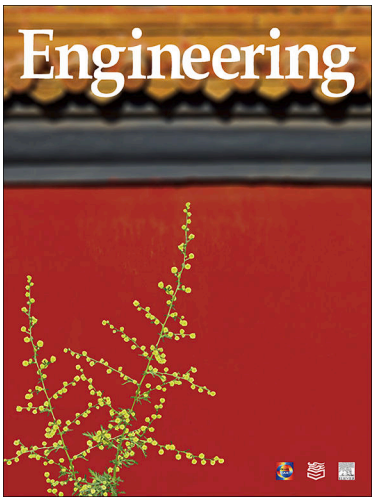
Journal Pre-proofs



Review

A Unified Framework for Combinatorial Optimization Based on Graph Neural Networks

Yaochu Jin, Xueming Yan, Shiqing Liu, Xiangyu Wang

PII: S2095-8099(26)00555-2
DOI: https://doi.org/10.1016/j.eng.2025.06.047
Reference: ENG 2469

To appear in: *Engineering*

Received Date: 18 June 2024
Revised Date: 4 February 2025
Accepted Date: 23 June 2025

Please cite this article as: Y. Jin, X. Yan, S. Liu, X. Wang, A Unified Framework for Combinatorial Optimization Based on Graph Neural Networks, *Engineering* (2026), doi: https://doi.org/10.1016/j.eng.2025.06.047

This is a PDF of an article that has undergone enhancements after acceptance, such as the addition of a cover page and metadata, and formatting for readability. This version will undergo additional copyediting, typesetting and review before it is published in its final form. As such, this version is no longer the Accepted Manuscript, but it is not yet the definitive Version of Record; we are providing this early version to give early visibility of the article. Please note that Elsevier's sharing policy for the Published Journal Article applies to this version, see: https://www.elsevier.com/about/policies-and-standards/sharing#4-published-journal-article. Please also note that, during the production process, errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

© 2026 THE AUTHORS. Published by Elsevier LTD on behalf of Chinese Academy of Engineering and Higher Education Press Limited Company

Research

AI for Process Manufacturing—Review

A Unified Framework for Combinatorial Optimization Based on Graph Neural Networks

Yaochu Jin [a], Xueming Yan [b,c,*], Shiqing Liu [d], Xiangyu Wang [d]

[a] *School of Engineering, Westlake University, Hangzhou 310024, China*

[b] *School of Information Science and Technology, Guangdong University of Foreign Studies, Guangzhou 510420, China*

[c] *Guangdong Engineering Research Center of Data Security Governance and Privacy Computing, Guangzhou 510420, China*

[d] *Faculty of Technology, Bielefeld University, Bielefeld 33501, Germany*

* Corresponding authors.

E-mail addresses: yanxm@gdufs.edu.cn (X. Yan).

Abstract: Graph neural networks (GNNs) have emerged as powerful tools for solving combinatorial optimization problems (COPs), exhibiting state-of-the-art performance in both graph-structured and non-graph-structured domains. However, existing approaches lack a unified framework capable of addressing a wide range of COPs. After presenting a summary of representative COPs and a brief review of recent advancements in GNNs for solving COPs, this paper proposes a unified framework for solving COPs based on GNNs, including graph representation of COPs, the equivalent conversion of non-graph-structured COPs to graph-structured COPs, graph decomposition, and graph simplification. The proposed framework leverages the ability of GNNs to effectively capture relational information and extract features from graph representations of COPs, offering a generic solution to COPs that can address the limitations of the state of the art in solving non-graph-structured and highly complex graph-structured COPs.

Keywords: Combinatorial optimization; Graph representation; Equivalent conversion; Graph decomposition; Graph simplification

1. Introduction

Combinatorial optimization is a combination of mathematics, operations research, and computer science [1]. Combinatorial optimization problems (COPs) are a class of problems in which the decision-maker is required to make a series of choices from a limited set of options; the goal is to find the optimal set of combinations that achieves the best outcomes. Combinatorial optimization plays a central role in tackling some of the most challenging problems across diverse domains [2–6], from logistics and transportation to resource allocation and scheduling. For example, logistics companies use combinatorial optimization to optimize delivery routes and schedules. Considering factors such as delivery locations, traffic conditions, and time windows, they can reduce transit times and fuel consumption and increase customer satisfaction by ensuring timely and efficient product delivery. In another case, educational institutions apply combinatorial optimization techniques to arrange class schedules and timetables for students. By taking into account variables such as classroom availability, teacher preferences, and student course requirements, these institutions can create schedules that minimize conflicts, optimize resource use, and foster an ideal learning environment for students.

Journal Pre-proofs

In general, a minimization COP is formulated as follows. Given a set of decision variables $X = \{x_1, x_2, \ldots, x_n\}$, where each variable represents a discrete choice or assignment, there is a finite set S of feasible solutions, each corresponding to a particular combination of the above decision variables, subject to a number of constraints $c_1, c_2, \ldots, c_k$. The COP has an objective function $f(x_1, x_2, \ldots, x_n)$ that maps each feasible solution to a real value, which is used to quantify the quality of the solution. The target is to find the best combination of decision variables $X^* = \{x_1', x_2', \ldots, x_n^*\}$ that minimizes the objective function while satisfying all the given constraints. Unlike continuous optimization problems, COPs involve variables that can take only specific, discrete values, rather than an arbitrary value within a given range. Therefore, the key challenge in solving a COP lies in identifying the most optimal solution from a large set of available choices, with the target of achieving the best outcome in terms of the defined objective function. The decision variables involved in a COP fall into multiple groupings, including binaries, integers, categories, and permutations. Various kinds of decision variables reflect different aspects of the decision-making process, such as selecting or excluding items, assigning tasks to resources, or determining the order of operations. COPs can be further classified based on the number of objectives involved in the problem. For instance, single-objective COPs focus on optimizing one objective function [7,8], while multi-objective combinatorial optimization problems (MOCOPs) often deal with multiple conflicting objectives simultaneously [9,10].

A key challenge of COPs is the potential for combinatorial explosion, meaning that the number of feasible solutions grows exponentially with the size of the problem instance. Consequently, to find the optimal solution, it is often necessary to explore a vast search space, which makes these problems computationally challenging. To cope with this issue, efficient algorithms and heuristics have been developed to find near-optimal solutions in a reasonable time. With the development of machine learning and graph representation learning technologies, graph neural networks (GNNs) have been adopted to solve traditional COPs, with remarkable success. Thanks to their ability to learn inherent graph topologies, GNNs can be trained to efficiently find optimal solutions to many traditional COPs by learning from historical data. The use of GNNs offers a novel approach to efficiently address the challenges in various COP tasks.

The main purpose of this paper goes beyond providing a review of existing work on using GNNs for solving COPs. Instead, it aims to propose a generic and unified approach to solving COPs by means of GNNs. Recently, many survey papers have been published focusing on GNNs [11–13], COPs [1,14], or a combination of both fields [15,16]. Ref. [14] focuses on machine-learning-based methods for solving COPs by highlighting applications in the telecommunications domain. Similarly, Ref. [1] investigates machine-learning-based methods for COPs, emphasizing the advantage of these methods over handcrafted heuristics. Refs. [15,16] are reviews that summarize recently published papers on the use of GNNs for COPs and propose new taxonomies.

Unlike the survey papers mentioned above, this paper proposes a unified framework for COPs based on GNNs, motivated by the hypothesis that all COPs can be represented by graphs and then solved with the help of GNNs. More specifically, some COPs can naturally be represented by graphs, such as traveling salesman problems (TSPs), while others can be converted into equivalent graph-structured formulations. Then, various GNNs can be adopted to solve different classes of graph-represented COPs, including multi-objective, constrained, and dynamic COPs. The main contributions of this work are summarized as follows:

(1) This paper introduces a novel taxonomy that categorizes COPs based on their inherent suitability to be represented by graph structures. In contrast to existing taxonomies that primarily emphasize fundamental concepts or specific applications, our taxonomy aims to align COPs with the workflow typically followed by GNNs.

(2) We divide existing methods for solving COPs into two categories: non-GNN-based approaches, which include exact solvers, approximate solvers, and machine learning methods; and GNN-based approaches. This categorization makes it easier for us to distinguish our proposed generic and unified approach from existing GNN-based methods designated for COPs that can be directly represented with graphs only.

(3) To the best of our knowledge, this is the first time a generic and unified GNN approach for solving COPs has been proposed. We show that, by converting a non-graph-structured COP into a graph-represented COP, or by decomposing a complex COP into several sub-COPs, each of which can be represented by a single graph, a unified GNN-based framework for solving all types of COPs can be established.

The remainder of this paper is organized as follows. Section 2 introduces a new taxonomy of COPs, which divides COPs into graph-structured and non-graph-structured COPs. Section 3 outlines the basic concepts and learning methods of GNNs,

Journal Pre-proofs

while Section 4 gives a brief introduction to graph-based machine learning methods for solving COPs. Section 5 proposes a unified GNN-based framework for COPs, including methods for the equivalent conversion of non-graph-structured COPs into graph-structured ones, graph simplification, and the decomposition of complex COPs represented by graphs. This is followed by a discussion of open challenges in Section 6 and a summary of the paper in Section 7.

2. A taxonomy of COPs

In this section, we provide a taxonomy of COPs based on the representation of their solutions—specifically, on whether the solutions of a COP can be directly represented by graphs or not. This distinction is crucial, as it significantly influences the selection of problem-solving approaches and optimization techniques. Graph-structured COPs incorporate graph theory into their problem-solving methods, using nodes and edges to represent variables and constraints.

Graph structures are particularly effective for problems in which relationships and interdependencies are visualized and managed. Interestingly, graph-based methods can effectively leverage the relationships between nodes and edges, making them well-suited for graph-structured COPs that naturally present themselves as graphs, such as network optimization and resource-allocation issues. In such cases, the interactions between elements are often complex, and a graph structure can clearly capture these relationships, enabling efficient solutions through methods such as GNNs. In contrast, non-graph-structured COPs address problems in which data relationships do not naturally form a network. These problems often involve the equivalent conversion of functions based on a set of variables without explicit interconnections that can be depicted as graphs, such as certain optimization or decision-making processes. COPs that do not exhibit a graph structure must be converted into a graph structure. This transformation typically involves extracting key elements from the problem and establishing the relationships between them, resulting in a graph that accurately reflects the actual problem. This conversion then makes it possible to leverage the advantages of graph structures and apply a series of relevant GNN methods to effectively solve the problem. Therefore, by clearly categorizing COPs into graph-structured and non-graph-structured COPs, we can more effectively select appropriate GNN methods to adapt to different types of COP problems. As illustrated in Fig. 1 [17–58], this taxonomy not only helps in understanding the fundamental differences between graph-based and non-graph-based COPs but also serves as a guide for selecting appropriate algorithms and techniques based on the problem structure.

Journal Pre-proofs

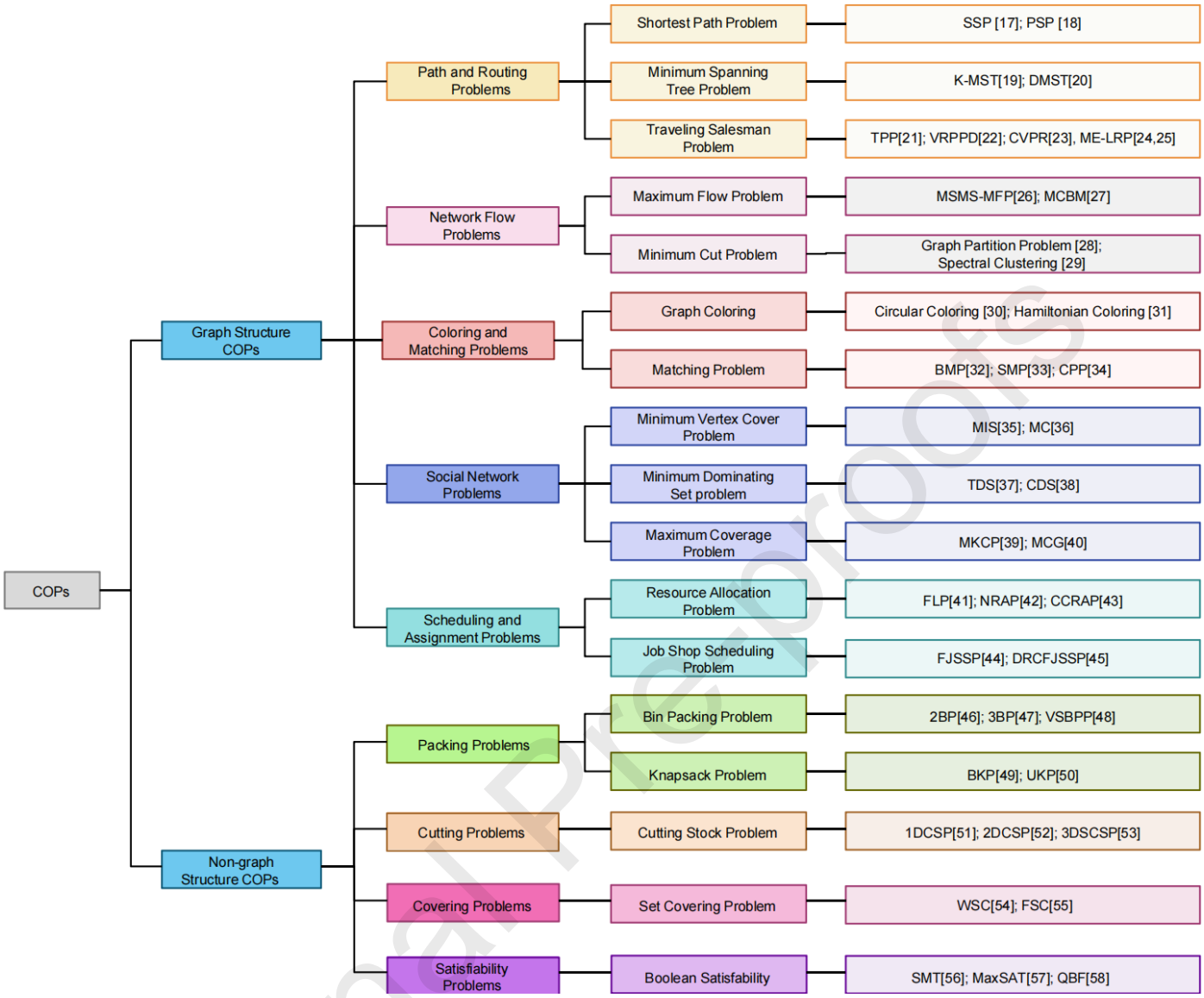


Fig. 1. A taxonomy of COPs. SSP: single-source shortest path problem (SP); PSP: all-pairs SP; *k*-MST: *k*-minimum spanning tree problem (MST); DMST: dynamic MST; TPP: traveling purchaser problem; VRPPD: vehicle routing problem (VRP) with pickup and delivery; CVRP: capacitated VRP; ME-VRP: multi-echelon VRP; MSMS-MFP: multi-source multi-sink maximum flow problem; MCBM: maximum cardinality bipartite matching; BMP: bipartite matching problem; SMP: stable marriage problem; CPP: Chinese postman problem; MIS: maximum independent set; MC: maximum clique; TDS: total dominating set; CDS: connected dominating set; MKCP: maximum *k*-coverage problem; MCG: maximum coverage problem with group budget constraints; FLP: facility location problem; NRAP: network resource-allocation problem; CCRAP: cloud-computing resource-allocation problem; FJSSP: flexible job shop-scheduling problem; DRCFJSSP: dual resource-constrained FJSSP; 2BP and 3BP: two-dimensional (2D) and three-dimensional (3D) bin packing problem, respectively; VSBPP: variable-sized bin packing problem; BKP: bounded knapsack problem; UKP: unbounded knapsack problem; 1DCSP, 2DCSP, and 3DCSP: one-dimensional (1D), 2D, and 3D cutting stock problem, respectively; WSC: weighted set cover; FSC: fractional set cover; SMT: satisfiability modulo theories; MaxSAT: maximum satisfiability problem; QBF: quantified-Boolean formula.

2.1. Graph-structured COPs

2.1.1. Path and routing problems

Path and routing problems are common types of COPs that involve finding the most efficient ways to navigate through a graph, considering various constraints and objectives. We introduce some classical path and routing problems, including the shortest path problem (SP), the minimum spanning tree problem (MST), and the TSP.

Definition 1: The shortest path problem. Given a graph $G = (V,E)$, where V and E denote the sets of vertices and edges, respectively, the shortest path problem aims to find an optimal path between two vertices such that the sum of the weights of its constituent edges is minimized, where a path (P) in an graph is a sequence of vertices $P = (v_1, v_2, ..., v_n) \in V \times V \times \cdots \times V$, such that v_i is adjacent to v_{i+1} for $1 \leq i < n$.

Some variations of the SP, such as the single-source shortest path problem (SSP) [17] and the all-pairs shortest path problem (PSP) [18], are also presented. These variations help to optimize routes and aid in decision-making for real-world applications. In particular, the SP has practical significance across multiple domains. For instance, it is used in road networks for navigation [59], path planning for autonomous robots [60], efficient data transmission in computer networks [61], and the optimization of supply-chain and logistics operations [62].

