intermediate nodes on the replacement path. We assume that $p(A) > p(B) > p(C) > \ldots$ is based on node IDs. We can also assign priority, say using node degree or node betweenness, based on the strategic importance of the node in the network topology.

Trimming rules can vary depending on the specific properties that are being preserved. In the current rule, the minimum completion time is preserved, but not necessarily the minimum hop count. To enforce this, we can require that each replacement path have, at most, one intermediate node. Other options include removing a specific label from a link and developing a link replacement rule. Note that the link replacement rule is a refinement of the node replacement rule. When a node is trimmed, all its adjacent links are removed as well. In situations where link labels are not deterministic but rather probabilistically known, it would be interesting to explore different probabilistic versions of the trimming rule. Clearly, more research is needed on local trimming in time-evolving graphs that maintain a given set of global properties.

3.1.2 *Dynamic Trimming*

Dynamic trimming is much more involved since its performance depends on various factors in different applications. Dynamic trimming is an online version of a trimming process based on local information only for a particular application (e.g., routing). This can lead to efficiency improvement because the decision is based on local information in a timely manner. In Fig.2(c), the path $D \rightarrow A \rightarrow B$ cannot be replaced by $D \rightarrow B$, but can be replaced at time unit 1. A more complex trimming occurs in a time-evolving graph with a probabilistic contact distribution. In a routing process in a dynamic network, should a message be forwarded at a new contact (which may lead to a less favorable path) or at a future contact (which may lead to a favorable path)? The notion of a forwarding set (a neighbor subset) is used in [42]. In this approach, the single-copy message is forwarded to a

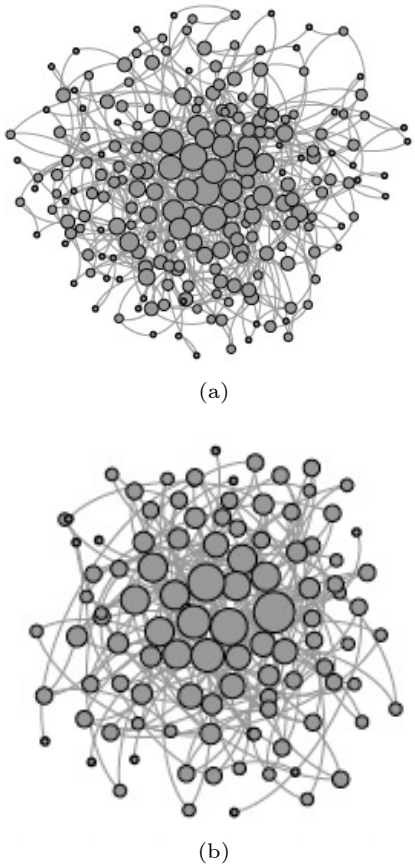
new contact if the contact belongs to the forward set (that is, the link connects to the neighbor in the set). This is analogous to a multi-bus riding: Should a passenger ride on the first bus to arrive, even though its route may be longer, or should the passenger wait for a later bus with a shorter route but an uncertain arrival time?

The answer to this question depends on the contact distribution and the settings for the objectives and constraints. For example, in a multi-copy message delivery application, the forwarding set becomes ''copy-varying'' if the objective is to minimize the delivery time of the first copy. On the other hand, if the utility of the message is time sensitive and decays over time, the forwarding set becomes ''time-varying'' when the objective is to maximize the utility. [43] shows that if the inter-contact time follows the exponential distribution and the message utility decays linearly over time, an optimal time-varying forwarding set can be derived. In fact, the forwarding set at the same intermediate node shrinks over time.

3.2 Structural Layering

The second approach is based on layering through the assignment of hierarchical levels to the nodes. Such a structure is either embedded in a given graph or is created by assigning an appropriate height to each node that will either define the levels of nodes or control the orientations of the links.

Embedded Layering. In many unstructured P2P systems, including Gnutella[44], the system structures are not only ''scale-free'' (SF) (i.e., the node degree distribution follows the power-law distribution), but also ''nested scale-free'' (NSF)[45]. SF hierarchy is defined by ranking levels based on node degrees. Let G' denote the set of subgraphs generated by iteratively removing the local nodes of the lowest degree and their connections in a network G. G satisfies the NSF if 1) G and all subgraphs in G' satisfy the SF, and 2) the standard deviation of the power-law exponents in G and all subgraphs in G' is $o(1)$. Condition 2) states that G and all subgraphs in G' are "similar" in structure, as shown in Fig.6. Fig.6(a) shows the largest strongly connected component formed in a Gnutella dataset[45, 46]. Fig.6(b) is derived from G by iteratively removing local peers to the lowest degree until only half of the peers remain. Hierarchical levels can be maintained by a labeling scheme (to be discussed in Subsection 3.3) that assigns each node a level called "height". The hierarchical structure can facilitate efficient implementations of pub-sub systems through push (moving up through the layered structure) and pull (coming down through the layered structure).



(a)

(b)

Fig.6. NSF discovered in a Gnutella dataset[45, 46]. (a) A P2P network with all peers. (b) Top 50% peers of (a).

- *Can we uncover more inherent layered structures, not only in the space dimension but also in time-and-space?*

A possible solution is to present an application in a dynamic network. A good collection of data traces that can represent time-and-space dimensions is also needed to gain insight. The work on the behavior of the small world in time and space dimensions[47] has the potential to explore the layered structure of a complex network.

Man-Made Layering. Several layered structures are made to assist certain applications. Suppose that when routing to a given destination, a destination-oriented directed acyclic graph (DAG) is maintained initially, as shown in Fig.7(a) with one destination sink D (we can ignore labels inside nodes, as will be discussed in Subsection 3.3). The advantage of maintaining a DAG is that a given source node can take any route without using a routing table to reach the destination. When a non-destination node (say node A in