Definition 2: The minimum spanning tree problem. Given an edge-weighted undirected graph, span a tree that connects all vertices while not having any cycles and that has a minimum total edge weight.

One representative example [63] involves a communication network striving for efficient data transmission and network design, where the MST is utilized to create a network with the minimum total communication cost. A related problem is the k-minimum spanning tree problem (k-MST) [19], which entails finding an MST that spans a specified subset of k vertices in the graph, while minimizing the total weight of the tree. Additionally, the dynamic minimum spanning tree problem (DMST) [20] addresses the challenge of updating a previously computed MST in response to changes in the underlying graph. These changes can include modifications to edge weights, the insertion of new vertices, or the deletion of existing vertices.

Definition 3: The traveling salesman problem. Given a list of cities and the distances between each pair of cities in a complete weighted graph $G = (V, E)$, find a tour with the minimum total weight that visits each city exactly once and returns to the origin city.

The TSP can be modeled as an undirected, weighted graph. It is a minimization problem in which the goal is to start and finish at a fixed vertex, while visiting every other vertex exactly once. The traveling purchaser problem (TPP) [21] and the vehicle routing problem (VRP) are generalizations of the TSP, both of which belong to the class of non-deterministic polynomial (NP)-hard problems. The TPP, in particular, is a procurement/routing problem that involves the strategic selection of a purchasing plan for a specified set of products from a subset of suppliers, coupled with the determination of an optimal visiting tour for the purchaser. This is done to efficiently satisfy a predefined demand for products. The VRP pertains to the operations of a delivery company, in which goods are distributed from one or more depots. Each depot has a set of home vehicles operated by drivers who navigate a given road network to serve a set of customers. Several variations and specializations of the VRP exist, addressing different operational constraints and objectives. These include the vehicle routing problem with pickup and delivery (VRPPD) [22], the capacitated vehicle routing problem (CVRP) [23], the multi-depot vehicle routing problem (MDVRP) [64], the electric vehicle routing problem (EVRP) [65], and the multi-echelon vehicle routing problem (ME-VRP) [24,25].

2.1.2. Network flow problems

Network flow problems are a class of COPs that deal with the flow of resources through a network. The input of these problems is a flow network, which is a graph with numerical capacities assigned to its edges. The objective of a network flow problem is to establish a flow by assigning numerical values to each edge, ensuring adherence to the capacity constraints. Additionally, the flow must satisfy the condition of having an incoming flow equal to the outgoing flow at all vertices, except for specifically designated terminals such as sources and sinks. Two notable types of network flow problems are the maximum flow problem (MFP), which focuses on finding the maximum flow from the source to the sink, and the minimum cut problem, which involves identifying the minimum capacity cut that separates the source and sink.

Definition 4: The maximum flow problem. Let $G = (V,E)$ be a network with a source $s \in V$ and a sink $t \in V$. The value of the flow is denoted by f, which represents the amount of flow passing from the source to the sink. The MFP is to determine the maximum flow value, denoted as f_{max}, that can be routed from the source to the sink within the network.

As a fundamental problem in network flow optimization, the MFP [66] has been extended to address more complex flow-related challenges, such as the multi-source multi-sink maximum flow problem (MSMS-MFP) [26] and maximum cardinality bipartite matching (MCBM) [27]. Unlike the MFP, the MSMS-MFP involves multiple source nodes and multiple sink nodes within the network [26]. For example, when applied to Internet routing, the MSMS-MFP optimizes the flow of data packets

between multiple sources and sinks to improve network performance, reduce latency, and ensure efficient resource utilization [67]. As a classical flow problem, MCBM deals with finding the largest possible matching in a bipartite graph. In such graphs, vertices can be divided into two disjoint sets, and a matching consists of edges that do not share any common vertices [68]. MCBM provides a versatile framework for modeling and solving problems involving optimal pairings or assignments between two distinct groups.

Definition 5: The minimum cut problem. Given a network $G = (V,E)$ with a source s and a sink t, a cut C partitions the vertices V into two disjoint sets, S and T (i.e., $S \cup T = V$ and $S \cap T = \emptyset$), such that $s \in S$ and $t \in T$. The minimum cut problem aims to find a cut with the minimum possible capacity among all the cuts in the network.

Variations of the minimum cut problem encompass a wide range of scenarios, including weighted graphs, directed graphs, scenarios with designated terminals, and the partitioning of vertices into more than two sets. For instance, the graph partition problem [28], a recognized NP-hard challenge, involves partitioning a given graph into two or more parts. This process adheres to additional constraints, such as achieving a balance in the sizes of the parts created by the cut. Additionally, spectral clustering [29,69,70], a graph-based clustering method, leverages the spectral properties of matrices derived from the data. This method aims to find a partition that minimizes criteria related to the cut in the graph, where the cut refers to the sum of the weights of the edges connecting different partitions.

2.1.3. Coloring and matching problems

Coloring and matching problems involve constraints related to the relationships between vertices or edges. In graph coloring [71], colors are assigned to vertices with the constraint that adjacent vertices must not share the same color. The optimization goal in graph coloring is to minimize the number of colors used. In contrast, the matching problem [72] revolves around the selection of edges in such a way that no two chosen edges share a common vertex. The objective can be to maximize or minimize the number of edges in the matching, depending on the specific requirements of the problem.

Definition 6: Graph coloring. Given an undirected graph $G = (V, E)$, where V is a set of vertices and E is a set of edges, a graph coloring is an assignment of colors to the vertices of G such that no two adjacent vertices share the same color and the number of colors required to achieve a valid coloring is minimized.

There are various graph coloring variants, each based on different constraints and objectives, such as clique coloring [73], circular coloring [30], and Hamiltonian coloring [31]. These variants offer solutions to complex problems across different fields by capturing specific relationships and arrangements within graphs. Clique coloring aims to assign colors to vertices such that cohesive groups, or cliques, within a graph are represented by the same color. This approach has applications in network design [74], resource allocation [75], and conflict resolution [76]. Circular coloring, which involves assigning colors to vertices arranged in a circular layout, is commonly applied in frequency assignment problems [77] and scheduling scenarios [78]. Hamiltonian coloring focuses on ensuring that no two adjacent vertices along a Hamiltonian cycle share the same color, which finds applications in circuit design [79] and routing problems [80].

Definition 7: The matching problem. Given an undirected graph $G = (V, E)$, where V represents the set of vertices and E denotes the set of edges, a matching M in G is a subset of edges such that no two edges in M share a common vertex.

Commonly referred to as the bipartite matching problem (BMP) [32], this problem seeks to determine the largest possible set of non-overlapping edges in a graph, where each edge connects vertices from the two distinct partitions of the bipartite graph. In a weighted bipartite graph, the optimization problem aims to find a maximum-weight matching, in what is also known as the assignment problem [81]. It is noteworthy that, while the goal of the assignment problem is to maximize weights, it does not inherently guarantee stability. However, it has a wide range of applications, from project management [82] to logistics [83]. In comparison, the stable marriage problem (SMP) seeks a stable matching between two equally sized sets of elements, where each set has an ordering of preferences [33]. In addition, the Chinese postman problem (CPP) [34] aims to find a minimum-weight perfect matching in general graphs as a subproblem. When a graph has a Eulerian circuit, it inherently solves the CPP for that graph. However, in cases where a Eulerian circuit is absent, the optimization objective is to identify the minimum number of edges that must be duplicated (or the subset of edges with the minimum total weight) to transform the graph into a multigraph with a Eulerian circuit [84]. The CPP has proven to be a versatile and powerful tool in optimization, with several combinatorial problems being effectively reduced to it [85].

2.1.4. Social network problems

Social network problems within the class of graph-based COPs leverage the powerful framework of graph theory to model and address various challenges and optimization goals in social network contexts. Typical problems in this area include the minimum vertex cover (MVC), minimum dominating set (MDS), and maximum coverage problem (MCP). Each of these problems presents unique challenges that help in understanding and optimizing different aspects of social networks.

Definition 8: The minimum vertex cover. Given an undirected graph $G = (V, E)$, find a set $S \subseteq V$ such that, for every edge $(u,v) \in E$, where (u,v $\in V$) are its endpoints, at least one of u or v is in S. The MVC aims to minimize the cardinality of the vertex cover set, denoted as $|S|$.

The maximum independent set (MIS) [35] and maximum clique (MC) [36] are closely related variants of the MVC problem in graph theory. Vertex cover optimization serves as a model for a wide range of real-world and theoretical problems. For example, the distribution of facilities in municipal services that ensures that a set of monitor stations adequately covers the connections between other stations may be modeled as a vertex cover minimization problem [86].

Definition 9: The minimum dominating set. Given an undirected graph $G = (V, E)$, find a dominating set $D \subseteq V$, such that every vertex in V is either in D or adjacent to a vertex in D. The MDS aims to minimize the cardinality of the dominating set, denoted as $|D|$.

Both the total cominating set (TDS) [37] and the connected dominating set (CDS) [38] are common variants of the MDS that are extensively studied in the context of algorithmic problem-solving, especially in the design and optimization of networks [87]. In a TDS, every vertex in the graph must either be included in the set or be adjacent to a vertex that is part of the set. The goal is to find the smallest possible TDS for the graph. In contrast, a CDS requires that the set forms a connected subgraph, ensuring that every vertex in the set is either adjacent to or can be reached by some other vertex in the set. CDSs are particularly relevant in the fields of network design and communication protocols [88], where connectivity is crucial.

Definition 10: The maximum coverage problem. Given a weight graph $G = (V, E, w)$ where w:$E \rightarrow \mathbb{R}^+$, find a set $S \subseteq V$ that maximizes the total weight of edges connected to vertices in V.

The maximum k-coverage problem (MKCP) [39] is a typical extension of the MCP in which an additional constraint limits the number of subsets (sets) that can be chosen. Instead of aiming to cover as many elements as possible, the goal is to select a subset of k sets to maximize coverage. The key constraint here is the fixed number k, which acts as a budget or limitation on the number of subsets that can be selected. In addition, by introducing an additional constraint related to the budget associated with each group (subset), the MCP can transition to the MCP with group budget constraints (MCG) [40]. In the MCG, each subset (group) is associated with a budget, and the goal is to maximize the coverage of elements subject to the constraint that the total budget spent on selected groups does not exceed a specified limit.

2.1.5. Scheduling and allocation problems

Scheduling and allocation problems are concerned with the effective allocation of resources, tasks, and time and are commonly modeled as COPs, with graph models being a prevalent representation method. However, not all problems require the use of graph models, and employing other mathematical models or algorithms is sometimes more convenient or efficient [89]. In this context, we focus on graph-based COPs, such as resource-allocation problems (RAPs) [90] and job shop-scheduling problems (JSSPs) [91].

Definition 11: The resource-allocation problem. Given a directed graph $G = (V, E)$, where the vertices V may represent tasks, projects, or other entities and the directed edges E represent relationships or dependencies between them, the RAP is to minimize or maximize a performance metric related to the efficiency or satisfaction of the resource allocation when the resource demand for each node is satisfied and resource allocations are non-negative.

The optimization of the RAP [92,93] is typically formulated as a mixed-integer non-linear programming problem that is non-convex and NP-hard. The facility location problem (FLP) is a classic RAP that involves determining the optimal locations for facilities to meet demand in the most cost-effective way [41]. In addition, a series of extensions of RAP, such as the network resource-allocation problem (NRAP) [42], the human-resource-allocation problem (HSAP) [90], and the cloud-

computing resource-allocation problem (CCRAP) [43], demonstrate the adaptability of RAPs in different application domains.

Definition 12: The job shop-scheduling problem. Given N_J independent jobs, J_1 ,J_2 ,...,J_{N_J}, and a processing time for each job of p_k for $k = 1, 2, ..., N_J$, the JSSP is about the scheduling of a set of independent tasks to optimize a certain performance metric. A mathematical definition of the JSSP is given as follows:

Minimize: A performance metric (e.g., total completion time or makespan)

Subject to

$$S_k \geq \max(C_l)\ \forall k = 1, 2, ..., N_J \quad (1)$$

$$S_k + C_k = T_k\ \forall k = 1, 2, ..., N_J \quad (2)$$

$$T_k \leq T_l\ \forall k, l = 1, 2, ..., N_J \text{ and } k \neq l \quad (3)$$

where each job can only be processed on one machine, and the start time of each job cannot precede the completion time of its predecessors. S_k and T_k denote the start time and completion time of job J_k, respectively, and p_k denotes its processing time.

A JSSP can be usually formulated as a graph optimization problem, depending on the nature of the scheduling problem. Nodes in the graph often represent tasks or events, and edges between nodes may represent dependencies or constraints between tasks. The complexity of scheduling problems arises from the need to efficiently allocate limited resources to meet specific objectives. As technology evolves and societal priorities shift, scheduling problems are being adapted to incorporate new considerations and objectives, evolving toward greater flexibility, human-centricity, and sustainability [94]. For example, flexible job shop-scheduling problems (FJSSPs) [44] introduce a set of jobs organized into operations that need to be processed on a sequence of machines, unlike the single-machine focus of classical, non-flexible environments. Often, an FJSSP that incorporates the human element is referred to as a dual resource-constrained flexible job shop-scheduling problem (DRCFJSSP), indicating the simultaneous consideration of both machines and workers as critical resources in the scheduling process [45].

2.2. Non-graph-structured COPs

2.2.1. Packing problems

Packing problems [95] attempt to pack objects into containers with the goal of either maximizing occupancy within a single container or minimizing the total number of containers required to accommodate all objects. Two typical examples of the packing problems are the bin packing problem (BPP) [96] and the knapsack problem [97].

Definition 13: The bin packing problem. Given a finite set of items $I = 1, 2,... ,N_I$, each with a positive size s_i, where i represents the item, and a set of identical bins or containers, each with a fixed positive capacity B, the objective of the BPP is to minimize the number of bins used to pack all items while adhering to the capacity constraint of each bin.

The BPP is known to be NP-hard and involves packing a finite set of items, each with a weight, into a finite number of bins [98]. Various practical variations of the BPP exist, mainly depending on the dimensions of the bins, placement constraints, and priorities. For example, evolved versions of the BPP that incorporate an additional dimension, such as width or height, include the two-dimensional bin packing problem (2BP) [46], the three-dimensional bin packing problem (3BP) [47], and the variable-sized bin packing problem (VSBPP) [48]. In addition to the traditional BPP, a notable variant is the online bin packing problem (OBP), where items with varying volumes arrive sequentially, and decisions about each item must be made in real time. Here, the decision-maker faces the choice of either selecting and packing the current item into the available bins or letting it pass [99].

Definition 14: The knapsack problem. Given a set of n items, indexed as $i = 1, 2, ... ,n$, where each item i has an associated weight w_i and a value v_i, and given a knapsack with a fixed weight capacity W, the objective is to maximize the total value of items placed in the knapsack without exceeding its weight capacity.

The knapsack problem is similar to a scenario in which someone has a fixed-size knapsack and needs to decide which items should be placed into it to maximize the total value. Each item has a weight and a value, and decisions are based on the knapsack's capacity constraints [100]. The 0-1 knapsack problem [101] is one of the most common and well-known variants of this problem; two other prevalent variants are the bounded knapsack problem (BKP) [49] and the unbounded knapsack problem (UKP) [50]. In the BKP, multiple copies of each item are available, but there is a limit on how many copies of each item can be selected. The decision variable now represents the number of copies of item i to include, with integer values ranging from 0 to a specified upper bound. Conversely, the UKP allows an unlimited number of copies of each item, simplifying the decision-making process. These variants illustrate the adaptability of the knapsack problem to model diverse real-world scenarios with varying constraints on the number of item copies that can be selected [102].

2.2.2. Cutting problems

Cutting problems, specifically cutting stock problems (CSPs), present optimization challenges that focus on efficiently cutting large sheets or stock lengths of raw material into smaller pieces to meet specific demands or requirements.

Definition 15: The cutting stock problem. Given a set of $\mathbb{N}_C$ items or raw materials, each with a specific length, a set of raw material rolls or stock lengths, each with a fixed length, and the demand for each item, indicating the number of pieces needed, the CSP is to minimize the number of raw material rolls used to satisfy the demand for the items.

The CSP is an NP-hard optimization challenge that arises in industrial applications and can be related to the knapsack problem [103]. It can be classified based on the dimensionality of the cuts, including the one-dimensional cutting stock problem (1DCSP) [51], the two-dimensional cutting stock problem (2DCSP) [52], and the three-dimensional cutting stock problem (3DCSP) [53]. The CSP challenge is relevant in numerous industrial processes and involves a diverse array of materials such as steel bars, rolls of paper or aluminum, wooden boards, metal sheets, printed circuit boards, glass or fiberglass sheets, and leather, among many others. Effective cutting strategies are crucial for reducing costs and increasing efficiency. Consequently, the CSP holds significant economic importance due to its direct impact on cost savings through optimized material utilization [104].

2.2.3. Covering problems

Covering problems aim to find a subset of elements that cover or satisfy a certain criterion. The set cover problem (SCP) is a well-known COP with a variety of applications.

Definition 16: The set cover problem. Given a finite set of elements, denoted as U, where $U = 1, 2, ..., \mathbb{N}_U$, a collection of subsets of U, denoted as $\mathfrak{S} = \mathfrak{S}_1, \mathfrak{S}_2, ..., \mathfrak{S}_m$, where each $\mathfrak{S}_i$ represents a subset of U, and a cost associated with selecting each subset $\mathfrak{S}_i$, the objective of the SCP is to minimize the total cost by selecting a minimum number of subsets such that the union of the selected subsets covers all elements in U.

The SCP is known to be NP-hard in the strong sense and poses challenges from the perspective of theoretical approximation [105]. Typically, the SCP is characterized as the challenge of efficiently covering the rows of an $\mathbb{N}_U$-row, an m-column, and a zero–one matrix (a_{ij}) using a subset of the columns while minimizing costs. This versatility can be extended to two notable variants, the weighted set cover (WSC) [54] and fractional set cover (FSC) [55]. In the WSC, each set is associated with a positive weight, symbolizing its cost, with the aim of identifying a set cover with the minimum total weight, while the FSC permits the selection of fractions of sets, rather than whole sets. The SCP holds practical significance, as it can be applied to model a diverse array of real-world problems, including but not limited to scheduling [106], manufacturing [107], service planning [108], and information retrieval [109].

2.2.4. The satisfiability problem

The satisfiability problem (SAT) aims to determine the satisfiability of a logical formula—that is, whether there exists an assignment of truth values to variables that makes the formula true. When each clause contains at most two literals (binary clauses), the problem is known as a Boolean satisfiability problem (2-SAT).

Definition 17: The Boolean satisfiability problem. Given a Boolean formula ϕ in conjunctive normal form (CNF), which is a conjunction (AND) of clauses, each clause is a disjunction (OR) of literals. A literal is either a variable x_r or its negation

$\neg x_r$, where r is an index. The aim of the 2-SAT is to determine whether there exists an assignment of truth values (true or false) to the variables in ϕ that makes the entire formula true.

The SAT is an NP-complete decision problem that involves determining whether a propositional logic formula can be satisfied with appropriate value assignments to its variables [110]. It has impacted various related decision and optimization problems, which are referred to as SAT extensions. One notable extension of the SAT is satisfiability modulo theories (SMT) [56]. Other extensions include maximum satisfiability (MaxSAT) [57] and quantified-Boolean formulas (QBF) [58]. These extensions represent paradigmatic constraint-satisfaction problems with broad practical applications, including hardware and software design [111] and bioinformatics [112], among others.

3. Graph neural networks

This section begins by introducing the fundamental concept of GNNs. Next, representative GNNs are presented in detail. Finally, GNNs are categorized and reviewed based on their learning methods—namely supervised, unsupervised, and reinforcement learning (RL).

3.1. Basic concept

Most machine-learning-based methods focus on solving data in the Euclidean space [113,114], such as text or images. These regularly structured data types encapsulate information about the context or position of elements, which are organized consistently. Unlike Euclidean data, many real-world problems can be represented as graph-structured data, such as scheduling [115], recommendation systems [116], and biology networks [117], among others. Graph-structured data exhibits more complex structures within a non-Euclidean space, typically arising from varying connections between nodes (also called "samples" in other machine learning methods); this requires the development of techniques capable of extracting structures.

3.1.1. Graph embedding

In the past decade, various graph embedding (also called "network embedding") methods have been proposed to convert the nodes in a graph (or network) into dense and low-dimensional vectors while preserving the structure [118]. The basic idea of obtaining vector representations in low-dimensional space is that the original topology information, such as the often-used adjacency matrix, is sparse and discrete, which is not particularly effective for subsequent processing. One natural idea is to apply matrix factorization, such as singular value decomposition, to reduce the dimension of the adjacency matrix. Alternatively, random-walk-based methods have been explored [119], inspired by the natural language processing community. In DeepWalk [120], one sequence, referred to as one "sentence" in Word2Vec [121], is derived by implementing a random walk on nodes, referred to as "words" in Word2Vec. Subsequently, the Skip-Gram model [121] proposed in Word2Vec is employed to learn the embedding vectors of nodes by considering the context in one sentence and the co-occurrence rate among sentences to describe the local topology information of a node. Node2Vec [122] employs a biased random walk strategy that allows for a choice between breadth-first sampling and depth-first sampling. Various methods incorporating additional information, such as node labels or attributes, have also been proposed to enhance the representation ability [123,124]. Apart from approaches that use random walk strategies to extract structure information, LINE [125] generates embedding vectors of nodes by focusing on the connection between two nodes. The obtained embedding vectors of nodes are further used for downstream applications, including node classification [126], link prediction [127], recommendation [128], and visualization [129], to name a few.

Graph embedding methods aim to provide dense, continuous, and low-dimensional embeddings as the input to other off-the-shelf machine-learning methods to handle downstream tasks. These methods excel in specific scenarios but encounter significant limitations when it comes to inductive learning, which requires the model to effectively generalize to unseen data. Moreover, traditional graph embedding techniques often encounter challenges with exacting high-level representations, since they typically rely on random walk methods that may not capture the complexity inherent to graph data.

3.1.2. The GNN timeline

Recently, GNNs have attracted increased attention in the machine learning community due to their ability to handle complex graph data more effectively than traditional graph embedding methods. GNNs perform graph representation learning to extract and learn more precise and meaningful representations, thereby better reflecting the relational properties of the

data. More specifically, GNNs update node or edge embeddings by iteratively aggregating and combining information from their neighbors to support downstream tasks such as node or graph classification and link prediction.

Fig. 2 [130–163] provides a timeline of milestones in GNN research since 2009, when the term "graph neural network" was first introduced [130]. During the same period, a related concept referred to as neural networks for graphs (NN4G) was proposed [131]. Both frameworks provide a general idea for updating node embeddings through the exchange of information with neighboring nodes. After that, GNNs failed to attract widespread attention until around 2015 [132,133], then two fundamental GNNs—the graph convolutional network (GCN) [134] and ChebNet [135]—were published. In 2017, research on GNNs entered a new era with numerous GNNs being proposed, including graph sample and aggregate (GraphSAGE) [136], message-passing neural networks (MPNNs) [137], and graph attention networks (GATs) [138]. Subsequently, numerous studies have built on the classic GNNs mentioned above [139–142] and integrated these networks with other machine learning techniques [143,144,164,165], such as large language models [166]. Since 2020, research on GNNs has increasingly aimed to address their limitations, such as neighbor explosion problems [145–147], generalization problems [148,149,167], homophily assumptions [150,168–170], limited discrimination power [151,152], and hardware limitations and efficiency [171]. In the meantime, significant efforts have been made to apply GNNs to a wide range of practical, real-world challenges, such as dynamic graph-based problems [153–155,172], attack and defense issues [156], biologically based problems [157–159,173], billion-scale recommender systems [160], and others [161–163]. As mentioned above, the prevailing research trends in GNNs primarily focus on overcoming their inherent limitations, such as scalability, generalization, explainability, and interpretability [174]. Furthermore, there is ongoing development of advanced model architectures and their application across diverse scenarios, including COPs [175], protein-structure prediction, network-based recommendation [176], and forecasting [177].

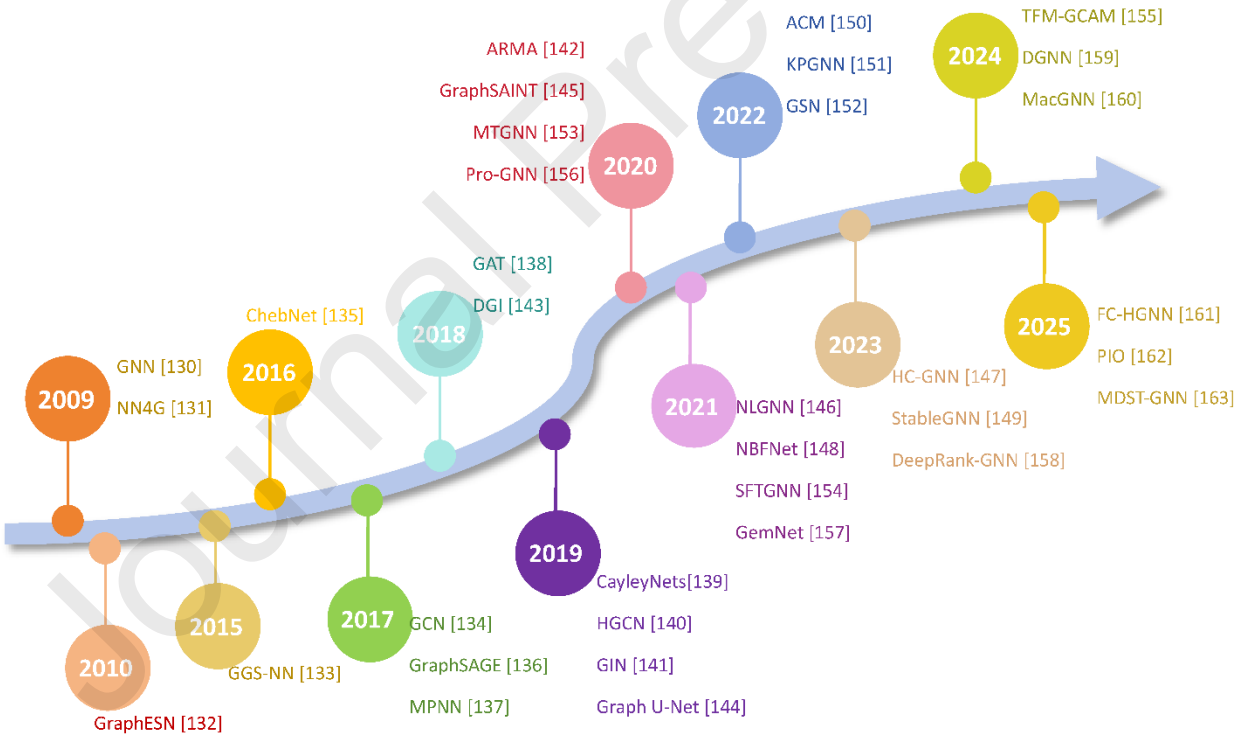


Fig. 2. Timeline of the development of GNNs. NN4G: neural networks for graphs; GraphESN: Graph Echo State Networks; GGS-NN: gated graph sequence neural network; GCN: graph convolutional network; GraphSAGE: graph sample and aggregate; MPNN: message-passing neural network; GAT: graph attention network; DGI: deep graph infomax; HGCN: hyperbolic graph convolutional neural network; GIN: graph isomorphism network; ARMA: auto-regressive moving average; GraphSAINT: graph sampling based inductive learning method; MTGNN: multivariate time series GNN; Pro-GNN: Property GNN; NLGNN: non-local GNN; NBFNet: neural Bellman–Ford network; GemNet: geometric message passing neural network; SFTGNN: spatial–temporal fusion GNN; ACM: adaptive channel mixing; KPGNN: *K*-hop message passing GNN; GSN: graph substructure network; HC-GNN: hierarchical community-aware GNN; TFM-GCAM: traffic flow matrix-based graph

convolution network with attention mechanisms; MacGNN: macro GNN; DGNN: DNA-origami-based GNN; FC-HGNN: heterogeneous GNN built on brain functional connectivity; PIO: probabilistic improvement optimization; MDST-GNN: multi-distance spatial–temporal GNN.

Without loss of generality, GNNs can be classified into two categories: spectral-based approaches and spatial-based approaches [16,178]. Using graph signal procession theory, new spectral-based GNNs are proposed by designing graph filters, while spatial-based GNNs are aimed at exploring the local topology by introducing novel aggregation or message-passing methods. Below, we provide a brief review of representative GNNs according to this taxonomy. Additionally, to specify the methods shown in Fig. 2 that fall between spectral-based and spatial-based GNNs, we highlight the names of spectral-based GNNs. Spectral-based methods have been less popular in recent years due to their comparatively limited scalability and flexibility. However, they remain a crucial aspect of GNNs, as they hold significant theoretical importance.

3.1.3. Spectral-based GNNs

Spectral-based GNNs are based on graph signal processing theory [179], which mainly relies on graph Fourier transformation, inverse graph Fourier transformation, and matrix transformation. The basic idea of spectral-based GNNs is to conduct convolution operations on the input data such that the frequency within a certain range in the frequency domain is enhanced by graph filters.

Given a graph with an adjacency matrix $\boldsymbol{A}$and a node degree diagonal matrix $\boldsymbol{D}$, then the normalized graph Laplacian matrix is

$$\boldsymbol{L} = \boldsymbol{I}_n - \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{\mathrm{T}} \quad (4)$$

where $\boldsymbol{I}_n$ denotes the $n \times n$ identity matrix, $\boldsymbol{\Lambda}$ and $\boldsymbol{U}$ are the diagonal matrix of the eigenvalues and its corresponding matrix of the eigenvectors, respectively. Since $\boldsymbol{L}$ is symmetric positive semi-definite, the columns of $\boldsymbol{U}$ are orthogonal to each other, and $\boldsymbol{U}^{\mathrm{T}}\boldsymbol{U} = \boldsymbol{I}_n$. We assume that there is a graph signal $x \in \mathbb{R}^n$ and a graph filter g; then, the graph convolution operation can be expressed as follows:

$$x *_G g = \mathcal{F}^{-1}(\mathcal{F}(x) \odot \mathcal{F}(g)) \quad (5)$$

where $*_G$ denotes the operator on graph G, $\odot$ represents the element-wise product, the graph Fourier transformation $\mathcal{F}(x) = \boldsymbol{U}^{\mathrm{T}}x$, and the inverse graph Fourier transformation $\mathcal{F}^{-1}(\hat{x}) = \boldsymbol{U}^{\mathrm{T}}\hat{x}$. Here, $\hat{x}$ represents the graph signal in the spectral domain. Therefore, the convolution operator can be rewritten as

$$x *_G g_\theta = \boldsymbol{U}\left(\boldsymbol{U}^{\mathrm{T}}x \odot \boldsymbol{U}^{\mathrm{T}}g\right) = \boldsymbol{U}g_\theta\boldsymbol{U}^{\mathrm{T}}x \quad (6)$$

where $g_\theta = \mathrm{diag}(\boldsymbol{U}^{\mathrm{T}}g)$, and the θ in g_θ represents learnable parameters in the graph filter g. It should be noted that the structure information $\boldsymbol{A}$ of a graph is contained in the matrix $\boldsymbol{U}$.

In ChebNet [135], the researchers define a graph filter as $\sum_{k=0}^{K}\theta_k T_k(\tilde{\boldsymbol{\Lambda}})$ where k denotes the polynomial order index of the Chebyshev polynomial here in this section, K is the maximum polynomial order, θ_k is the learnable coefficient, $\tilde{\boldsymbol{\Lambda}} = 2\boldsymbol{\Lambda}/\lambda_{\max} - \boldsymbol{I}_{\mathrm{n}}$, and $\lambda_{\max}$ is the largest eigenvalue of $\boldsymbol{L}$. $T_k(\tilde{\boldsymbol{\Lambda}})$ represents the Chebyshev polynomials, which are obtained by $T_k(\tilde{\boldsymbol{\Lambda}}) = 2\tilde{\boldsymbol{\Lambda}}T_{k-1}(\tilde{\boldsymbol{\Lambda}}) - T_{k-2}(\tilde{\boldsymbol{\Lambda}})$. Therefore, the convolution operator in ChebNet is

$$x *_G g = \boldsymbol{U}\left(\sum_{k=0}^{K}\theta_k T_k(\tilde{\boldsymbol{\Lambda}})\right)\boldsymbol{U}^{\mathrm{T}}x = \sum_{k=0}^{K}\theta_k T_k(\tilde{\boldsymbol{L}})x \quad (7)$$

where $T_k(\tilde{\boldsymbol{L}}) = 2T_k(\boldsymbol{L})/\lambda_{\max} - \boldsymbol{I}_n$. By applying the Chebyshev polynomial with different K values, the model can capture information at different ranges of neighbors of a node in a graph.

Later, the GCN [134] was proposed to consider a special case of ChebNet when $K = 1$ and $\lambda_{\max} = 2$. Thus, the graph filter is written as

$$x *_G g = \theta(\boldsymbol{I}_{\mathrm{n}} + \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2})x \qquad (8)$$

However, the researchers found that the GCN will be more stable upon replacing $\boldsymbol{I}_n + \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$ with $\tilde{\boldsymbol{D}}^{-1/2}\,\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-1/2}$, where $\tilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}_{\mathrm{n}}$ and $\tilde{\boldsymbol{D}}$ is the diagonal degree matrix of $\tilde{\boldsymbol{A}}$. A closer look at Eq. (8) shows that the graph filter can be treated as an aggregation of the mean information from one node's neighbor ($\boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$) together with its own embedding ($\boldsymbol{I}_n$); thus, the GCN establishes a bridge between spectral-based GNNs and spatial-based GNNs.

Recently, a large body of research has been proposed based on the idea of designing effective graph filters. A simple graph convolution (SGC) has been proposed that contains a simple fixed low-pass filter, resulting in competitive experimental results, better scalability, and improved computational efficiency [177,180]. Later, a novel frequency adaptive graph convolutional network (FAGCN) [181] was proposed to learn both low-frequency and high-frequency signals.