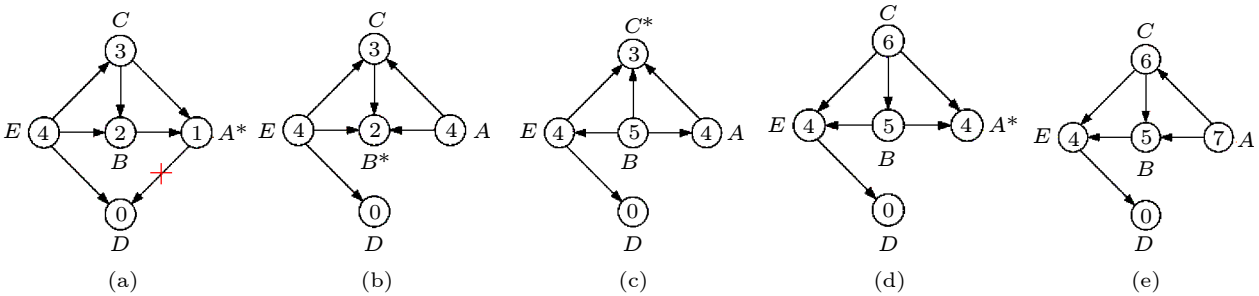
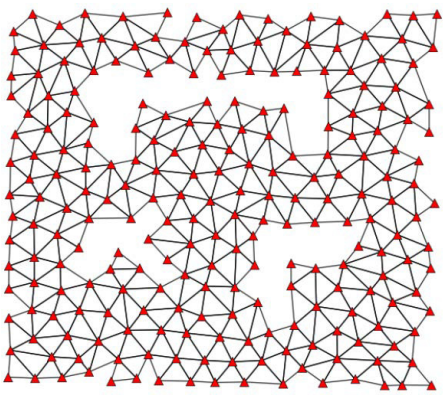
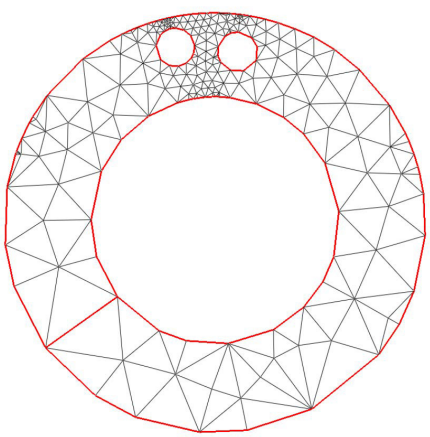


Fig.7. Full link reversal process (a) to (e) after a broken link (A, D) in destination-oriented DAG (a). Labels inside nodes correspond to heights. (a) Reversal at A. (b) Reversal at B. (c) Reversal at C. (d) Reversal at A again. (e) Destination-oriented DAG.

Fig.7(b)) has no outgoing link (node A becomes a sink) due to a broken link, it reverses all adjacent links through a full link reversal process[48]. By the reversal of a link, we mean that the orientation of the corresponding link is reversed. This may trigger the full link reversal of other nodes, as shown in Figs.7(c), 7(d), and 7(e), respectively. Eventually, the processes end up forming a new destination-oriented DAG, as shown in Fig.7(e). The orientation of each link is systematically determined by assigning appropriate heights to the two connecting nodes (details to be discussed in Subsection 3.3). Another application of dynamic destination-oriented DAGs is used to construct an efficient implementation of the classical max flow problem[49]. In this approach, the orientations of the links are dynamically calculated and adjusted by the heights of each node (except the destination sink) in multiple rounds, while maintaining the destination-oriented DAG structure. Flows associated with neighboring nodes iteratively and gradually flow among themselves, but eventually flow toward the sink, based on link orientations determined by the height of the node. The efficient maintenance of DAG in a dynamic network is certainly a challenge, as we will discuss in Subsection 3.3. A related challenge is finding an efficient way to maintain DAGs simultaneously for multiple destinations.



(a)



(b)

Fig.8. Conformal mapping[51]. (a) Graph with nonconvex holes. (b) Graph with only circles (convex) after virtual coordinate map.

3.3 Structural Remapping

In some applications, the complexity of a problem can be reduced or even removed by carefully remapping from one representation to another representation or from one domain to another.

Remapping Representation. In the Euclidean space, greedy geographic routing[50] is commonly used to greedily reduce the Euclidean distance between the source and destination. However, such a greedy process may get stuck at a local minimum, such as in one of the nonconvex holes shown in Fig.8(a). This situation happens when the bottom-right node sends a message greedily to the top-left node and the message becomes stuck at a nonconvex hole; it cannot make further progress towards the destination. By mapping the Euclidean space to the hyperbolic space, [52] shows that carefully assigning each node a virtual coordinate in the hyperbolic plane allows the greedy algorithm to succeed in finding a route to the destination. Conformal mapping[51] using Ricci flow

guaranties the delivery of greedy forwarding. A key idea is to transform holes into perfect circles which are convex, as shown in Fig.8(b), to avoid being stuck at an intermediate node during a greedy routing process.

• *Instead of mapping from one representation to another, such as when mapping the Euclidean space to the non-Euclidean space, can we remap a problem from one domain to a different domain?*

Fourier analysis, which converts a signal from its original time or space domain to a representation in the frequency domain, provides a good example of how to address this question. The Fast Fourier Transform (FFT) is usually used to compute the discrete Fourier transform of a sequence. The more recent GNNs use a similar approach that adopts the spectral theory for learning and analysis[53]. We show one potential mobile application below through remapping.

Remapping Domain. In mobile applications in MANETs and VANETs, due to the high mobility of the nodes, the contacts are highly irregular. It is difficult to derive a useful structure for dynamic graphs. [54] observes that in social contact networks, node (personal) contacts in the physical layer are influenced by the corresponding social features in the social relationship layer. As confirmed from several real traces, including INFOCOM 2006[55] and MIT Reality Mining[10], the frequency of the personal contacts of two nodes (persons) is dependent on their feature distance. The closer the distance, the higher the contact frequency. Here, each person is represented by a social feature profile. Social features represent either physical features, such as gender, or logical ones, such as occupation. In Fig.9, each person is described by three characteristics: gender (male and female), occupation (professional and student) and nationality (say country 1, country 2, and country 3). Suppose that we group all individuals with the same features in one node. Two nodes are connected if they differ in exactly one feature; a generalized hypercube is generated. In a generalized hypercube, the number of nodes along a dimension can be more than two.

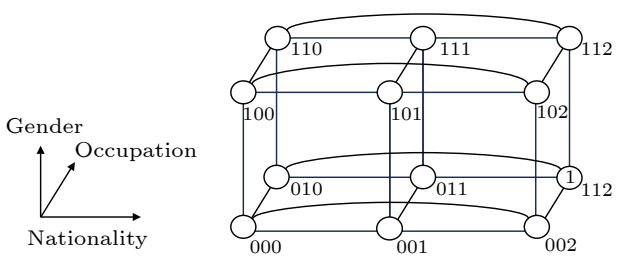


Fig.9. (a) Static and (b) structured feature space as a 3-D generalized hypercube[54].

In this way, we convert a routing process in a highly mobile and unstructured contact space (M space) to one in a static and structured feature space (F space) represented as a generalized hypercube. A generalized hypercube can easily support shortest-path routing as well as node-disjoint multiple-path routing. Note that links in a generalized hypercube correspond to strong links (terminology commonly used in social networks) between nodes with one feature difference. Each node corresponds to a community of people with common feature and the most frequent contacts. Although links that are not in a generalized hypercube do exist, they correspond to weak links with lower probabilities for their larger feature distance.

The above example shows an application of the influence of social networks on the underlying contact network. As more applications are related to social networks, we need to address the following question.

• *How can we systematically uncover the influence that social relationships have on the structure of an underlying network for socially-rich applications?*

More data traces are needed to validate any hypothesis on the influence that social relationships and features have on the underlying network contact structure.