3.1.4. Spatial-based GNNs

Spatial-based GNNs aim to directly aggregate the neighbors' information of each node to one node through edges, which can be considered as a message-passing strategy. Due to the intuitive characteristics of passing messages through edges, spatial-based GNNs are more interpretable. Additionally, spatial-based GNNs do not need to solve the products and inverses of large matrices, making them efficient in handling large-scale graphs. For these reasons, spatial-based GNNs are attracting increasing research attention [179].

Scholars have proposed a general framework of message-passing strategies for MPNNs [137], as follows:

$$\boldsymbol{m}_v^{t+1} = \sum_{u\in N(v)} \boldsymbol{M}_t\left(\boldsymbol{h}_v^t, \boldsymbol{h}_u^t, e_{vu}\right) \qquad (9)$$

$$\boldsymbol{h}_v^{t+1} = \boldsymbol{M}_t\boldsymbol{U}_t\left(\boldsymbol{h}_v^t, \boldsymbol{m}_v^{t+1}\right) \qquad (10)$$

where $\boldsymbol{h}_v^t$, $\boldsymbol{h}_u^t$, and e_{vu} are the embedding vectors of node v, node u in the tth hidden layer, and the edge features, respectively, and t denotes the index of the hidden layer here. $N(v)$ refers to a set of neighbors of node v, and $\boldsymbol{m}_v^{t+1}$ represents the aggregated embedding vector, which encompasses the embeddings of neighbors and their corresponding edges. $\boldsymbol{M}_t$ and $\boldsymbol{U}_t$ are two learnable matrices in the tth hidden layer, which are optimized through back-propagation. In general, the operators in Eq. (9) are considered to be aggregation by the propagation of information among neighbors, while Eq. (10) combines information aggregated from neighbors and target nodes by generating a new embedding vector. Various GNNs are proposed with different aggregation or combination methods. Spatial-based GNNs aggregate information from neighbors in an iterative way; that is, T-hop information could be obtained by a GNN with T hidden layers, where T denotes the number of hidden layers here. However, research has found that over-smoothing may occur when the number of hidden layers is large, as a result of the similarity of the embeddings of all nodes in a graph after long-distance neighbor information exchange. Multiple methods have been introduced to address this issue, such as skip connections [182], neighborhood re-definition [183], and adaptive aggregation [184], to name just a few.

Later, GNNs based on attention methods emerged as highly promising approaches, drawing considerable interest within the research community. In contrast to other aggregation methods, attention mechanisms assume that the importance of neighbor nodes to a target node is different and should be decided by both the target node and its corresponding neighbor nodes. By learning the importance of nodes, attention-based methods have shown promising results in many fields [185,186]. Here, we introduce a novel work, the GAT [138], which has served as a baseline for several follow-up works. The embedding of node v in the $(t+1)$th hidden layer is updated as follows:

$$\boldsymbol{h}_v^{t+1} = \sigma\left(\sum_{u\in N(v)\cup v} \alpha_{vu}^{t+1}\boldsymbol{W}^{t+1}\boldsymbol{h}_u^t\right) \qquad (11)$$

where $\boldsymbol{h}_u^t$ and $\boldsymbol{h}_v^{t+1}$ are the embedding vectors of the neighbors in the tth hidden layer and the target node in the $(t+1)$th hidden layer, respectively. σ is the activation function, $\boldsymbol{W}^{t+1}$ is a learnable matrix in the $(t+1)$th hidden layer, and α_{vu}^{t+1} is the attention parameter of node u to node v, which can be obtained as follows:

$$\alpha_{vu}^{t+1} = \frac{\exp(\mathrm{LeakyReLU}(\boldsymbol{a}^{\mathrm{T}}[\boldsymbol{W}^{t+1}\boldsymbol{h}_v^{t+1} || \boldsymbol{W}^{t+1}\boldsymbol{h}_u^{t+1}]))}{\sum_{u\in N(v)\cup v}\exp(\mathrm{LeakyReLU}(\boldsymbol{a}^{\mathrm{T}}[\boldsymbol{W}^{t+1}\boldsymbol{h}_v^{t+1} || \boldsymbol{W}^{t+1}\boldsymbol{h}_u^{t+1}]))} \qquad (12)$$

where $\boldsymbol{a}$ is a learnable vector. To generate several α_{vu}^{t+1} at one time by learning different $\boldsymbol{W}^{t+1}$, the GAT can be expanded to a multi-headed GAT, increasing the expression ability of the original GAT. Following this work, many other definitions and calculation methods have been proposed to express node neighbors' different degrees of importance [187–189].

3.2. Training algorithms for GNNs

Depending on the way in which GNNs learn from data, learning algorithms can be classified into supervised learning, where the model learns from labeled data; unsupervised learning, where the model explores patterns and structures in unlabeled data; and RL, where the model refines its behavior through interactions with an environment and by learning from trials and errors. The best learning approach for solving COPs depends on the nature of the problem, the availability of data, and the specific goals of the optimization.

3.2.1. Supervised training

Supervised learning stands out from other learning algorithms primarily due to the presence of labeled samples, which are used to enable a GNN to learn a mapping from input data to corresponding output labels. A loss function is usually designed to minimize the discrepancy between the predicted output generated from the GNN model and the true labels. For COPs with known solutions, supervised learning can be employed as a training method. The most commonly used loss functions include mean squared error (MSE) for graph regression tasks, logistic loss for binary classification tasks, and cross-entropy loss for multilabel classification tasks. Additionally, custom loss functions may be developed to consider the specific characteristics of a COP, thereby further increasing the solution's effectiveness and efficiency. In COPs, custom penalty terms are often used to enforce constraint satisfaction. For instance, a fundamental constraint in the TSP is that each city must be visited exactly once. To ensure compliance with this rule, penalty terms are integrated into the loss function. The penalty is set to infinity if a solution violates this constraint and to 0 otherwise [190]. In this way, the custom loss function tailored to the specific nature of the COP, along with the supervised loss function, increases the precision of the learning process.

When a GNN is adopted for combinatorial optimization, a supervised learning algorithm can be used to achieve an end-to-end GNN model for one-shot solution prediction [191], often in combination with a local search method or a heuristic search method. Refs. [191,192] apply GNNs with supervised learning to assess the existence of solutions under specific constraints for the graph colouring problem (GCP) and TSP, respectively. Refs. [193–195] use supervised learning to solve graph-matching problems. Since the solutions generated by GNNs in an end-to-end fashion may not be feasible or may be sub-optimal, local or heuristic search methods are usually used to fine-tune or improve the quality of the obtained solutions. For example, for the TSP, the output of a GNN will be a probability matrix instead of a solution; therefore, beam search can be conducted [196]. More specifically, the GNN model outputs a probability matrix in which the element p_{ij} represents the probability of selecting city j as the next destination from city i. Note that p_{ij} is different from the p_k in definition 12. Beam search then utilizes this probability matrix to construct feasible solutions by maintaining a fixed number of partial solutions at each step. Starting from an initial city, beam search iteratively expands these partial solutions based on the probabilities provided by the GNN. At each step, it considers all possible next moves for each partial solution, scoring them based on the probabilities. Then the top-scoring partial solutions are selected for the next iteration. This process repeats until complete solutions are constructed.

3.2.2. Unsupervised training

Unlike supervised learning, the unsupervised training of GNNs aims to uncover the structure of data without predefined labels, making it suitable for multi-model COPs with multiple optimal solutions and for COPs with no available ground-truth labels. The choice and design of the loss function are crucial, since the loss function serves as a guiding metric, influencing the learning direction of a GNN. Therefore, various loss functions have been proposed to reveal the structure of a given graph and generate representations of nodes. Most recently, a specific extension of unsupervised learning called self-supervised learning has been proposed that can generate labels from the input data itself, enabling GNNs to be trained on pretext tasks to obtain node representations or conduct transfer learning [197]. The graph variational autoencoder (GraphVAE) [164] is a representative GNN model trained by unsupervised learning to obtain node representations. This process involves mapping a graph into a latent space distribution via an encoder and then reconstructing it through a decoder. The loss function for the GraphVAE aims to minimize both the graph reconstruction loss between the input and output graphs, and the Kullback–Leibler (KL) divergence loss between the learned latent space distribution and the standard normal distribution. After training, the GraphVAE enables the extraction of latent representations, which are useful for generating meaningful molecular graphs [198,199] or learning structural information for reconstruction purposes [200,201].

When GNNs are employed for combinatorial optimization, unsupervised GNNs are often conducted in an end-to-end fashion to generate solutions by minimizing loss functions, reflecting the characteristics of a given problem. For instance, many ideas for solving GCPs apply unsupervised learning to obtain color assignments. Ref. [202] utilizes a margin loss to minimize the difference between a threshold and node-pair distance. In Ref. [203], a loss function inspired by the Potts model in statistical mechanics is proposed, aiming to maximize the probability difference between two connected nodes. Furthermore, the researchers improve GNN performance by introducing self-information to the loss function [204]. Unsupervised GNNs are also exploited to solve other COPs, such as constraint satisfaction problems [205–207] and non-periodic two-dimensional tiling problems [208]. For example, Tönshoff et al. [206] employed a negative log-likelihood loss function based on soft variable assignments to maximize the probability of satisfying a set of given constraints. This approach enables the model to leverage knowledge from these constraints to better uncover the underlying data structure. In general, unsupervised learning methods can solve problems by exploring the data structure alone, which is advantageous for COPs in which the ground truth is difficult to obtain. However, the online learning paradigm hinders generalization and may lack efficiency compared with other methodologies.

3.2.3. RL-based training

RL has recently gained significant popularity in the machine learning community, due to its ability to handle dynamic environments. In RL, agents continually refine their policies through interactions with their environment, learning to adapt based on the outcomes of their actions. When RL is integrated with a GNN, the current state is represented by existing solutions or their segments, and the GNN models the agents. Reward functions are crucial in RL, as they guide the models to optimize the policy. They define "good" behavior within the context of a given problem, directing the learning process by specifying the desirable outcomes for which the agent should strive.

GNNs, when combined with RL, are particularly well suited for solving COPs that require sequential decision-making. Thus, they are apt for addressing path and routing problems, as presented in many works [209,210]. For instance, Structure2Vec deep *Q*-network (S2V-DQN) learns the policy of selecting each city iteratively when solving a TSP [211], thereby constructing a feasible solution sequentially. In comparison, an RL-based GNN model, ANYCSP [212] (where CSP denotes constraint satisfaction problem within this model name), has been applied to solve constraint satisfaction. Rather than constructing a complete solution through multiple steps, this model generates one solution in each action and outputs the best result among all steps.

4. Graph-based machine learning for COPs

4.1. Traditional optimization algorithms for COPs

COPs exist in many different domains, including network analysis, logistics, cryptography, operational research, and computer algorithms, serving as an essential foundation for practical applications [1,213]. Instead of dealing with infinite continuous variables, combinatorial optimization aims to find optimal solutions for a given problem by selecting, combining, and permuting objects in a finite set with certain constraints. The main challenge in solving COPs is the rapidly growing search space. The potential combinations of feasible solutions can undergo exponential growth, despite there being a small number of variables in the problem. Consequently, solving COPs requires the dedicated design of algorithms that can find promising solutions effectively.

4.1.1. Exact solvers

Traditional methods for solving COPs can be roughly categorized into two classes: exact solvers [214–216] and approximate solvers [217–219]. Exact solvers ensure that the global optimal solutions to an input problem will be found by the optimization algorithm. Branch-and-bound [220] and branch-cut-and-price [221] optimization techniques, along with their variants, are usually adopted by exact solvers as the general framework for solving COPs and can achieve a good performance in small-scale problems with an integer programming formulation. Exact solvers branch the original problem into subproblems and bound the objective values in order to explore feasible solutions in the search space. The algorithm first initializes the lower bound and the upper bound of the objective function. Then, based on a set of selected variables, multiple subproblems are generated by splitting the search space into distinct regions. Concorde[†] is a representative solver customized for routing problems. It has been adopted to calculate the optimal solutions to all 110 benchmark problems in TSP library

[†] https://www.math.uwaterloo.ca/tsp/concorde.html

(TSPLIB) [222], where the largest instance has 85 900 cities. Concorde combines both branch-and-bound and cutting-plane methods to reduce the search space and solve linear programming relaxations of the original problem iteratively. In contrast to Concorde, which is specifically designed for TSP-like routing problems, Gurobi† is a generic solver for any forms of COPs, including integrality constraints, linear constraints, bound constraints, and quadratic constraints. It has interfaces with different programming languages and platforms, such as Python and MATLAB. The Gurobi interface allows the user to put a customized optimization model into the solver and obtain the result without changing the environment.

4.1.2. Approximate solvers

However, the exponential time complexity hinders the application of exact solvers under certain circumstances. Since exact algorithms typically lead to an extremely large computing overhead, it is non-trivial to use exact solvers for combinatorial optimization as the scale of problems continues to grow, even though the algorithm is guaranteed to produce an optimal solution within a finite period of time. For example, it may take an exact solver several months or even years to find the ideal solution to certain difficult problems, whereas real-time decision makers typically demand answers within a few minutes. In contrast, approximate solvers are often significantly faster than exact solvers, since they do not require an exhaustive search across all feasible solutions, especially for large-scale problems. Approximate solvers tackle COPs by using metaheuristic and heuristic algorithms such as genetic algorithms [223], simulated annealing [224], greedy algorithms [225], and local search methods [226]. Although such methods cannot guarantee finding the global optimum, they are able to explore the search space efficiently and obtain near-optimal solutions within a reasonable amount of time, making them preferable for real-time decision-making tasks. The Lin–Kernighan heuristic (LKH) [217,218,227] is a representative approximate solver designed for solving the TSP and its variants.

Since the objective of the TSP is to determine a shortest route for a traveling salesman to visit each city once and then return to the starting city, the search space of possible solutions grows exponentially with the number of cities. Approximate solvers based on LKH solve the TSP by iteratively updating an initial solution with a local search; during each optimization step, small modifications based on the previous solution are made to improve its quality. OR-Tools‡ is an open-source project developed by Google with the primary aim of providing efficient tools and algorithms to solve the operational research problems that arise in a wide range of applications of that company. Having gone through years of development and expansion, OR-Tools integrates a variety of algorithms, including both exact and approximate solvers for solving different types of COPs in vehicle routing, scheduling, flows, and bin packing. It supports researchers and developers in modeling a problem in different programming languages and solving it with the provided commercial or open-source solvers, such as CPLEX, Gurobi, solving constraint integer programs (SCIP), and GNU linear programming kit (GLPK).

4.1.3. Machine learning methods

Traditional methods such as exact solvers and heuristic algorithms used to be the foundation of optimization methods in practical applications. Nevertheless, with the continuous development of data science and the remarkable surge in computational power, machine learning techniques have recently emerged as a compelling paradigm for efficiently addressing optimization problems for particular COPs, as illustrated in Table 1 [41,191,192,196,207,209–211,228–248]. Compared with the traditional approaches mentioned above, machine learning methods show significant advantages in several key factors that contribute to the popularity and effectiveness of machine learning in solving COPs.

Table 1

Machine learning methods for solving various COP tasks.

Model	Tasks	Training methods	Reference
Ptr-Net	TSP	SL	Vinyals et al. [228]

† https://www.gurobi.com/
‡ https://developers.google.com/optimization

Journal Pre-proofs

Ptr-Net	TSP, KP	RL	Bello et al. [229]
Ptr-Net	VRP	RL	Nazari et al. [210]
Ptr-Net	TSP	RL	Ma et al. [230]
Structure2Vec	TSP, MVC, MAXCUT	RL	Dai et al. [211]
Transformer	TSP, VRP	RL	Wu et al. [231]
Attention model	KP, TSP, CVRP	RL	Grinsztajn et al. [232]
NeuroSAT	SAT	SL	Selsam et al. [233]
NeuroSAT	SAT	CL	Duan et al. [207]
LeNSE	MAXCUT, MaxVC	RL	Ireland et al. [234]
GCN	MIS, MVC, MC, SAT	SL	Li et al. [235]
GCN	QAP	SL	Nowak et al. [236]
GCN	MaxVC, MCP	SL	Manchanda et al. [237]
Gated GCN	FLP	SL	Liu et al. [41]
Gated GCN	TSP	SL	Joshi et al. [196]
Att-GCRN	TSP	SL, RL	Fu et al. [238]
GAT	TSP	RL	Deudon et al. [239]
GAT	TSP, VRP	RL	Kool et al. [209]
GNN	TSP	SL, RL	Joshi et al. [240]
GNN	TSP	SL	Dwivedi et al. [241]

Journal Pre-proofs

GNN	TSP	SL	Hudson et al. [242]
GNN	TSP	SL	Prates et al. [192]
GNN	MVC, MC, MAXCUT	RL	Abe et al. [243]
GNN	GCP	SL	Lemos et al. [191]
ECO-DQN	MAXCUT	RL	Barrett et al. [244]
DeepACO	KP	RL	Ye et al. [245]
BQ-NCO	KP, TSP	RL	Drakulic et al. [246]
G2SAT	SAT	SL	You et al. [247]
NSNet	SAT	SL	Li et al. [248]

Ptr-Net: pointer network; SL: supervised learning; KP: knapsack problem; MAXCUT: maximum cut; MaxVC: max vertex cover; CL: contrastive learning; LeNSE: learning to navigate subgraph embeddings; QAP: quadratic assignment problem; Att-GCRN: graph convolutional residual neural network with attention mechanisms; ECO-DQN: exploratory combinatorial optimization deep *Q*-network; DeepACO: neural-enhanced framework for ant colony optimization; BQ: bisimulation quotienting; NCO: neural combinatorial optimization; NSNet: neural satisfiability network.