4 Learning Through Labeling and Coding

We study here special distributed and localized labeling and coding as a way of learning to uncover relevant structures. They typically involve nodes that interact with others through the passing of messages in a restricted vicinity. Each node usually performs simple tasks, such as maintaining and propagating information labels. Together, these nodes achieve the desired global objective. Again, a localized solution is a distributed solution in which there is no sequential propagation of information. A set of labels is usually used to describe nodes at different levels or with different link orientations. It is assumed that each node knows the k-hop information (or the local horizon in the theoretical community) for a small constant k. Each node has a distinct ID within the k-hop for symmetry breaking. Note that a centralized solution

[10] https://en.wikipedia.org/wiki/Reality_mining, Nov. 2025.

can be converted to a distributed solution. For example, Dijkstra's shortest path can be implemented in a distributed way; each leaf node will report to the root its distance information at each relaxation round. The root will inform which leaf node corresponds to the shortest path from the root to all potential new leaves. The propagation back and forth between the root and the leaves is not efficient because it requires multiple rounds of information exchanges.

In the following Subsections 4.1–4.5, we will first focus on static and dynamic labeling, respectively, followed by a subsection on coding, which bears some resemblance to GNNs. We pose some challenging questions on capture structures in a dynamic network environment. Finally, we take a position that uses self-stabilization as a possible way to address these challenges with an example of lightning network applications.

4.1 *Static Labels*

The static labeling process refers to a labeling process in which each node is labeled a small number of times for a given network topology. We use colors such as white, gray, and black to distinguish different labels to simplify our discussion.

The connected dominating set (CDS)[11] is frequently used to select a virtual backbone in wireless networks and MANETs[56] for efficient routing and transmission. A set in G is a dominating set (DS) (or a core) if any node in G not in the set is a neighbor of a node in the set. By ensuring connectivity, CDS can support efficient broadcasting and, at the same time, reduce transmission collisions caused by retransmissions in wireless networks. In a general graph, finding a minimum CDS is NP-hard. In a special graph, such as interval graphs, the minimum CDS can be solved linearly[57]. In a localized CDS construction for a given general graph, two colors are used: black for CDS nodes and white for non-CDS nodes. Initially, all nodes are white. If a node has two unconnected neighbors, it is labeled black. All black nodes form a CDS. In Fig.10, all nodes except A are labeled black. A trimming (or pruning) process can be applied locally to change the black color back to the white color if the neighborhood of a black node is covered by other connected black nodes with higher priority[58]. In Fig.10, E and F are changed to white after the trimming process, assuming the priority as follows: $p(A) > p(B) > p(C) \ldots$

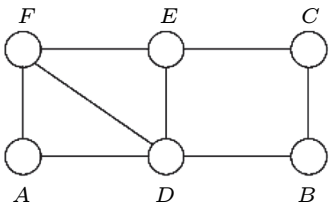


Fig.10. Graph sample for static labeling for DS, CDS, and MIS.

The maximal independent set (MIS)[12] has many applications in CS that avoid conflicts in resource allocations, including wireless channels. A set in graph G is an independent set (IS) if two nodes in the set are not neighbors in G. Initially, all nodes are white. If a node is the local 1-hop maximum (in terms of priorities) among white neighbors, it is colored black (and becomes a clusterhead). A and B are colored black. A node with a black neighbor is labeled gray as C, D, and F (and is removed from the next round of competition). This process repeats until there is no white node. The final MIS in Fig.10 is A, B, and E, all colored black. The color of each node does not have to be self-determined (i.e., selected by the node itself). It can also be neighbor-designated (i.e., selected either by a neighbor or the node itself) like in the following localized DS calculation that uses only one round: each node selects one winner (say, the one with the highest priority) from its 1-hop neighborhood including itself. A node is colored black if it is selected by at least one node. This process ends in one round. In Fig.10, A, B, and C are selected as DS. In this case, A is selected by A, D, and F. B is selected by B, C, and D. C is selected by C and E. One question can be posed: Is there another efficient, localized labeling scheme besides self-determined and neighbor-designated?

Labels are frequently used to represent levels in the node hierarchy. For nested scale-free (NSF) networks[45], labels are given iteratively as follows. Initially, all nodes are unassigned. An adjusted node degree is defined as the number of unassigned neighbors. In the first round, nodes that are local minimum in terms of the adjusted node degree are assigned to level 1. These nodes are assigned. We repeat the above process and adjust the level by adding 1, until all nodes are assigned. Note the difference between

[11] https://en.wikipedia.org/wiki/Connected_dominating_set, Nov. 2025.
[12] https://en.wikipedia.org/wiki/Maximal_independent_set, Nov. 2025.

Figs.11(a) and 11(b) in terms of the hierarchy of nodes. We aim to derive a structure with only one node at the top level. In NSF, there may still be multiple top-level nodes that are not connected to each other. In this case, it is assumed that an external server is used to connect them.

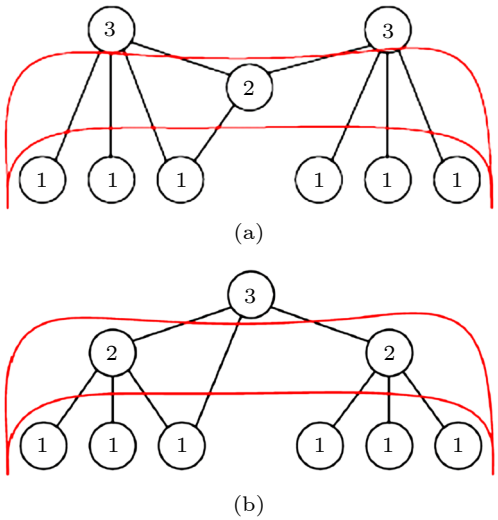


Fig.11. Labeling based on (a) node degree and (b) nested node degree.

4.2 *Dynamic Labels*

The dynamic labeling process refers to a labeling process in which several nodes are repeatedly labeled a large number of times. By large, we mean a non-constant number. The Bellman-Ford algorithm[59] maintains the shortest path and distance information from each node to a destination. Each distance estimation at a node can be considered a labeling process which involves many rounds of routing table update in case of a link failure. PageRank[36] and HITS (also known as hubs and authorities)[60] are two other examples of dynamic labeling used to rank websites.

In link reversal[48], the levels associated with the nodes correspond to the heights of the nodes in such a way that a DAG is always maintained to the sole destination as the sink. In Fig.7, each node is initially assigned a distinct level, called height, with the destination being the lowest level: 0. The orientation of each link is defined by the relative heights of the two end nodes. The node with the highest height points to the node with the lowest height. Suppose that a connected digraph is no longer a destination-oriented DAG. Then, there are sinks other than the destination. We can simply raise the levels of these sinks, but only so that they are higher than their highest neighbors by 1. All links associated with these nodes reverse their links in full link reversal. A link reversal can cause ripple effects and trigger neighboring nodes to reverse their links. In fact, each node can participate in multiple rounds of reversals, such as node A in Fig.7. In general, the number of reversals is $O(n^2)$, where n is the number of nodes. This high cost in slow convergence is detrimental to the practicality of this approach. Although partial link reversal[48] improves performance by reversing a subset of links at each reversal, it does not reduce the overall complexity.

We pose two questions: 1) Are there any other ways to represent labels? 2) Can a label be associated with a link? The answer to both questions is "yes". An elegant result uses a binary label associated with each link as a label for a link reversal[61]. Initially, a destination-oriented DAG is maintained. Two rules are used on non-destination sinks due to broken links. This approach not only unifies the full link reversal (1 for all links initially and using rule 1 only) and the partial link reversal (0 for all links initially and using rule 1 and rule 2), but also provides a complexity analysis.

4.3 Coding

We can explore two fronts. The first is to design a hybrid centralized-and-distributed method that can enjoy the merits of both centralized and distributed solutions. The key issue is how a centralized solution can provide some guidance to a distributed one. The Internet routing offers some ideas. The distributed solution (the distance vector using the Bellman-Ford solution) is used in intra-domain routing, whereas the centralized (but implemented in a distributed way) path vector is applied in inter-domain routing.

The software-defined network (SDN) control over distributed routing also offers some interesting information[62]. The proposed architecture achieves both flexibility and robustness by controlling distributed routing; it inserts fake nodes and links to create an augmented topology for a distributed solution.

The second front is devising a hybrid distributed-and-localized method that has the problem-solving capability of a distributed solution and uses static labels. The work in [63] sheds some light on such a labeling method, designed to support optimal fault-tolerant routing in an n-dimensional binary hypercube (or n-D cube) with faulty nodes. Instead of maintaining labels for routing capabilities to a particular desti-

nation, it maintains a label that indicates the routing capability to a set of destinations within a hop-count through a unique coding method. The distributed labeling process enjoys some of the properties of a localized solution. This labeling method codes faulty information that propagates only to the affected region.

Let $S(u) = k$, an integer, be the safety status of node u, where k is referred to as the level of safety (which is also a measure of routing capability), and u is called k-safe. A faulty node is 0-safe, which corresponds to the lowest level of safety, while an n-safe node (also referred to as a safe node) corresponds to the highest level of safety.

Safety Level. Let $(S_0, S_1, \ldots, S_{n-1})$ be the ascending safety-level sequence of node u's n neighboring nodes in an n-cube, such that $S_i \leq S_{i+1}$, $0 \leq i \leq n$. The safety level of node u is defined as: if $(S_0, S_1, \ldots, S_{n-1}) \geq (0, 1, 2, \ldots, n-1)$, then $S(u) = n$, else if $(S_0, S_1, \ldots, S_{k-1}) \geq (0, 1, \ldots, k-1)$ and $S_k = k-1$, then $S(u) = k$.

Specifically, if a node is labeled i, then it can find a shortest route to any nodes within i hops, and there is at least one node $i+1$ hops away that cannot be reached through a shortest path, i.e., a Hamming-distance path. When the safety level of a node is n in an n-D cube, this node can reach any node through a shortest path since the diameter of an n-D cube is n. This node is called a "safe" node. The safety level of a node and the safety levels of its neighbors satisfy a special constraint[13]. The safety level of a node is determined iteratively on the basis of the safety levels of its neighbors by rounds of exchange-and-update. Initially, all faulty nodes have a level 0 (the lowest) and all non-faulty nodes have a level n (the highest). Unlike link reversal, each safety level is decided, at most, once. In fact, if the safety level of a node is i, then the level of this node is decided exactly in round i. As the diameter of an n-D cube is n, at most $n-1$ rounds are needed. Table 2 shows the process of calculating the safety level of node 0101 and its four neighbors in Fig.12 in four rounds.

Table 2. Safety Level Calculation of Node 0101 and Its Four Neighbors in Fig.12

Node ID	Init.	R_1	R_2	R_3	R_4
0101	4	4	2	2	2
1101	4	4	4	4	4
0001	4	1	1	1	1
0111	4	1	1	1	1
0100	0	0	0	0	0

Safety levels not only provide information about routing capabilities, but also guide the routing process. No routing table is needed for optimal routing. Fig.12 shows a 4-D hypercube with four faulty nodes (colored black). At an intermediate node of the self-guided process, the next hop is the highest safety-level neighbor selected from a subset of neighbors. This subset includes only neighbors that are on the shortest paths from the intermediate node to the given destination. The binary addresses of these neighbors are closer to the destination by one binary bit. In Fig.12, node 1101 selects neighbor 0101 (with a safety level of 2) between two neighbors 1001 and 0101 on the route to 0001. Table 2 shows details of how safety levels are updated in the round. Note that the safety levels are stabilized as soon as the safety levels remain unchanged between two adjacent rounds, as R_2 and R_3 in Table 2. In general, safety levels show a delicate balance between efficiency (in terms of rapid building of structures) and utility (in terms of structural utility). The application of safety

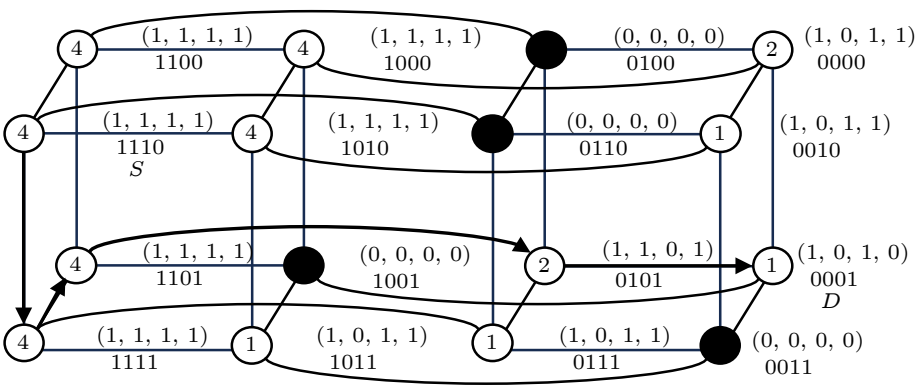


Fig.12. Safety levels (inside each node) and safety vectors in a 4-D binary hypercube[64].

[13] A constraint based on the minimal level assignment across neighbors.

level has been used in optimal fault-tolerant broadcast. The model has been extended to more sophisticated binary safety vectors[64] and extended safety vectors[65]. The way in which binary safety vectors[64] are defined resembles GNN with each node u associated with a status vector $\boldsymbol{S}(u)$, called safety vectors $\boldsymbol{S}(u) = (u_1, u_2, \ldots, u_n)$ in a binary hypercube n-D. Each safety vector is calculated on the basis of the neighbors' safety vectors through message exchanges. In the k-th message exchange, u_{k+1} is determined based on the k-th element of neighbors in their safety vectors.