(1) **Learning from data.** Firstly, machine learning methods have the ability to learn from historical data, whereas traditional methods solve each problem independently. It should be noted that many COP instances share the same problem structure and differ only in their parameter values. Thus, learning from historical data enables machine learning methods to extract generic paradigms from various problem instances, leading to solutions that are more likely to be optimal for unseen instances. In contrast, traditional methods often rely on predefined rules based on the given case, which may not be universally optimal for all problems.

(2) **Handling large-scale problems.** Secondly, machine learning performs efficiently when dealing with large-scale optimization tasks. Handling large-scale problems is a challenging task that often requires efficient algorithms and scalable approaches. Exact solvers may encounter computational limitations when dealing with large-scale problems due to their exponential time complexity. On the other hand, population-based heuristics always struggle with parallelization and an enormous computational cost, since a huge amount of candidate solutions must be evaluated to find an optimal one. Machine learning methods based on deep neural networks can leverage the power of parallel computing, leading to a faster training process and less inference time. This advantage is crucial when dealing with a massive amount of data in large-scale problems.

(3) **Adaptation to dynamic environments.** Thirdly, machine learning methods can flexibly adapt to dynamic environments. Machine learning models can be continuously retrained as new data becomes available, allowing them to adapt to dynamic environments. As new data is continuously integrated into the learning process, machine learning models can not only maintain relevance but also improve their performance over time. This adaptability is particularly useful in scenarios where the optimization problem changes in a dynamic environment, since the model can remain effective and capable of providing optimal or near-optimal solutions across various states of the optimization environment. In contrast, exact solvers and heuristic algorithms with fixed rules and predefined strategies may lack adaptability in dynamic environments.

(4) **Providing the uncertainty of solutions.** Finally, machine learning models can provide different solutions, each with its own probability, whereas traditional solvers only generate deterministic solutions. In certain optimization scenarios with inherent uncertainty and variability, deterministic solutions may not be suitable and may prove to be less effective. However, machine learning models that can provide uncertainty measures along with solutions offer insights into the likelihood of different outcomes to the decision-maker. Thus, these probabilistic solutions accommodate the complexities of uncertain environments by providing a more informed and adaptive decision-making framework.

Early work on solving graph-based COPs via machine learning techniques can be traced back to the pointer network [228], in which an encoder–decoder structured model based on recurrent neural networks (RNNs) is proposed to generate feasible permutations for Euclidean TSP instances. An RNN model known as the encoder is employed to process all nodes in the input graph of a TSP instance. The encoder generates vector representations as the encoding for each node. Subsequently, the decoder, which is also implemented as an RNN model, takes the encoded node vectors as the inputs and performs an attention mechanism to generate probability distributions over the input nodes via a softmax layer. A pointer network can output a feasible permutation for the input TSP instance through an iterative decoding process. The encoder–decoder model is trained in a supervised learning manner with pre-calculated instance–solution pairs as labeled data. Similarly, Bello et al. [229] trained an RNN model via RL to make predictions on the distribution of city permutations in the TSP. The negative value of the tour length was adopted as a reward signal, and the parameters of the RNN model were optimized using the active search as a policy gradient method. In contrast to the pointer network, this work adopts RL to avoid the need for optimal solutions during the training process. However, early work on graph-based COPs is built upon the seq2seq [249] framework and uses sequential data as the input, which fails to effectively leverage the inherent graph structure characteristics of the original problems. Additionally, sensitivity to the input order usually has a negative influence on model performance, especially when the problem has multiple optimal solutions, such as the symmetric TSP.

4.2. GNNs for solving COPs

Recent advances in graph representation learning [250] are stimulating the usage of GNNs in tackling COP tasks. GNN models can analyze and process graph-structured data by leveraging node and edge features to learn the inherent relationships of the input graph, regardless of node order. Since many NP-hard COPs can be considered as sequential decision-making tasks on graphs, it is a natural choice to use graph-based machine learning models to solve complex optimization problems. GNN-based approaches to COP typically involve two stages. The first stage is to learn the graph representation of the original problem, in which node and edge features are processed through GNN models via the message-passing scheme to capture graph pattern information. The second stage is to generate feasible solutions to the problem based on the learned graph representation. Both autoregressive and non-autoregressive methods can be adopted in this process. GNNs have demonstrated remarkable ability to address a wide range of COP tasks, which can be generally categorized into three main classes: node classification tasks, edge prediction tasks, and graph-level prediction tasks.

(1) **Node classification tasks.** GNNs can effectively tackle node classification COPs such as graph coloring, in which adjacent nodes must have different colors. To do so, a GNN learns node embeddings that capture local and global graph structures through message passing between nodes. This allows it to understand each node's context and output color probabilities. The model is trained to balance the local constraints of neighbor colors with the global goals of minimizing the total number of colors, often outperforming traditional methods in large and complex graphs.

(2) **Edge prediction tasks.** GNNs also perform well at edge prediction COPs such as TSPs. Each city is considered as a node in a graph, and the connections between two cities are represented as edges. Through iterative message passing between different nodes, a GNN can capture complex relationships between cities. The model then uses these learned representations to predict the likelihood of edges being part of the optimal tour. The model is trained to minimize the total distance of traversing all nodes in the graph. Compared with specialized TSP solvers, GNN-based methods perform better in terms of flexibility and scalability, making them well-suited for real-world TSP instances.

(3) **Graph-level prediction tasks.** GNNs can also be adopted to solve graph-level COPs, such as the graph isomorphism problem. In order to determine whether two graphs are structurally equivalent, a GNN processes the entire graph structures to make predictions about their properties or relationships. By performing multiple layers of node feature aggregation and graph-wide pooling operations, the model captures both local structural details and global topological characteristics. The output is typically a graph-level embedding, which can be used for downstream tasks such as graph classification or similarity measurement.

Journal Pre-proofs

4.2.1. Non-autoregressive approaches

The use of non-autoregressive methods for solving graph-based COPs has attracted increasing attention, given their ability to generate solutions simultaneously without depending on sequential processing. In contrast to autoregressive methods that generate each solution step-by-step, non-autoregressive methods show significant advantages in terms of inference speed and search efficiency. Li et al. [235] combined GCNs with classic heuristics to predict solutions for several NP-hard problems, including maximal independent set, MVC, maximal clique, and satisfiability problems. A GCN model is trained to make predictions on the likelihood of each vertex belonging to the optimal solution to the input graph. To avoid generating a diffuse likelihood map when there are multiple optimal solutions to an input graph, a hindsight loss [251] is adopted in the training process to fit a diverse collection of solutions. Nonetheless, this approach suffers from the limitation of poor generalization, as local searching accounts for a large part of the model performance. Nowak et al. [236] investigated how a GNN model can make predictions on unseen quadratic assignment problems by learning from a set of solved cases. Liu et al. [41] proposed a learning-based method based on residual gated GCNs [252] to predict Pareto optimal solutions for multi-objective facility location problems (MO-FLPs) in an end-to-end manner. The MO-FLP is transformed into a bipartite graph representation with two distinct sets of nodes denoting the candidate facilities and customers, respectively. Two GNN models are trained cooperatively with labeled data to learn the latent space embeddings for the nodes and edges of the input graph. Finally, a set of non-dominated solutions are sampled from the output probability distribution to approximate the ground-truth Pareto set of the problem.

Another branch of work focuses on solving routing problems such as the TSP and its variants in a non-autoregressive fashion. Joshi et al. [196] trained a residual gated GCN that takes a TSP graph as the input and outputs a heat map of probabilities at which each edge belongs to the shortest path (the optional solution). The algorithm introduces additional edge features as part of the input graph information and adds residual connections for each graph convolutional layer to alleviate the over-smoothing issue occurred in the vanilla GCN. Finally, the edge representations of the last message-passing layer are passed through a multi-layer classifier to output the probability of each edge occurring in the final solution. During the prediction, beam search is adopted to convert the adjacency matrix into a set of feasible TSP tours in parallel. Despite its effectiveness, the model in Ref. [196] can only be trained and evaluated on TSP instances with a fixed number of cities. To generalize a trained model to larger cases than it had previously seen, Fu et al. [238] expanded the work in Ref. [196] and proposed the reuse of a model pre-trained on small-scale cases to predict heat maps on arbitrary large cases. During the preliminary stage, a large number of TSP instances with a fixed number of C cities are randomly generated, together with optimal solutions computed by the exact solver Concorde. These case–solution pairs serve as a labeled dataset to train a graph convolutional residual neural network with attention mechanisms (Att-GCRN) via supervised learning. Once trained, the Att-GCRN model is able to predict the heat map for any given TSP instance with Ccities. During the prediction stage, an arbitrary large instance is first divided into a set of sub-graphs, each of which has exactly C cities. The pre-trained Att-GCRN model generates heat maps for all sub-graphs. Then, all the heat maps are merged into a complete heat map via graph fusion. Finally, a Monte Carlo tree search (MCTS) is adopted to search for optimal solutions based on the merged heat map. In contrast to other MCTS-based approaches, in which the solution is built step-by-step, the model in Ref. [238] generates the solution non-autoregressively, where each state in the MCTS is a complete tour of the TSP.

4.2.2. Autoregressive approaches

Many highly-structured combinatorial optimization tasks can be considered as sequential decision-making processes on graphs [196], where each node in a graph represents distinct decision points related to the original task, and the edges encapsulate the relationships between these decisions. The sequential nature of the optimization process becomes evident in this scenario. The selections on each node are no longer independent of each other, as they are in non-autoregressive methods. In contrast, the decision on one node will directly impact the feasible choices of all subsequent nodes, implying the interdependence and constraints inherent in the original problem. An intuitive example of this kind of sequential decision constraint can be found in the classic TSP. In a single-objective TSP task, where the target is to minimize the total traveling distance by visiting each city once, the decision of the salesman at each step regarding which city to visit next will definitely affect all subsequent decisions as well as the final result. In fact, the choice of the departure city does not matter in the TSP, since the salesman will return to the starting city after traversing all cities. What really matter are the decisions made at each step within the construction process of the entire route. When deciding on which city to visit next, an intuitive method is to simply select the neighboring city nearest to the current location among all unvisited cities at each step. This greedy selection strategy can efficiently construct a feasible solution, and each choice appears to be optimal for the current state. Taking the global optimization target into consideration, however, this greedy strategy may not be preferable for the TSP, as locally optimal decisions are made based on the current state at each step, without considering subsequent implications. As a result,

for TSP instances with specific city distributions, while the partial solution constructed in the early stage may have a short traveling distance, it may also lead to a situation in which the traveling cost between each pair of unvisited cities remains very large, resulting in an overall traveling distance that is far from the global optimal solution. A similar situation occurs in non-routing problems such as the GCP. Since the aim of the GCP is to assign colors to the nodes of a graph in such a way that no two adjacent nodes share the same color, the decision made at each node regarding the assignment of colors will significantly influence all subsequent decisions, creating a ripple effect throughout the entire graph. Therefore, it is crucial to consider the common scenario of sequential decision restrictions when designing autoregressive methods for COP tasks.

This sequential decision-making process fits seamlessly with machine learning models that have advantages in capturing temporal dependencies and complicated relationships within sequential data. A classic work that combines an autoregressive method with graph representation learning to solve COPs is presented by Kool et al. [209]. Their proposed framework is specifically designed for complex routing challenges such as VRPs. Their method employs GATs [138] within an encoder–decoder framework, trained using REINFORCE with a greedy rollout baseline. In this formulation, a VRP instance is represented as a graph with $N_{\mathrm{VRP}} + 1$ nodes, where N_{VRP} is the number of customers, and an additional node represents the depot. The model learns to construct feasible solutions by sequentially selecting nodes to visit, considering vehicle capacity constraints. The attention-based encoder first generates embeddings for all nodes; then, the decoder constructs a route autoregressively, selecting one node at a time. For each step, the decoder model takes not only the node embeddings produced by the encoder but also a problem-specific mask and the corresponding context of the current state. The mask refers to the set of nodes that have been visited, and the context consists of information about the first and last node existing in the partial route. A key feature of this method is its adaptability. Using a single set of hyperparameters, the model can be applied to various VRP variants, such as VRPs with time windows or multiple depots. Dai et al. [211] encoded TSP instances as graph-structured data by using GNNs, which can capture the graph topology information of the original problem. The usage of GNN models can also better reflect the order-independent properties of TSP routes compared with previous sequence-to-sequence models [228,229]. Motivated by the situation in which most optimization instances share the same problem structure and only differ in data, Ref. [211] proposes a machine learning framework that can automatically learn heuristic algorithms for graph-based combinatorial optimization tasks by combining RL and graph embedding techniques. A graph embedding model is trained to determine actions at each step according to the current state of the input graph, and the learned policy serves as a meta-algorithm to incrementally construct solutions to the input problem. The proposed framework can generate feasible solutions in an autoregressive manner based on the graph structures by adding nodes to the partial solution step by step. The Structure2Vec model [253] is adopted as the graph embedding network, and the model is trained to capture the neighborhood information of each node in the context of graphs. A greedy policy is parameterized by the Structure2Vec model and trained via *Q*-learning strategies. Deudon et al. [239] extended the neural combinatorial optimization framework in Ref. [229] and efficiently combined it with traditional heuristics to solve COPs. Instead of using the long short-term memory (LSTM) architecture, the scholars used attention mechanisms to design the critic for the model. A framework proposed in Ref. [239] follows an encoder–decoder structure, in which the encoder maps the inputs to a set of continuous representations, and the decoder generates an output sequence in an autoregressive manner. At each step, the previously generated outputs are taken as additional inputs to the model, allowing it to consider past information when making the decision on the next city to visit. Similar to the work in Ref. [254], the action vectors are encoded as a set based on neural attention mechanisms, instead of as a sequence as they are in Ref. [229].

4.2.3. GNNs combined with other technologies

Many other machine learning technologies are always combined with GNNs to further improve the model performance on various graph-based tasks.

(1) **GNNs with unsupervised learning.** In general, the reliance on labeled data in supervised-learning-based methods may hinder their ability to generalize to large-scale problems [240,255]. Training a deep neural network model with good performance often requires a large number of optimal solutions, which can indeed be calculated by well-established solvers for conventional optimization problems. However, for many real-world optimization tasks with no exact mathematical formulations, it is non-trivial to collect a large number of optimal solutions for training a GNN model. To solve this problem, Wang et al. [256] proposed an unsupervised learning framework based on the relaxation-plus-rounding strategy. The relaxed solutions are parameterized by a GNN model, which can be trained in an end-to-end manner via a back-propagation process. Since the direct evaluation of many practical combinatorial problems can be expensive and time-consuming, an alternative approach is to learn a proxy of the original objective function first, and then optimize the proxy instead of the original objective in order to reduce the evaluation overhead.

(2) **GNNs with RL.** Since training a GNN model with a sufficient amount of labeled data can be non-trivial in scenarios where it is difficult to optimize the objective functions of COPs, another solution is to train the model through trial and error without the assistance of any labeled data. In fact, RL is about training an agent to make a sequence of decisions within an environment in order to maximize the cumulative reward it receives over time. The agent learns from its experiences by updating a policy or value function based on the received rewards. Due to inherent problem structures, many COPs can be considered as graph-based sequential decision problems. Therefore, it is a natural choice to train GNN models with RL techniques while solving COPs. For example, Dai et al. [211] trained a graph embedding model with the deep Q-network (DQN) method in RL to determine the order of nodes while constructing a sequential solution. Deudon et al. [239] used a policy gradient to train an attention-based model in order to predict the distribution over city permutations for TSP tasks. Kool et al. [209] trained an encoder–decoder model using the REINFORCE algorithm to build TSP solutions incrementally. Their model's performance on different COP tasks achieved state-of-the-art results with RL technologies.