Safety Vector. Initially, all faulty nodes have the safety vector of $(0, 0, \ldots, 0)$. u_1 is set to 0 if it is adjacent to a faulty link; otherwise, it is set to 1. Suppose $(u_1^{(i)}, u_2^{(i)}, \ldots, u_n^{(i)})$ is u's neighbor's safety vector along dimension i,

$$u_{k+1} = \begin{cases} 0, & \text{if } \sum\limits_{1 \le i \le n} u_k^{(i)} \le n - k - 1, \\ 1, & \text{otherwise.} \end{cases}$$

Unlike GNNs where the feature vectors are fed to an NN, the safety vectors are directly mapped to the routing capability of the node u in the hypercube. Specifically, if $u_k = 1$, u can reach any k-hop node through the shortest path. In general, the safety vector provides more precise information than the safety level. In Fig.12, node 0001 is associated with the vector $(1, 0, 1, 1)$ instead of safety level 1. This means that node 0001 can reach any nodes in 1-hop, 3-hop (in this case, nodes 1011, 1010, and 1100), and 4-hop (node 1000) through the shortest paths. In fact, the safety level corresponds to the length of the longest non-zero prefix of the corresponding safety vector.

Extended safety vectors[65] go one step further by capturing fault information within d distance in a precise way. It has been shown that $d = 2$ is sufficient to capture the information of the faulty link in a natural way. The fault information outside d hops is captured in a special coding just like a binary safety vector.

4.4 Challenges

In the theoretical community, a large amount of research has been done on the merits and limitations of distributed[66] and localized solutions[67]. We discuss here some additional challenges, starting with one fundamental question:

- *In a complex network, especially in a dynamic environment, how can we deal with the complexity of building a structure along with the change of topology?*

In a dynamic network, network topology and node status change. How can the construction of an underlying structure be better integrated with changes? Unless these changes are infrequent, many solutions will become useless if they cannot be executed quickly enough. However, a quick execution may not offer optimal results. Localized algorithms seem to be the answer, but have limitations in terms of obtaining a competitive approximation ratio[68]. An open discussion of the relative merits of an asymptotic bound analysis vs an average case bound is needed. In some cases, an asymptotic bound analysis has a hidden large constant coefficient. An average case bound, on the other hand, has some rare bad cases but is generally competitive. The notion of average case itself is a subject of research. Smoothed analysis[69] gives a promising approach to expressing the average case through a natural model of noise or perturbation in a given distribution.

Mobility will create another serious problem: view inconsistency. In a mobile application, both neighborhood information exchanges (for k-hop information) and asynchronous Hello message exchanges cause delays, which will generate inconsistent neighborhood and location information. Some early results on maintaining multiple views[70] seem promising, but more work is needed to address the general challenge in this area.

Another promising area is the integration of the process of building a structure with a change of topology due to node movement. This approach is different from most existing approaches where structure rebuilding occurs after a topology change. A simpler version of topology change is to add and remove nodes and/or links that frequently appear in sensor networks. [71] shows that although the construction of an MIS requires $\log n$ rounds, if the MIS is built based on a graph with random priority nodes, an addition/delete operation requires one round of adjustment in expectation.

Another challenge is the convergence time for the dynamic labeling process.

- *How do we handle the long convergence time for the dynamic label that usually occurs in a distributed solution?*

A centralized solution is efficient and flexible, but is less scalable and suffers from a single point of failure. A localized solution supports parallel executions

for scalability and is quick to respond to changes because local information does not propagate. However, localized solutions are limited in problem solving capabilities[68]. Distributed solutions are fault-tolerant and scalable and offer good problem solving capabilities. They suffer from slow convergence, as shown in the Bellman-Ford algorithm for distributed routing and in the link reversal for computing a destination-oriented DAG. A related question is the following: Many structures that are constructed through labels and codes are done in a static graph.

• *How can they reconstruct and reorganize in a seamless way, especially in a dynamic and changing environment?*

4.5 Self-Organization

We address two challenges raised in the previous subsection. In *Swarm Intelligence*: *From Natural to Artificial Systems*, Bonabeau, Dorigo, and Theraulaz[72] defined a notion of "self-organization" in social insects (including humans) as a set of dynamical mechanisms where structures appear at the global level from interactions at the lower level. This ability to leverage decentralized interactions between individual insects (or agent) to achieve coordinated and adaptive responses is the key capability in agentic AI[73] and embodies AI[74]. Prehofer and Bettstetter[75] further refined the concept with four principles:

- P_1: local interactions with global properties,
- P_2: minimizing the maintained state,
- P_3: adaptive to changes (self-healing),
- P_4: implicit coordination.

Here, we use the construction of a lightweight overlap network[37] on top of lightning networks (LNs)[38] to illustrate these principles. LN was proposed to address the scalability issue of the blockchain. Using smart contracts, trusted neighbors (or simply neighbors) in LNs can set up micropayment channels (or simply channels) that support instant transactions without blockchain confirmation. Such a transaction can be done via neighbors directly or non-neighbors indirectly with a path of microchannels connecting these nodes. However, since each user (node) has to commit a certain amount of money to the adjacent channel to a trusted friend, LNs face the liquidation issue. In Fig.13(a), A commits \$5 on channel (A, C). If A can transfer \$4 to B via C, the remaining funds on (A, C) and (C, B) become (\$1, \$5) and (\$0, \$7), respectively. Suppose A and B become trusted friends and A reallocate \$2 to the channel connecting to B as shown in Fig.13(b), the transfer \$4 from A to B must be done in a more complex way through multiple paths: A directly to B and A indirectly to B via C.

In [37], Wu and Jiang introduced a simple clustering method on top of an LN, where each cluster is headed by a supernode, which is trusted by its neighbors in the cluster. All funds in the cluster are pooled and managed by the supernode, which assigns funds to channels connected to other clusterheads. In Fig.13(c), supernode C pools all funds of its cluster and assigns funds to channels connected to two other supernodes E and F. The supernodes must be connected for fund circulations. In addition, as nodes come and go, ideally local changes should have a minimum effect on global status (i.e., avoiding global status changes). Supernode status is defined as follows using 2-hop view.

Supernode Selection. All nodes are initially regular nodes. A regular node that has two non-directly connected neighbors is a supernode candidate. Then, a supernode u becomes a regular node again if any

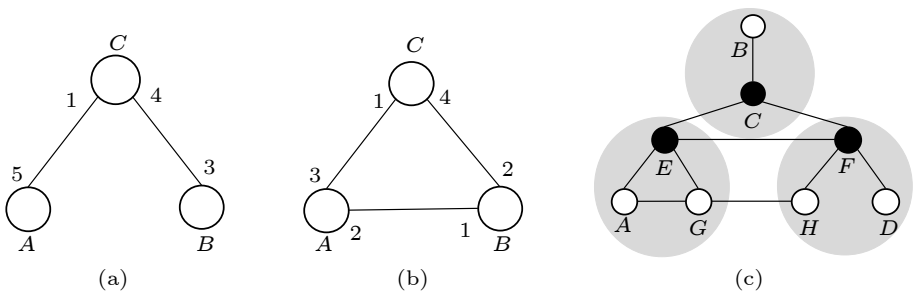


Fig.13. (a) and (b): Lightning network for scaling blockchain. Each number corresponds to the amount of fund (\$) and each node committed to a channel connecting to a trusted neighbor. (c) Illustration of supernodes and their connections to enhance liquidation through pooling funds within each cluster.

[11] https://en.wikipedia.org/wiki/Blockchain, Nov. 2025.

two neighbors of u are connected by a path (also called a replacement path) (constructed from the local 2-hop view of u, that is, the neighbor set plus neighbors' neighbor sets) such that for each node v (excluding two end nodes) on the path, $p(v) > p(u)$, where $p(\cdot)$ is the priority function.

The supernode selection is selected based on 1-hop and 2-hop neighborhood information; thus it satisfies property P_1. As each node is labeled supernode or regular node, the maintained state is minimized for the property P_2. The self-healing property of P_3 is satisfied based on the following property: each inclusion/exclusion of a node after the node joins/leaves the blockchain affects the status of the local neighborhood of two hops[37]. Finally, there is no need for synchronization among nodes, i.e., implicit coordination of the property P_4 to determine the status of the node. That is, the status of a node will not propagate beyond the neighborhood.

Fig.13(c) shows the result of supernode selection where the shade areas correspond to the areas of supernodes C, E, and F, respectively. Initially, all nodes are regular nodes. Nodes C, E, F, G, H are supernode candidates as each has two non-directly connected neighbors. G's 1-hop neighbor set includes A, E, and H and its 2-hop neighbor set includes C and F. However, G does not see B and D. Also, G does not see the link (C, F) in its 2-hop neighorhood. However, G can be replaced through a replacement path (A, E, F, H) for any pair of neighbors of G' and hence G becomes a regular node. Similarly, H can be replaced and becomes a regular node as shown in Fig.13(c) showing the final distribution of supernodes. Finally, edges can also be removed without losing connectivity[37] to further increase the liquidation of the fund transfer in LN.

5 Conclusions

In this paper, we took a snapshot of several key issues in uncovering useful structures in a complex network. We reviewed the existing graph model for a complex network and then provided a discussion of intersection graphs with two special cases: a unit disk graph along the space dimension and an interval graph along the time dimension. In addition, we discussed a general time-evolving graph in both time and space. We looked at three possible ways to build a useful structure for a complex network: trimming, layering, and remapping. We discussed distributed and localized labeling and coding as a way of learning to uncover the structure for a specific application. We offered concrete ideas and possible solutions for each approach in the hope of triggering further discussions. Finally, we listed some challenges in a dynamic and mobile environment and presented a potential distributed or localized solution based on self-organization.

The approaches discussed in this paper resemble modern graph learning to a certain extent, including message passing among neighboring nodes. The recurring exchange of neighborhood information until a stable equilibrium is reached also has some similarity to recurrent GNNs[76, 77]. In other aspects, they are different: the nested structural layering discussed in this paper dissects a network through layering rather than feature aggregation used in convolutional GNNs[53, 78]. In terms of time and space, the time-evolving graph is used here, while in graph learning, spatial-temporal GNNs[79, 80] are applied. Using NNs, GNNs are more powerful and general in discovering and classifying graph structures and nodes. Our approaches are lightweight and operate directly on graphs without resorting to NN training. They are suitable for particular graph structures and/or particular applications. Even among graph learning, there are various choices for a particular type of graphs. For example, for a unit disk graph, as node connectivity is based on proximity in a 2-D Euclidean space, the 2-D convolution in CNNs is perhaps more convenient and efficient than graph convolution in GNNs. There is still an open problem regarding the choice of cost-effective methods to discover and represent useful structures in a complex network.

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. The author declares that there are no other competing interests.

References

[1] Wu Z, Pan S, Chen F, Long G, Zhang C, Yu P S. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Networks and Learning Systems*, 2021, 32(1): 4–24. DOI: 10.1109/TNNLS.2020.2978386.

[2] Veličković P, Cucurull G, Casanova A, Romero A, Liò P, Bengio Y. Graph attention networks. In *Proc. the 6th International Conference on Learning Representations*, Apr. 30–May 3, 2018.

[3] Barabási A L, Jeong H, Néda Z, Ravasz E, Schubert A,

Vicsek T. Evolution of the social network of scientific collaborations. *Physica A: Statistical Mechanics and Its Applications*, 2002, 311(3/4): 590–614. DOI: 10.1016/S0378-4371(02)00736-7.

[4] Boccaletti S, Latora V, Moreno Y, Chavez M, Hwang D U. Complex networks: Structure and dynamics. *Physics Reports*, 2006, 424(4/5): 175–308. DOI: 10.1016/j.physrep.2005.10.009.

[5] West D B. Introduction to Graph Theory. Prentice Hall, 2001.

[6] Bondy J A, Murty U S R. Graph Theory with Applications. Macmillan, 1976.

[7] Newman M E J. Models of the small world. *Journal of Statistical Physics*, 2000, 101(3): 819–841. DOI: 10.1023/A:1026485807148.

[8] Wang C, Tang S, Yang L, Guo Y, Li F, Jiang C. Modeling data dissemination in online social networks: A geographical perspective on bounding network traffic load. In *Proc. the 15th ACM international Symposium on Mobile Ad Hoc Networking and Computing*, Aug. 2014, pp.53–62. DOI: 10.1145/2632951.2632973.

[9] Kleinberg J. The small-world phenomenon: An algorithmic perspective. In *Proc. the 32nd Annual ACM Symposium on Theory of Computing*, May 2000, pp.163–170. DOI: 10.1145/335305.335325.

[10] Wu J. Extended dominating-set-based routing in ad hoc wireless networks with unidirectional links. *IEEE Trans. Parallel and Distributed Systems*, 2002, 13(9): 866–881. DOI: 10.1109/TPDS.2002.1036062.

[11] Newman M, Barabási A L, Watts D J. The Structure and Dynamics of Networks. Princeton University Press, 2006.

[12] Wasserman S. Social Network Analysis: Methods and Applications. Cambridge University Press, 1994. DOI: 10.1017/CBO9780511815478.

[13] Malewicz G, Austern M H, Bik A J C, Dehnert J C, Horn I, Leiser N, Czajkowski G. Pregel: A system for large-scale graph processing. In *Proc. the 2010 ACM SIGMOD International Conference on Management of Data*, 2010, pp.135–146. DOI: 10.1145/1807167.1807184.

[14] Liao X F, Zhao W J, Jin H, Yao P C, Huang Y, Wang Q G, Zhao J, Zheng L, Zhang Y, Shao Z Y. Towards high-performance graph processing: From a hardware/software co-design perspective. *Journal of Computer Science and Technology*, 2024, 39(2): 245–266. DOI: 10.1007/s11390-024-4150-0.

[15] Murphy R C, Wheeler K B, Barrett B W, Ang J A. Introducing the graph 500. *Cray Users Group*, 2010, 19: 45–74.

[16] Wu J. Handbook on Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless, and Peer-To-Peer Networks. Auerbach Publications, 2005. DOI: 10.1201/9780203323687.

[17] Li D, Wu J, Liu Z, Zhang F. Towards the tradeoffs in designing data center network architectures. *IEEE Trans. Parallel and Distributed Systems*, 2017, 28(1): 260–273. DOI: 10.1109/TPDS.2016.2610970.