(3) **GNNs with contrastive learning.** Contrastive learning has emerged as a state-of-the-art paradigm in graph representation learning, offering advantages in data efficiency, generalization ability, and versatility [257,258]. This approach enables machine learning models to learn principal features from data without the distraction of irrelevant information, helping the model better understand the differences between samples to learn more meaningful representations. The contrastive learning loss function is designed to minimize the distance between similar data pairs while maximizing the distance between different data pairs. This property of not relying on labeled data has proven effective in various domains, including graph representation learning and various applications of COPs. The advantages of graph contrastive learning in solving various graph-based COP challenges lie in its unique ability to learn robust and discriminative graph representations without relying on explicit labels. This approach leverages the inherent structure of COPs by contrasting different views or augmentations of the same graph, enabling the model to capture essential problem-specific features and relationships. Unlike traditional supervised methods, graph contrastive learning can generalize better to unseen instances and various scales by focusing on underlying patterns rather than specific solutions. Therefore, it addresses the challenge of limited labeled data and provides an efficient framework for tackling diverse COP tasks. Duan et al. [207] introduced contrastive learning into combinatorial problems and investigated the effect of an augmentation design involving pre-training for SAT problems. The researchers argued that label-preserving augmentations can lead to better performance on combinatorial problems than label-agnostic augmentations [259,260]. Jing et al. [261] proposed GCARec, a novel graph contrastive learning framework for recommendation systems that addresses the challenges of sparse supervision signals and noisy interactions in GCN-based methods. The adaptive data augmentation scheme calculates edge retention probabilities using attention mechanisms, employing a generalizable approach that can outperform baseline methods on real-world datasets. Similarly, Jiang et al. [262] introduced AdaGCL, an adaptive graph contrastive learning framework that addresses the challenges of noisy and skewed user behavior data in GNN-based recommender systems. The framework employs two trainable view generators—a graph generative model and a graph denoising model—to create adaptive contrastive views, introducing additional high-quality training signals that help alleviate data sparsity and noise issues in collaborative filtering.

4.2.4. Challenges in GNN-based optimization

Despite their effectiveness, GNN-based methods present several challenges when applied to various COPs in real-world scenarios. Firstly, the scalability of such methods remains a significant issue, particularly for large-scale graphs. The computational complexity of message-passing operators can increase significantly with graph size, making these methods less efficient in training and inference processes. Secondly, dynamic graphs introduce another challenge for GNN-based methods. In certain scenarios where the graph structure changes over time, the model should have the ability to efficiently update representations as the graph topology changes, which could be challenging for fixed GNN structures. Moreover, most existing methods incorporate task-specific knowledge into the design of GNN models, which limits their transferability to other graph-based COP tasks. Addressing these challenges is essential for the further application of GNN-based methods in solving real-world COP tasks.

5. A unified framework for COPs with GNNs

In this section, we introduce a unified framework that employs GNNs to analyze and represent the structural variations of graphs across different COPs. We begin by elaborating the essential components in the proposed framework, providing a systematic approach for integrating GNNs with COPs. We then expand the discussion to explore how this unified graph framework can be effectively applied to address multi-objective, constrained, and dynamic COPs. By leveraging the capabilities of GNNs, the framework aims to enhance the understanding and processing of COPs, thereby offering robust solutions across varied domains.

5.1. A unified graph framework for solving COPs

The unified graph framework proposed in this study integrates multiple key components to deliver a systematic and flexible approach for modeling, analyzing, and solving a wide range of COPs. As illustrated in Fig. 3, the primary elements of the unified graph framework include the graph representation of COPs, equivalent conversion of non-graph-structured COPs, graph decomposition, and graph simplification.

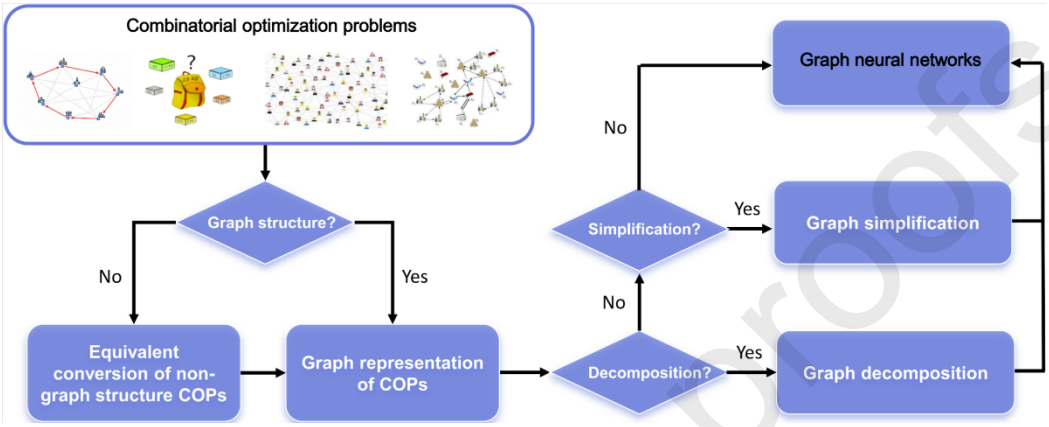


Fig. 3. A unified graph framework for COPs.

COPs that inherently have a graph structure are represented as graphs in which nodes correspond to elements, and edges depict relationships or constraints. This representation involves transforming a COP into a graph structure and designing nodes and edges that accurately reflect the problem's features and constraints. A GNN then leverages these graph representations to propagate information among nodes and capture complex dependencies. Section 5.2 will discuss graph representation methods for COPs with graph structure information using GNNs.

Conversely, COPs that exist as sequences or sets without an inherent graph structure must be converted into graph-like structures. This equivalent conversion process involves defining nodes and edges based on the characteristics of the COPs, thus enabling the application of GNN techniques. Section 5.3 will discuss various types of non-graph-structured problems and explain their transformation into suitable graph representations. Furthermore, given the inherent complexity of some problem structures, graph decomposition and simplification become critical techniques. COPs often involve large, complex graphs that pose computational challenges. Graph decomposition techniques break down the primary graph into smaller, manageable subgraphs, facilitating localized optimizations and more efficient exploration of solution spaces. Section 5.4 will cover various decomposition methods, including graph partitioning and hierarchical decomposition, and discuss their applicability and effectiveness in different COP problems. Graph simplification, discussed in Section 5.5, reduces the complexity of graphs by eliminating redundant elements and pruning unnecessary branches, thus making problems more tractable for optimization algorithms. Simplifying and decomposing the problem graph increases the computational efficiency, improves scalability, and enables the implementation of tailored optimization strategies. The flexibility of this unified graph framework accommodates various COP structures, including graphs, sequences, and sets, making it broadly applicable across different application domains. To make our proposed unified graph framework more concrete, we present the location-routing problem (LRP) as an illustrative example. The LRP is a complex COP that simultaneously determines the optimal locations for facilities (e.g., satellites or distribution centers) and the routing of vehicles to serve a set of customers [25]. The objective is to minimize the total cost, including facility opening costs and transportation expenses, while satisfying customer demands and adhering to constraints such as vehicle capacity and delivery time windows.

In our unified graph framework, we model potential facility sites and customer locations as nodes in a graph. Edges represent possible routes between facilities and customers or between customers themselves, with weights reflecting factors such as the transportation cost, distance, or travel time. At the same time, non-graph-structured constraints, such as vehicle capacities and service time windows, are integrated into the graph based on equivalent conversion by assigning relevant

attributes to nodes and edges. Moreover, to handle the complexity of large-scale LRPs, we apply graph decomposition techniques by partitioning the graph into smaller subgraphs based on geographic proximity or demand similarity, thus enabling parallel processing and more manageable computations. Graph simplification methods are also employed to eliminate redundant nodes and edges, reducing computational overhead. By leveraging GNNs within this unified graph representation, we effectively capture the intricate relationships among facilities, vehicles, and customers, facilitating efficient and scalable solutions to the LRP. This example demonstrates how our framework transforms complex COPs into unified graph-based representations, making them more tractable for analysis and solution using advanced GNN-based algorithms. Leveraging GNN-based algorithms within this framework facilitates efficient problem analysis and solution strategies. Additionally, the integration of graph theory with combinatorial optimization encourages interdisciplinary collaboration, while the adoption of advanced machine learning techniques enhances adaptability and performance in optimization. Overall, the unified graph framework provides a systematic and versatile approach to tackling complex optimization challenges, fostering innovation and efficiency across disciplines.

5.2. Graph representation of COPs

5.2.1. Basic representation of GNNs in COPs

COPs typically involve interactions and constraints among multiple decision variables, where graph structures provide an intuitive and effective means to depict these relationships. In such graphs, nodes represent the variables or components of the problem, while edges indicate the relationships or constraints between these variables.

GNNs are good at leveraging the inherent structures within graphs to increase the resolution of COPs. By aggregating features from neighboring nodes, GNNs can dynamically update node states, which is fundamental in solving problems such as the shortest path, where the goal is to find the most efficient route between nodes [263]. Similarly, in constructing minimum spanning trees, GNNs simulate edge weights to create optimal connectivity with minimal costs [264,265]. Additionally, in scenarios where understanding node importance is crucial, such as in network analysis, GNNs provide estimations of node centrality, which helps identify key nodes based on their structural roles in the graph [266].

Moreover, GNNs have advanced data representation capabilities that facilitate efficient problem-solving in areas requiring high accuracy in node and edge representations, such as coloring and matching problems. Using machine learning architectures such as Sinkhorn, communicative message passing neural network (CMPNN), and seeded graph matching network (SGMNet), GNNs can address image coloring tasks and semantic matching challenges by learning and applying precise node embeddings, which significantly improve the quality and efficiency of the solutions [191,202,203,267].

5.2.2. Examples of COP graph representation

By modeling COPs as graph structures of nodes and edges, GNNs can leverage their profound capabilities in various applications such as pattern recognition, network dynamics, and predictive analytics. We provide some examples of how GNNs integrate complex relational data with advanced algorithms to optimize network operations, resource management, decision-making processes, and so on. These examples showcase the versatility and power of GNNs in extracting meaningful solutions from graph structures and thus contributing significantly to advances in various fields.

(1) **Complex network optimization.** In the realm of complex network optimization, GNNs excel by integrating relational information among nodes with sophisticated algorithms. For the TSP, an NP-hard problem that seeks the shortest possible route for visiting a set of locations and returning to the origin, GNNs combine node relational data with heuristic algorithms or RL to find optimal solutions [192,211]. Similarly, in network flow problems such as maximum flow and minimum cut, GNNs deploy flow classification and node deployment techniques to manage and optimize the flow of resources through a network, ensuring optimal operation and resource utilization [268–270].

(2) **Resource management and scheduling.** Resource management and scheduling are critical in many industrial and technological environments, where efficient resource allocation and timing are key in operational success. In resource-allocation scenarios, GNNs model each resource as a node and optimize allocation by analyzing dependencies and exclusions represented as edges [271,272]. For job shop scheduling, in which tasks must be efficiently assigned to production units, GNNs use Markov decision processes to determine optimal task arrangements and scheduling, thereby increasing productivity and reducing operational delays [273,274].

(3) **Social network analysis and decision-making.** In social network analysis, GNNs are particularly effective in solving problems such as the MVC and the MDS by transforming these challenges into node classification tasks. This method involves labeling each node based on whether it belongs to a particular set and using a GNN to predict these classifications based on the nodes' features and relationships. This approach allows for efficient and accurate decision-making, leveraging the structural information within the network to solve these graph-based decision problems [275].

5.2.3. Benefits of graph representation

GNNs effectively utilize the intrinsic structure of graphs (i.e., nodes and edges) to capture the topological and relational features necessary for solving various types of COPs. By designing appropriate graph representations that reflect the inherent features and constraints of each problem, GNNs can deliver robust and efficient optimization solutions across a variety of domains, from routing and scheduling to resource allocation and beyond. Moreover, GNNs can optimize performance and enhance decision-making processes by tailoring graph representations to reflect the specific features and constraints of each problem.

The use of graph representation in conjunction with GNNs marks a significant advance in tackling COPs. Structuring problems as graphs makes it possible to leverage the inherent strengths of GNNs to effectively capture and analyze complex relational data. This graph representation of COPs allows GNNs to adaptively respond to changes in problem data and structure, offering dynamic solutions that traditional methods cannot easily provide. As a result, the application of graph representation not only increases the accuracy and efficiency of solving COPs but also expands the possibilities for innovation in optimization strategies across diverse industries. Capitalizing on the capabilities of GNNs through graph representations unlocks new potential in computational optimization, setting the stage for solving some of the most challenging problems across various applications.

5.3. Equivalent conversion of non-graph-structured COPs

5.3.1. General process for equivalent conversion

In many areas of combinatorial optimization, problems inherently lack graph structures, which means that traditional relationships and interactions among elements are not depicted as nodes and edges [276,277]. To leverage the capabilities of GNNs for these non-graph-structured COPs, the problems must first be converted into equivalent graph structures. This transformation is essential for effective feature learning, problem modeling, and solution derivation by means of GNNs.

The conversion process involves a detailed mapping in which each entity, resource, or constraint within the problem is represented as a node, and all relationships—including dependencies, exclusions, and priorities—are represented as edges. This reconfiguration translates the abstract elements of the problem into a graph format, laying a solid foundation for further analysis by means of a GNN. Through this transformation, a GNN can effectively capture the complexities and correlations inherent in the problem, offering novel insights and resolution methods.

5.3.2. Examples of equivalent conversion

The process of transforming non-graph-structured COPs into graph formats significantly expands the application of GNNs. This equivalent conversion makes it possible to effectively exploit GNNs' graph-based analytical capabilities, leading to more robust and scalable solutions. The following examples demonstrate how different types of COPs can be converted into graph structures, each benefiting from the unique insights and optimizations that GNNs provide.

(1) **The bilevel knapsack problem (BLKP).** The BLKP can be structured as a tripartite graph containing leader nodes, follower nodes, and constraint nodes, which represent different categories of items with attributes such as profit and weight (Fig. 4). The leader makes its decisions first, while the follower responds by making the best possible choices. The leader has insight into the follower's information, and the objective of the BLKP is to find an optimal solution by considering the follower's reactions from the leader's perspective. This detailed graph-based configuration facilitates the management of hierarchical relationships and simultaneous constraints, making it easier to optimize decisions that simultaneously affect multiple layers of the problem [278]. The graph structure helps in visualizing and tackling the dependencies and interactions between different groups, increasing the efficiency of resource allocation and optimization [279].

Journal Pre-proofs

Journal Pre-proofs

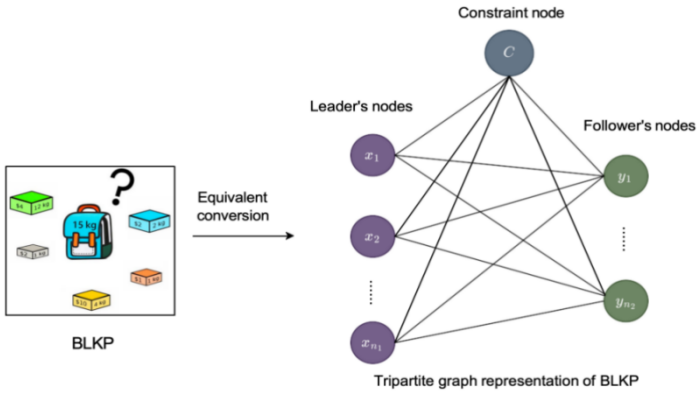


Fig. 4. Tripartite graph representation of the BLKP with an equivalent conversion. The n_1 leader's nodes, the n_2 follower's nodes, and one constraint node form a tripartite structure.

(2) **The CSP.** In the CSP, transforming the problem into a graph structure by modeling the variables and constraints as nodes and edges significantly improves the approach to solving it. Variables are represented as column nodes and constraints are represented as constraint nodes within a bipartite graph, where the edges indicate the impact of variables on constraints [280]. This setup allows a GNN to extract and learn the features from this graph representation, focusing on the interactions between variables and constraints. By learning the optimal relationships and flow within the graph, the GNN can predict and evaluate potential solutions effectively, thereby optimizing the column generation method that is crucial for solving the CSP. The integration of GNNs with RL further refines the solution process: By treating the column selection phase as a sequential decision-making problem, RL utilizes the features learned by the GNN to guide the decision-making process, thereby increasing the efficiency and speed of finding optimal cutting patterns [281].

(3) **The SCP.** In converting the SCP to a graph, universe nodes represent all possible elements that need coverage, element nodes correspond to specific items requiring coverage, and subset nodes represent the groups or sets that can provide this coverage. This transformation into a graph simplifies the visualization and management of the complex relationships between sets and their elements, facilitating more effective coverage strategies and cost management. It enables a GNN to assess and optimize the coverage dynamically, ensuring that the most cost-effective and comprehensive solutions are identified [282,283].

(4) **The 2-SAT.** The 2-SAT can be transformed into a directed acyclic graph that represents the logic circuit of the problem, with nodes designated for inputs and various logic gates. This graph structure allows GNNs to apply sophisticated algorithms tailored to logic processing, effectively solving the 2-SAT by analyzing the circuit's logical flow and gate interactions. The graph-based approach not only simplifies the representation of complex logical relationships but also improves the ability of GNNs to perform logic simplification and optimization tasks [284–286].

Converting non-graph-structured COPs into graph structures is a crucial step in applying GNNs to solve them. These examples underscore the adaptability and effectiveness of GNNs in dealing with various complex COPs transformed into graph formats. By leveraging graph representations, GNNs can be used to unlock new potentials in computational optimization, offering solutions that are both innovative and directly aligned with the inherent structure of each problem. This capability highlights the critical role of graph conversion in broadening the applicability of GNNs across diverse industries and complex optimization challenges. However, this equivalent conversion in our unified graph framework is not straightforward and involves several challenges. Firstly, accurately representing problem elements is a grand challenge. Non-graph-structured COPs often lack explicit nodes and edges, requiring careful analysis of the problem's components to determine which elements should be represented as nodes and which relationships should be represented as edges. Secondly, capturing complex constraints and relationships presents significant difficulties. COPs often involve complex constraints such as time windows and capacity limits. To address these challenges, our framework may also need to combine several advanced strategies, such as graph construction techniques and custom GNN architectures, to generate high-quality solutions even for complex real-world problems.