[18] Brandstädt A, Le V B, Spinrad J P. Graph Classes: A Survey. SIAM, 1999. DOI: 10.1137/1.9780898719796.

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

[20] 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.

[21] 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.

[22] Ferreira A. Building a reference combinatorial model for MANETs. *IEEE Network*, 2004, 18(5): 24–29. DOI: 10.1109/MNET.2004.1337732.

[23] Dijkstra E W. A note on two problems in connexion with graphs. In *Edsger Wybe Dijkstra: His Life, Work, and Legacy*, Apt K R, Hoare T (eds.), Association for Computing Machinery, 2022, pp.287–290. DOI: 10.1145/3544585.3544600.

[24] Camp T, Boleng J, Davies V. A survey of mobility models for ad hoc network research. *Wireless Communications and Mobile Computing*, 2002, 2(5): 483–502. DOI: 10.1002/wcm.72.

[25] Clementi A, Silvestri R, Trevisan L. Information spreading in dynamic graphs. In *Proc. the 2012 ACM Symposium on Principles of Distributed Computing*, Jul. 2012, pp.37–46. DOI: 10.1145/2332432.2332439.

[26] Sawwan A, Wu J. Minimizing epidemic viral total exposure under the droplet and aerosol models. In *Proc. the 2021 IEEE International Conference on Digital Health*, Sept. 2021, pp.318–326. DOI: 10.1109/ICDH52753.2021.00057.

[27] Bollobás B. Random Graphs (2nd edition). Cambridge University Press, 2011, pp.215–252. DOI: 10.1017/CBO9780511814068.

[28] Drobyshevskiy M, Turdakov D. Random graph modeling: A survey of the concepts. *ACM Computing Surveys*, 2020, 52(6): Article No. 131. DOI: 10.1145/3369782.

[29] Erdös P, Rényi A. On random graphs I. *Publicationes Mathematicae Debrecen*, 1959, 6: 290–297.

[30] Kleinberg J M. Navigation in a small world. *Nature*, 2000, 406(6798): 845–845. DOI: 10.1038/35022643.

[31] Malkov Y A, Yashunin D A. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2020, 42(4): 824–836. DOI: 10.1109/TPAMI.2018.2889473.

[32] Chung F R K. Spectral Graph Theory. American Mathematical Society, 1997. DOI: 10.1090/cbms/092.

[33] Wu J, Yang S H. SmallWorld model-based polylogarithmic routing using mobile nodes. *Journal of Computer Science and Technology*, 2008, 23(3): 327–342. DOI: 10.1007/s11390-008-9136-9.

[34] Hua Y, Miller K, Bertozzi A L, Qian C, Wang B. Efficient and reliable overlay networks for decentralized federated learning. *SIAM Journal on Applied Mathematics*, 2022, 82(4): 1558–1586. DOI: 10.1137/21M1465081.

[35] Borgatti S P, Everett M G. A graph-theoretic perspec-

tive on centrality. *Social Networks*, 2006, 28(4): 466–484. DOI: 10.1016/j.socnet.2005.11.005.

[36] Page L, Brin S, Motwani R, Winograd T. The PageRank citation ranking: Bringing order to the web. Technical Report, Stanford InfoLab, 1999. http://ilpubs.stanford.edu:8090/422/, Dec. 2025.

[37] Wu J, Jiang S. On increasing scalability and liquidation of lightning networks for blockchains. *IEEE Trans. Network Science and Engineering*, 2022, 9(4): 2589–2600. DOI: 10.1109/TNSE.2022.3166707.

[38] Poon J, Dryja T. The bitcoin lightning network: Scalable off-chain instant payments. Technical Report, Satoshi Nakamoto Institute, 2016. https://nakamotoinstitute.org/library/lightning-network/, Dec. 2025.

[39] Räcke H. Optimal hierarchical decompositions for congestion minimization in networks. In *Proc. the 40th Annual ACM Symposium on Theory of Computing*, May 2008, pp.255–264. DOI: 10.1145/1374376.1374415.

[40] Lua E K, Crowcroft J, Pias M, Sharma R, Lim S. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys & Tutorials*, 2005, 7(2): 72–93. DOI: 10.1109/COMST.2005.1610546.

[41] Santi P. Topology control in wireless ad hoc and sensor networks. *ACM Computing Surveys*, 2005, 37(2): 164–194. DOI: 10.1145/1089733.1089736.

[42] Conan V, Leguay J, Friedman T. Fixed point opportunistic routing in delay tolerant networks. *IEEE Journal on Selected Areas in Communications*, 2008, 26(5): 773–782. DOI: 10.1109/JSAC.2008.080604.

[43] Xiao M, Wu J, Liu C, Huang L. TOUR: Time-sensitive opportunistic utility-based routing in delay tolerant networks. In *Proc. the 2013 IEEE INFOCOM*, Apr. 2013, pp.2085–2091. DOI: 10.1109/INFCOM.2013.6567010.

[44] Ripeanu M. Peer-to-peer architecture case study: Gnutella network. In *Proc. the 1st International Conference on Peer-to-Peer Computing*, Aug. 2001, pp.99–100. DOI: 10.1109/P2P.2001.990433.

[45] Zheng H, Wu J. NSFA: Nested scale-free architecture for scalable publish/subscribe over P2P networks. In *Proc. the 24th IEEE International Conference on Network Protocols*, Nov. 2016. DOI: 10.1109/ICNP.2016.7784413.

[46] Leskovec J, Kleinberg J, Faloutsos C. Graph evolution: Densification and shrinking diameters. *ACM Trans. Knowledge Discovery from Data*, 2007, 1(1): Article No. 2. DOI: 10.1145/1217299.1217301.

[47] Tang J, Scellato S, Musolesi M, Mascolo C, Latora V. Small-world behavior in time-varying graphs. *Physical Review E*, 2010, 81(5): 055101. DOI: 10.1103/PhysRevE.81.055101.

[48] Gafni E, Bertsekas D. Distributed algorithms for generating loop-free routes in networks with frequently changing topology. *IEEE Trans. Communications*, 1981, 29(1): 11–18. DOI: 10.1109/TCOM.1981.1094876.

[49] Malhotra V M, Kumar M P, Maheshwari S N. An $O(|V|^3)$ algorithm for finding maximum flows in networks. *Information Processing Letters*, 1978, 7(6): 277–278. DOI: 10.1016/0020-0190(78)90016-9.

[50] Stojmenovic I. Position-based routing in ad hoc networks. *IEEE Communications Magazine*, 2002, 40(7): 128–134. DOI: 10.1109/MCOM.2002.1018018.

[51] Sarkar R, Yin X, Gao J, Luo F, Gu X D. Greedy routing with guaranteed delivery using Ricci flows. In *Proc. the 2009 International Conference on Information Processing in Sensor Networks*, Apr. 2009, pp.121–132.

[52] Kleinberg R. Geographic routing using hyperbolic space. In *Proc. the 26th IEEE International Conference on Computer Communications*, May 2007, pp.1902–1909. DOI: 10.1109/INFCOM.2007.221.

[53] Bruna J, Zaremba W, Szlam A, LeCun Y. Spectral networks and locally connected networks on graphs. In *Proc. the 2nd International Conference on Learning Representations*, Apr. 2014.

[54] Wu J, Wang Y. Hypercube-based multipath social feature routing in human contact networks. *IEEE Trans. Computers*, 2014, 63(2): 383–396. DOI: 10.1109/TC.2012.209.

[55] Scott J, Gass R, Crowcroft J, Hui P, Diot C, Chaintreau A. CRAWDAD Cambridge/haggle (v. 2006-09-15). Technical Report, IEEE DataPort, 2006. https://ieee-dataport.org/open-access/crawdad-cambridgehaggle-v-2006-09-15, Dec. 2025.

[56] Wu J, Dai F. A generic distributed broadcast scheme in ad hoc wireless networks. *IEEE Trans. Computers*, 2004, 53(10): 1343–1354. DOI: 10.1109/TC.2004.69.

[57] Tian J, Ding H. Solving minimum connected dominating set on proper interval graph. In *Proc. the 6th International Symposium on Computational Intelligence and Design*, Oct. 2013, pp.73–76. DOI: 10.1109/ISCID.2013.25.

[58] Dai F, Wu J. An extended localized algorithm for connected dominating set formation in ad hoc wireless networks. *IEEE Trans. Parallel and Distributed Systems*, 2004, 15(10): 908–920. DOI: 10.1109/TPDS.2004.48.

[59] Ford Jr L R, Fulkerson D R. Flows in Networks. Princeton University Press, 2016.

[60] Kleinberg J M. Hubs, authorities, and communities. *ACM Computing Surveys*, 1999, 31(4es): Article No. 5. DOI: 10.1145/345966.345982.

[61] Charron-Bost B, Függer M, Welch J L, Widder J. Time complexity of link reversal routing. *ACM Trans. Algorithms*, 2015, 11(3): Article No. 18. DOI: 10.1145/2644815.

[62] Vissicchio S, Tilmans O, Vanbever L, Rexford J. Central control over distributed routing. In *Proc. the 2015 ACM Conference on Special Interest Group on Data Communication*, Aug. 2015, pp.43–56. DOI: 10.1145/2785956.2787497.

[63] Wu J. Safety levels-an efficient mechanism for achieving reliable broadcasting in hypercubes. *IEEE Trans. Computers*, 1995, 44(5): 702–706. DOI: 10.1109/12.381957.

[64] Wu J. Adaptive fault-tolerant routing in cube-based multicomputers using safety vectors. *IEEE Trans. Parallel and Distributed Systems*, 1998, 9(4): 321–334. DOI: 10.1109/71.667894.

[65] Wu J, Gao F, Li Z, Min Y. Optimal, and reliable commu-

nication in hypercubes using extended safety vectors. *IEEE Trans. Reliability*, 2005, 54(3): 402–411. DOI: 10.1109/TR.2005.853439.

[66] Fich F, Ruppert E. Hundreds of impossibility results for distributed computing. *Distributed Computing*, 2003, 16(2): 121–163. DOI: 10.1007/s00446-003-0091-y.

[67] Suomela J. Survey of local algorithms. *ACM Computing Surveys*, 2013, 45(2): Article No. 24. DOI: 10.1145/2431211.2431223.

[68] Kuhn F, Moscibroda T, Wattenhofer R. The price of being near-sighted. In *Proc. the 17th Annual ACM-SIAM Symposium on Discrete Algorithm*, Jan. 2006, pp.980–989. DOI: 10.5555/1109557.1109666.

[69] Spielman D A, Teng S H. Smoothed analysis: An attempt to explain the behavior of algorithms in practice. *Communications of the ACM*, 2009, 52(10): 76–84. DOI: 10.1145/1562764.1562785.

[70] Wu J, Dai F. Mobility control and its applications in mobile ad hoc networks. *IEEE Network*, 2004, 18(4): 30–35. DOI: 10.1109/MNET.2004.1316759.

[71] Censor-Hillel K, Haramaty E, Karnin Z. Optimal dynamic distributed MIS. In *Proc. the 2016 ACM Symposium on Principles of Distributed Computing*, Jul. 2016, pp.217–226. DOI: 10.1145/2933057.2933083.

[72] Bonabeau E, Dorigo M, Theraulaz G. Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press, 1999. DOI: 10.1093/oso/9780195131581.001.0001.

[73] Wang F, Liu S. Empowering virtual agents with intelligent systems. *Communications of the ACM*, 2025, 68(8): 8–9. DOI: 10.1145/3735504.

[74] Duan J, Yu S, Tan H L, Zhu H, Tan C. A survey of embodied AI: From simulators to research tasks. *IEEE Trans. Emerging Topics in Computational Intelligence*, 2022, 6(2): 230–244. DOI: 10.1109/TETCI.2022.3141105.

[75] Prehofer C, Bettstetter C. Self-organization in communication networks: Principles and design paradigms. *IEEE Communications Magazine*, 2005, 43(7): 78–85. DOI: 10.1109/MCOM.2005.1470824.

[76] Scarselli F, Gori M, Tsoi A C, Hagenbuchner M, Monfardini G. The graph neural network model. *IEEE Trans. Neural Networks*, 2009, 20(1): 61–80. DOI: 10.1109/TNN.2008.2005605.

[77] Dai H, Kozareva Z, Dai B, Smola A, Song L. Learning steady-states of iterative algorithms over graphs. In *Proc. the 35th International Conference on Machine Learning*, Jul. 2018, pp.1106–1114.

[78] 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.

[79] Seo Y, Defferrard M, Vandergheynst P, Bresson X. Structured sequence modeling with graph convolutional recurrent networks. In *Proc. the 25th International Conference on Neural Information Processing*, Dec. 2018, pp.362–373. DOI: 10.1007/978-3-030-04167-0_33.

[80] Jain A, Zamir A R, Savarese S, Saxena A. Structural-RNN: Deep learning on spatio-temporal graphs. In *Proc. the 2016 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2016, pp.5308–5317. DOI: 10.1109/CVPR.2016.573.

Jie Wu is Chief Scientist of China Telecom and Dean of China Telecom Cloud Computing Research Institute, Beijing. Before he joined China Telecom, he was 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. He serves on several editorial boards, including IEEE/ACM Transactions on Networking. Dr. Wu is/was general chair/co-chair for IEEE IPDPS'23, ACM MobiHoc'23, and IEEE CCGrid'24 as well as program chair/cochair for IEEE INFOCOM'11 and CCF CNCC' 13. He was an IEEE Computer Society Distinguished Visitor, ACM Distinguished Speaker, and chair for the IEEE Technical Committee on Distributed Processing (TCDP). Dr. Wu is a Fellow of the AAAS and a Fellow of the IEEE. He is the recipient of the 2011 China Computer Federation (CCF) Overseas Outstanding Achievement Award. He is a Member of the Academia Europaea (MAE).