5.3.3. Benefits of the equivalent conversion of COPs

Converting non-graph-structured problems into graph formats significantly expands the capabilities of GNNs, extending their applicability and increasing their problem-solving effectiveness. This equivalent conversion makes it possible to utilize GNNs' advanced graph-based learning algorithms, which excel in capturing and analyzing intricate relationships and structures within data. Consequently, GNNs can perform detailed feature extraction, probe into the deeper dynamics of the problem, and generate insights that inform more effective and often more innovative solutions.

These GNN-based approaches are particularly advantageous in complex optimization scenarios, where traditional algorithms may struggle to process and interpret the underlying relationships and constraints effectively. By visualizing and manipulating problems as graphs, GNNs can better analyze the impact of various elements and their interconnections, leading to solutions that are not only optimal but also more comprehensible in terms of strategic decision-making. Additionally, this conversion increases computational efficiency, enabling faster processing and analysis, which is crucial in scenarios involving large datasets or requiring real-time decision-making.

5.4. Graph decomposition

Graph decomposition is a pivotal technique in solving COPs that involves breaking down large, complex problems into smaller, more manageable subproblems structured as graphs [287]. For example, the LRP can be decomposed into a compatible BMP and multiple VRPs [25]. As illustrated in Fig. 5, these VRPs can be classified into two types: One involves the VRP from the depot to satellites, while the other pertains to the VRP from satellites to customers. This strategy leverages the capabilities of GNNs to analyze complex relationships and patterns within these subgraphs, facilitating more efficient problem-solving. By reducing the complexity of the problem and increasing the solution efficiency, graph decomposition is essential when GNNs are deployed to tackle intricate COPs. It also helps overcome computational constraints, making the solution of each subproblem more feasible and efficient [114].

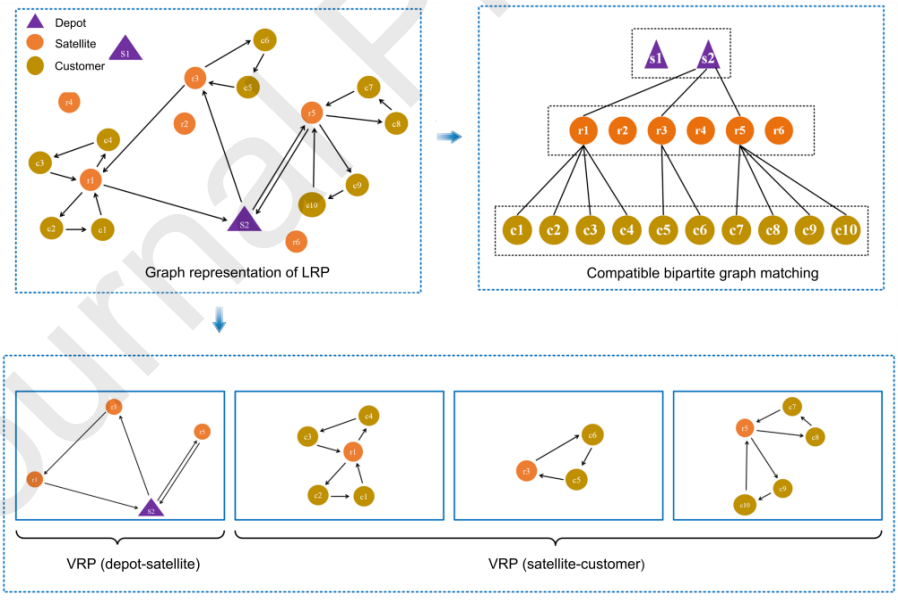


Fig. 5. An example of graph decomposition for the LRP, which can be decomposed into a compatible BMP and multiple VRP problems. s_1 and s_2 denote the depots, $r_1, \ldots, r_6$ denote the satellites, and $c_1, \ldots, c_{10}$ denote the customers.

(1) **Graph partitioning.** Graph partitioning is a method used to divide a graph into smaller, interconnected segments or partitions with the aim of minimizing the number of edges between these segments. This technique is particularly useful for problems in which the computational load can be distributed across different segments, thereby increasing the efficiency and reducing the processing time. For instance, in a large-scale TSP instance, a GNN using unsupervised learning techniques will

decompose the problem into smaller TSP problems. Heatmaps from pre-trained models are applied to these subproblems, which are derived from partitioning strategies. An attention-based mechanism then integrates these heatmaps into a unified heatmap, constructing a candidate set of edges for solving the original problem [288]. Additionally, in wireless Internet of Things (IoT) networks, GNN-based supervised learning algorithms decompose complex RAPs into simpler subproblems, such as link scheduling and joint channel and power allocation in device-to-device (D2D) networks. This decomposition permits the targeted and efficient resolution of each subproblem, simplifying the overall complexity and increasing resource distribution effectiveness [289].

(2) **Hierarchical decomposition.** Hierarchical decomposition involves breaking down a graph into layers or hierarchies, which allows each layer to be tackled sequentially or in a divide-and-conquer manner. This approach is beneficial for complex systems in which different layers can represent varying levels of detail or different subsystems within the larger problem context. For example, in a scenario involving resource allocation, a GNN combined with multi-agent RL techniques will decompose the network into collaborative decision-making problems among multiple agents. Each agent makes decisions based on the features of its respective node, with the GNN learning low-dimensional representations for each node to effectively address the problem [272]. Moreover, in addressing a CSP, a GNN will initially predict a subgraph containing potential solutions, which acts as a filter. This subgraph is subsequently processed using traditional combinatorial optimization solvers to finalize the solution. This two-step approach simplifies the initial problem into a more focused challenge, allowing for more efficient problem resolution [283].

These graph decomposition approaches not only reduce the complexity of large COPs but also increase the overall solution efficiency. By applying graph partitioning and hierarchical decomposition, GNNs can address intricate problems more effectively, supporting parallel processing and improving scalability [114].

5.5. Graph simplification

Graph simplification is a critical technique in the application of GNNs to tackle COPs. The intricate graph structures that result from modeling these problems often pose significant challenges in terms of encoding and computational feasibility [290]. In the complex landscape of combinatorial optimization, the structural complexity of problems can often impede both understanding and computational efficiency.

To address these challenges, GNNs utilize a range of graph simplification techniques that transform these intricate problem structures into more manageable forms, effectively reducing the problem scale. These methods not only facilitate deeper analysis and learning by GNNs but also increase the efficiency of the problem-solving process. Below, we explore several key techniques employed in graph simplification, each of which is tailored to optimize specific types of combinatorial problems by reducing their complexity and making them more amenable to effective resolution with advanced GNN algorithms.

(1) **Graph sampling and transformation.** Traditional mathematical graph theory methods, such as graph sampling and transformation, are utilized to streamline complex graph structures, making them more manageable for GNN processing. For example, in addressing large-scale sparse TSPs, these techniques help break down the problem into smaller, more manageable segments, making it easier for GNNs to effectively handle and learn from the underlying graph structure [288].

(2) **Machine-learning-based methods.** In solving the MVC problem, a combination of dynamic programming-like recursive structures and machine-learning-based binary classifiers are employed to simplify the graph during recursive and iterative processes. This method not only improves the manageability of the problem but also significantly reduces the computational overhead [270]. In addition, as illustrated in Fig. 6, for the SCP, a graph simplification based on hypergraph representation can be employed to predict the importance of each subset, which can achieve runtime speedups of up to 10 times on average when compared with state-of-the-art commercial solvers [282].

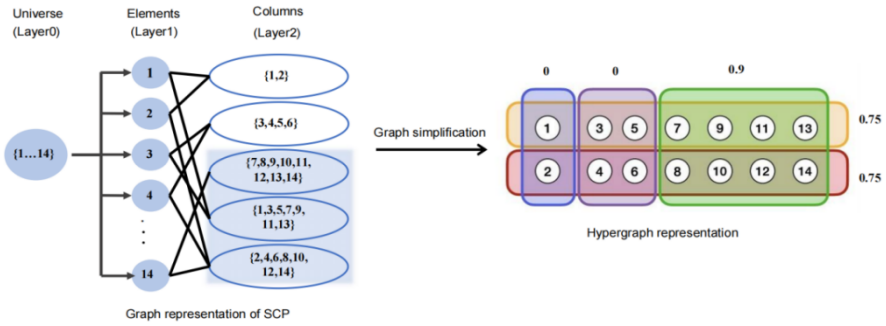


Fig. 6. An example of graph simplification using a hypergraph representation for the SCP. The hypergraph representation for the same set cover instance shows each subset corresponding to a hyperedge, illustrated as bounding boxes around the element nodes.

(3) **Reduction rules.** Specific rules, including the removal of isolated vertices, the merging of duplicate vertices, and the simplification of edges, are applied to reduce the complexity of mix-cut problems. These strategies help decrease the problem's scale, making it more tractable for GNN analysis and significantly reducing both computational time and resource consumption [291].

(4) **Adaptive motif and low-correlation pooling.** For challenges like the MDS problem, techniques such as adaptive motif and low-correlation pooling are utilized to ensure a simplified graph representation with reduced correlations. These methods aid in more effective problem-solving by focusing on essential relationships and minimizing noise [191].

(5) **Principal neighborhood aggregation (PNA).** In the double-layer knapsack problem, graph simplification through the use of GNNs and PNA transforms the problem into a single-layer optimization challenge. This approach not only increases the accuracy of the solution but also improves the overall efficiency of the process [275].

These graph simplification methods leverage various technologies and principles to reduce the complexity of graph models in COPs, thereby improving the efficiency and accuracy of problem-solving. Reducing the scale and complexity of the graphs allows GNNs to more effectively process and analyze the structures, leading to quicker and potentially more optimal solutions. These graph simplification methods utilize different technologies and principles to simplify and handle the graph structures in COPs, thereby improving the efficiency and accuracy of problem-solving.

5.6. Multi-objective combinatorial optimization problems

MOCOPs frequently emerge in real-world scenarios, often characterized by conflicting objectives [292]. These can be categorized into three distinct types based on their optimization targets: homogeneous objectives, heterogeneous objectives, and constraints as objectives. Unlike single-objective COPs, where a singular optimal solution exists, MOCOPs inherently lack a global optimum that satisfies all objectives simultaneously. Consequently, the focus shifts to identifying a Pareto set of solutions, each representing optimal trade-offs among competing objectives.

In applying the proposed unified graph framework to MOCOPs, the primary challenge is the generation of the Pareto set, which is a task more complex than finding a single solution to a typical combinatorial problem. To tackle this challenge, the unified graph framework for solving COPs can be basically adapted to include graph representation and equivalent conversion and graph decomposition [41,293]. Incorporating GNNs into this framework offers innovative approaches to deriving multiple solutions for MOCOPs.

(1) **A single GNN with diverse non-dominated outputs.** This strategy is particularly effective for objectives that share physical meanings but differ in parameters. For example, a preference-conditioned model can adaptively generate non-dominated solutions based on various user preferences [293]. As illustrated in Fig. 7 [293], the model is trained to solve bi-objective TSPs according to specific preferences and subsequently outputs the corresponding non-dominated solutions. This approach not only increases the model's utility in practical scenarios but also simplifies the complexity involved in managing multiple objective functions simultaneously. It is especially beneficial in scenarios where the trade-offs between objectives must be dynamically adjusted according to evolving user preferences or environmental conditions.

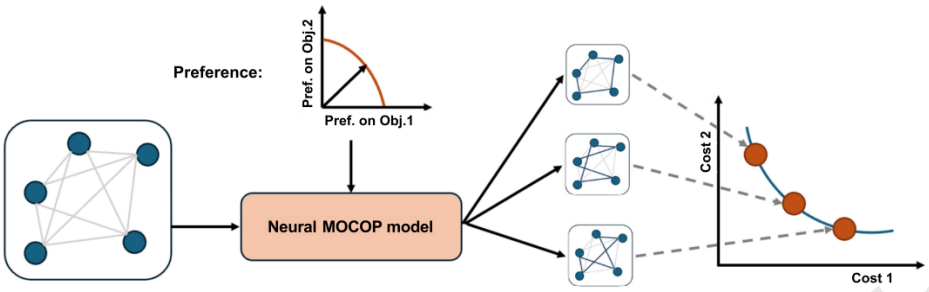


Fig. 7. An example of the use of a single neural MOCOP model to solve a bi-objective TSP (Obj.1 and Obj.2); the model outputs different solutions based on different preferences (Pref.) provided by the decision-makers [293].

(2) **Multiple GNNs for different objectives.** Suitable for objectives with distinct meanings, this strategy utilizes separate GNN models for different optimization tasks, such as node classification and edge prediction, to produce a comprehensive set of solutions by integrating the outputs from each model [41]. Employing multiple GNNs allows for specialized attention to the unique characteristics of each task, making it possible to optimize performance and accuracy, whereas a single model might struggle with conflicting objectives. In this way, each model can be optimized independently, enabling more effective parameter tuning and learning strategies tailored specifically to the needs of each task.

The use of GNNs has shown promising results in addressing MOCOPs, thanks to GNNs' ability to effectively process and analyze complex graph structures. However, the field remains relatively unexplored, with significant potential for further research. Challenges such as computational demand, data scarcity, and the need for tailored model architectures persist, making it essential to advance the development of GNN techniques that can efficiently handle diverse and competing objectives within the scope of MOCOPs.

5.7. Constrained COPs

Constrained COPs are concerned with finding an optimal solution for a given objective while satisfying a set of constraints. In constrained COPs, the target is to select a combination of variables from a limited set of candidate solutions in order to maximize or minimize the objective function, subject to a set of given limitations or requirements. Numerous COPs are derived from optimization tasks encountered in industrial manufacturing and daily life. Different constraints are commonly associated with these tasks due to the inherent limitations and requirements that must be considered when solving practical problems.

In the unified graph framework, constraints are inherently integrated into the graph structure. For graph-structured COPs, constraints are typically reflected as properties of nodes or edges. For example, in a capacitated FLP, each facility node has a capacity property representing the maximum number of services it can provide. For COPs without graph structures, constraints can be expressed as edges with embeddings on the graph structure of their equivalent conversions. Constraints for COPs can be classified into different categories according to specific problem structures and solution forms, as outlined below.

(1) **Feasibility constraints.** Many COPs have inherent limitations on the feasibility of solutions, especially for routing problems such as the TSP and VRP. In a TSP task with C cities, each feasible solution appears as a permutation of N integers. The permutation-based solutions of the TSP imply that each city can be visited only once. In other words, at each step, the traveling salesman can only select the next city from all the unvisited cities; the visited cities are unfeasible for the following part of the solution [242]. Similarly, the VRP task aims to find the optimal routes for multiple vehicles visiting a set of locations [209].

(2) **Resource constraints.** Many real-world tasks involve the allocation of limited resources, including time, cost, human resources, and physical materials [41], as the resources available to organizations or individuals are always not infinite. For example, in project management, there are constraints on the availability of human resources, equipment, and budget, which

must be considered when scheduling tasks and allocating resources efficiently. In certain special scenarios, reducing the consumption of limited resources is as important as optimizing the objective function [235,245].

(3) **Capacity constraints.** In real-world applications, there are limitations on the maximum amount of resources that can be utilized or allocated within a given system. Capacity constraints are crucial for various kinds of constrained COPs. For example, in the knapsack problem, each item has both value and weight properties, and the target is to maximize the value of the items that can be packed into a knapsack of limited capacity. The capacity constraint in the knapsack problem is the maximum weight capacity of the knapsack; the total weight of all items must not exceed this capacity [229,245]. In routing problems such as the VRP, each vehicle has a maximum carrying capacity. Each route in the solution must satisfy the vehicle's capacity limit to ensure that the total number of customers assigned to the vehicle does not exceed its capacity [210,231]. In constrained FLPs, the target is to define the optimal locations as well as the allocation strategy for a set of facilities, so that the demands of all customers can be satisfied. Thus, capacity constraints may arise from limitations on the maximum throughput or production capacity of each candidate facility.

The above examples demonstrate how capacity constraints exist in a wide range of COPs, affecting the scheduling, allocation, and utilization of resources in different contexts. Dealing with capacity constraints is crucial for generating feasible solutions to meet operational requirements and constraints in real-world scenarios.

5.8. Dynamic COPs

In most of the optimization scenarios mentioned above, the optimization task is assumed to be time-invariant, implying that the optimal solution is fixed once the parameters and constraints of the problem are determined. However, there are a wide range of dynamic scenarios in real-world systems in which the parameters, constraints, and optimal solutions of the optimization problem change over time. Dynamic COPs enable solutions to adapt to changing situations, ensuring that systems remain efficient and effective over time. For a graph-structured COP, the dynamic nature is reflected in the changing embedding values of its nodes and edges over time. For a COP without a graph structure, the dynamic attributes are apparent through the diverse structures of their equivalent conversions at different time points. Overall, dynamic COPs widely exist in practical optimization tasks due to several factors.

(1) **Dynamic environments.** Many real-world systems are dynamic in nature, which means that their parameters, limitations, and objectives might change over time. For example, in manufacturing, the demands, resources, and market conditions cannot be constant all the time. In a dynamic facility location task, it can be assumed that at the beginning of the optimization process, the distribution of clients is relatively compact at the beginning of the optimization process, with limited demand. In this case, the candidate factories closer to the customers with smaller fixed costs are preferable for a minimum total cost. However, as the business expands and the market demand changes, a growing number of clients will be distributed across various locations. As a result, the number of required factories increases, and the transportation cost has a much greater influence on the total cost compared with the fixed cost of candidate factories. Correspondingly, the optimal solution might also change based on the dynamic environment [294].

(2) **Uncertainty and robustness.** Real-world optimization problems often involve variability factors such as demand, resource availability, and other external factors. The uncertainty brings additional challenges into optimization and puts greater demand on the robustness of the solution. Dynamic COPs provide a framework for modeling and managing uncertainty by allowing for the incorporation of stochastic elements and the adaptation of solutions in response to changing conditions [295].

(3) **Real-time decision-making.** In applications such as logistics, transportation, and supply-chain management, decisions must be made in real time in order to respond to changing demands, traffic conditions, or even disruptions. For example, in a dynamic TSP case, the number of cities and the cost matrix are both time-varying. The target of the dynamic TSP is to determine the optimal route for each time step according to the changing parameters. Dynamic COPs provide a framework for real-time decision support, enabling the efficient allocation of resources and the optimization of operations as conditions evolve [296].

As the search space changes over time, it is essential to find optimal solutions dynamically, based on the changes in the optimization scenarios.

Journal Pre-proofs

5.9. Complexity analysis

To evaluate the scalability and practical applicability of our unified graph framework for COPs, we analyze the time and space complexity of each key component: graph representation and conversion, graph decomposition and simplification, and GNN application.

In the graph representation or equivalent conversion phase, transforming COPs into graph structures involves not only processing variables and constraints but also encoding problem-specific features. This includes mapping entities in the problem to nodes, defining edge connections based on relationships or constraints, and assigning feature vectors to nodes and edges to capture key attributes. Although the time complexity remains $O(N+M+E)$, in practical applications, feature extraction and encoding may involve additional computational steps, such as computing similarities or constructing adjacency matrices, which can affect the size of the constant factors [250]. In terms of space complexity, in addition to storing the features of nodes and edges, additional data structures may be required to support efficient graph traversal and operations, such as adjacency lists or adjacency matrices. For sparse graphs, an adjacency list is a more efficient choice, with a space complexity of $O(N+E)$. If an adjacency matrix is used, the space complexity increases to $O(N^2)$, which represents a significant storage overhead for graphs with large values of N. Therefore, in our proposed framework, we can employ optimized data structures based on graph sparsity to utilize space resources effectively [297].

In the graph decomposition and simplification phase, the choice of algorithms significantly affects the time and space complexity. For example, using spectral clustering for graph partitioning requires eigenvalue decomposition, resulting in a time complexity of $O(N^3)$, which is impractical for large graphs [298,299]. Therefore, we employ approximate algorithms or alternative methods, such as multilevel clustering or graph coarsening techniques, to reduce the complexity to $O(N\mathrm{lb}N)$ or even $O(N)$. Similarly, graph simplification techniques such as edge pruning based on edge weight thresholds or node aggregation based on feature similarity can be implemented using efficient algorithms to reduce the computational overhead [300].

In the application of GNNs, the time complexity per layer is $O(EF^2)$, where F is the feature dimension. However, in practice, computational costs can be reduced through optimization. For example, utilizing sparse matrix operations, mini-batch processing, and hierarchical sampling methods (e.g., GraphSAGE [301] and FastGCN [302]) can lessen the dependence on E, bringing the per-layer time complexity closer to $O(NF^2)$. Additionally, the depth L of the GNN affects both the model's expressive power and the computational load. While deeper networks can capture higher-order neighborhood information, they may also suffer from over-smoothing or gradient vanishing issues. Therefore, we strike a balance between depth and efficiency based on the specific characteristics of the problem, choosing an appropriate L. Regarding the space complexity of GNN training, storing the intermediate activation values required for back-propagation can consume a great deal of memory, especially when dealing with large graphs. To address this issue, techniques such as gradient checkpointing can be employed to recompute parts of the activation during back-propagation, thereby reducing memory usage. Furthermore, utilizing mini-batch processing and subgraph sampling methods permits the sequential processing of subsets of the graph, keeping memory requirements within manageable limits [303].

Our framework can also benefit from hardware acceleration, such as graphics processing unit (GPU) parallelism and distributed computing, which can further increase computational efficiency. For example, operations on different nodes or edges can be executed in parallel, and data parallelism can be achieved by distributing subgraphs across multiple processing units [304]. Potential extension methods for the framework can also be explored on ultra-large-scale graphs. For example, utilizing graph databases or distributed graph computing platforms can further expand the processing capability of the framework. In summary, through an in-depth analysis of time and space complexity, we have further demonstrated the advantages of the proposed unified graph framework in terms of scalability and computational efficiency. By employing optimization techniques and advanced algorithm designs in each component, we ensure the practicality and efficiency of the framework when solving large-scale COPs in the real world. This not only reflects the theoretical value of the framework but also showcases its great potential in practice.

6. Open challenges

Although the proposed unified framework offers a novel and effective method for solving COPs, it still presents several grand challenges. These include scalability, where the framework must efficiently handle increasingly large and complex datasets; graph transferability, which refers to the ability to apply learned models across different but related graph structures;

complexity, which refers to the intricate computations required by advanced GNN models; and trustworthiness, which mainly involves the reliability and privacy-preserving capability of the framework.

6.1. Scalability

Scalability is one of the key criteria for measuring the performance of combinatorial optimization algorithms in real-world applications. Most NP-hard COPs involve a large number of variables and constraints. As the problem size grows, the required computational resources increase exponentially. Traditional mathematical solvers can effectively solve small-scale instances within a limited time cost; however, the computational overhead for large-scale cases is often unaffordable. Evolutionary algorithms search for optimal solutions by iteratively evaluating a large number of candidates. As the scale of the problem grows, the solution space expands rapidly, making it difficult for an evolutionary algorithm to find the optimal solution within the finite cost of function evaluations. Learning-based methods for neural combinatorial optimization also present challenges in terms of scalability. Models trained on small-scale instances suffer from serious performance degradation on large-scale test cases. Thus, improving model scalability and optimization efficiency on large-scale cases remains an open challenge in real-world optimization tasks [238,305].

6.2. Graph transferability

Graph transferability is a challenging yet important topic in COPs. A key expectation for GNNs is the ability to generalize across unseen graphs within the same problem. This capability increases the efficiency of solving NP-hard COPs—not only in terms of time but also in reducing computational resource demands. Moreover, it is hypothesized that COPs of similar types, as illustrated in Fig. 1, should exhibit transferability, allowing a single well-trained GNN model to address multiple problems effectively. For instance, questions arise such as whether a single GNN can resolve all types of routing problems, or whether a model trained on TSPs can be adapted to solve VRPs. A particularly promising approach involves leveraging knowledge transfer by assessing the similarity among various COPs. Consequently, through techniques such as self-supervised learning, a GNN model can be trained to generalize across different COPs, potentially leading to broad applicability and enhanced problem-solving capabilities [25].

6.3. Complexity

In COPs, graphs used to represent different problems exhibit varying levels of complexity [306,307]. For some COPs, such as graph coloring, matching problems, and MVC, the associated graphs are relatively simple. Conversely, graphs in problems such as routing problems are inherently more complex, often incorporating specific attributes on their nodes and/or edges. Furthermore, the COP domain includes heterogeneous graphs, where the nodes within a single graph may represent different entities or concepts. Tailored approaches are required to address the challenges associated with these diverse graph structures. For simple graphs, the focus is often on extracting meaningful insights from the topology itself and understanding the underlying logic specific to each problem. In contrast, solving COPs represented by more complex graphs requires a robust mechanism to effectively utilize the available node and edge features and to discern the intricate relationships among various node types.

6.4. Trustworthiness

The issue of trustworthiness in combinatorial optimization has recently been attracting increasing attention. With the increased computing power of edge devices and growing demand for data privacy, federated learning has emerged as a novel paradigm for privacy-preserving distributed optimization. Federated learning allows multiple clients to collaboratively train neural network models while keeping local data private. The federated averaging algorithm enables model parameters to be shared and updated iteratively among participating clients. However, it has been argued that the transmission of weight parameters could lead to the exposure of sensitive data to some extent. Secure computing methods such as differential privacy and homomorphic encryption have been explored on top of federated learning, but the effectiveness of such algorithms in the context of federated combinatorial optimization is still under discussion [308,309].

7. Conclusions

In this survey, we conducted a comprehensive review of recent applications of GNNs in addressing COPs. Firstly, we provided an introduction to COPs, distinguishing between graph-structured and non-graph-structured problems, followed by a discussion on the foundational concepts of GNNs and recent methodologies within this domain. We offered an extensive

overview of GNN-based learning techniques for COPs, highlighting both traditional optimization algorithms and innovative applications of GNNs. A major contribution of this work is a unified graph framework tailored for COPs using GNNs. This framework can increase the ability to represent graph-structured problems and convert non-graph-structured problems into graph structures, facilitating the application of graph decomposition and simplification techniques. This capability is significant, as it broadens the applicability of GNNs to a wider array of optimization challenges, potentially increasing efficiency and solution quality across various domains.

The proposed unified graph framework based on GNNs can seamlessly integrate with existing graph-based learning techniques and traditional optimization methods. This framework's ability to handle multi-objective and constrained optimization scenarios underlines its importance in advancing the field of computational optimization. Moreover, it introduces a transformative approach to COPs, especially in dynamically changing environments. By exploring various directions, the proposed unified framework can significantly advance the utility and effectiveness of GNNs in solving complex COPs, aligning both theoretical and practical aspects. We hope that this survey will serve as a cornerstone for future research, inspiring innovative solutions and fostering a deeper understanding of the GNN-based learning approach in optimization.

Acknowledgments

This work was supported in part by the International Collaboration Fund for Creative Research Teams (ICFCRT) of National Natural Science Foundation of China (W2441019), the National Natural Science Foundation of China (62576116 and 62136003), the Natural Science Foundation of Guangdong Province (2026A1515011836), the Major Program of the Natural Science Foundation of Zhejiang Province (D25F020001), the Young Researcher Startup Program of Guangdong–Hong Kong–Macao Center for Applied Mathematics (2025A1515060002), and the Guangdong Philosophy and Social Science Foundation Regular Project (GD25YGG26).

References

[1] Bengio Y, Lodi A, Prouvost A. Machine learning for combinatorial optimization: a methodological tour d'horizon. Eur J Oper Res 2021;290(2):405–21.

[2] Jiang M, Shan K, He C, Li C. Efficient combinatorial optimization by quantum-inspired parallel annealing in analogue memristor crossbar. Nat Commun 2023;14:5927.

[3] Yin X, Qian Y, Vardar A, Günther M, Müller F, Laleni N, et al. Ferroelectric compute-in-memory annealer for combinatorial optimization problems. Nat Commun 2024;15:2419.

[4] Cautereels C, Smets J, Bircham P, De Ruysscher D, Zimmermann A, De Rijk P, et al. Combinatorial optimization of gene expression through recombinase-mediated promoter and terminator shuffling in yeast. Nat Commun 2024;15:1112.

[5] Ising machines offer an optimized solution. Nat Electron 2023;6(10):717.

[6] Si J, Yang S, Cen Y, Chen J, Huang Y, Yao Z, et al. Energy-efficient superparamagnetic Ising machine and its application to traveling salesman problems. Nat Commun 2024;15:3457.

[7] Wang R, Wang G, Sun J, Deng F, Chen J. Flexible job shop scheduling via dual attention network-based reinforcement learning. IEEE Trans Neural Netw Learn Syst 2023;35(3):3091–102.

[8] Min Y, Bai Y, Gomes CP. Unsupervised learning for solving the travelling salesman problem. In: Oh A, Naumann T, Globerson A, Saenko K, Hardt M, Levine S, editors. Proceedings of the 37th International Conference on Neural Information Processing Systems; 2023 Dec 10–16; New Orleans, LA, USA. Red Hook: Curran Associates, Inc.; 2023. p. 47264–78.

[9] Jiang Y, Zhan ZH, Tan KC, Zhang J. A bi-objective knowledge transfer framework for evolutionary many-task optimization. IEEE Trans Evol Comput 2022;27(9):1514–28.

[10] Zhang X, Liu S, Zhao Z, Yang S. A decomposition-based evolutionary algorithm with clustering and hierarchical estimation for multi-objective fuzzy flexible jobshop scheduling. IEEE Trans Evol Comput 2026;30(1):2–15.

[11] Sato R. A survey on the expressive power of graph neural networks. 2020. arXiv:2003.04078.

[12] Zhou J, Cui G, Hu S, Zhang Z, Yang C, Liu Z, et al. Graph neural networks: a review of methods and applications. AI Open 2020;1:57–81.

Journal Pre-proofs

[13] Asif NA, Sarker Y, Chakrabortty RK, Ryan MJ, Ahamed MH, Saha DK, et al. Graph neural network: a comprehensive review on non-Euclidean space. IEEE Access 2021;9:60588–606.

[14] Vesselinova N, Steinert R, Perez-Ramirez DF, Boman M. Learning combinatorial optimization on graphs: a survey with applications to networking. IEEE Access 2020;8:120388–416.

[15] Huang T, Ma Y, Zhou Y, Huang H, Chen D, Gong Z, et al. A review of combinatorial optimization with graph neural networks. In: Proceedings of 2019 5th International Conference on Big Data and Information Analytics; 2019 Jul 8–10; Kunming, China. Piscataway: IEEE; 2019. p. 72–7.

[16] Peng Y, Choi B, Xu J. Graph learning for combinatorial optimization: a survey of state-of-the-art. Data Sci Eng 2021;6(2):119–41.

[17] Meyer U, Sanders P. δ-stepping: a parallel single source shortest path algorithm. In: Bilardi G, Italiano GF, Pietracaprina A, Pucci G, editors. Algorithms—ESA'98. Proceedings of the 6th Annual European Symposium; 1998 Aug 24–26; Venice, Italy. Berlin: Springer; 1998. p. 393–404.

[18] Seidel R. On the all-pairs-shortest-path problem in unweighted undirected graphs. J Comput Syst Sci 1995;51(3):400–3.

[19] Katoh N, Ibaraki T, Mine H. An algorithm for finding *K* minimum spanning trees. SIAM J Comput 1981;10(2):247–55.

[20] Eppstein D. Offline algorithms for dynamic minimum spanning tree problems. J Algorithms 1994;17(2):237–50.

[21] Manerba D, Mansini R, Riera-Ledesma J. The traveling purchaser problem and its variants. Eur J Oper Res 2017;259(1):1–18.

[22] Lin C. A vehicle routing problem with pickup and delivery time windows, and coordination of transportable resources. Comput Oper Res 2011;38(11):1596–609..

[23] Mazzeo S, Loiseau I. An ant colony algorithm for the capacitated vehicle routing. Electron Notes Discrete Math 2004;18:181–6.

[24] Yan X, Huang H, Hao Z, Wang J. A graph-based fuzzy evolutionary algorithm for solving two-echelon vehicle routing problems. IEEE Trans Evol Comput 2020;24(1):129–41.

[25] Yan X, Jin Y, Ke X, Hao Z. Multi-task evolutionary optimization of multi-echelon location routing problems via a hierarchical fuzzy graph. Complex Intell Syst 2023;9(6):6845–62.

[26] Sering L, Skutella M. Multi-source multi-sink Nash flows over time. 2018. arXiv:1807.01098.

[27] Lahn N, Raghvendra S. A weighted approach to the maximum cardinality bipartite matching problem with applications in geometric settings. 2019. arXiv:1903.10445.

[28] Sotirov R. An efficient semidefinite programming relaxation for the graph partition problem. INFORMS J Comput 2014;26(1):16–30.

[29] Yan X, Zhong G, Jin Y, Ke X, Xie F, Huang G. Binary spectral clustering for multi-view data. Inf Sci 2024;677:120899..

[30] Hu X, Li J. Circular coloring and fractional coloring in planar graphs. J Graph Theory 2022;99(2):312–43.

[31] Fomin FV, Golovach PA, Lokshtanov D, Saurabh S, Zehavi M. Clique-width III: Hamiltonian cycle and the odd case of graph coloring. ACM Trans Algorithms 2018;15:9.

[32] Karp RM, Vazirani UV, Vazirani VV. An optimal algorithm for on-line bipartite matching. In: Ortiz H, editor. Proceedings of the 22nd Annual ACM Symposium on Theory of Computing; 1990 May 13–17; Baltimore, MD, USA. New York City: Association for Computing Machinery; 1990. p. 352–8.

[33] Iwama K, Miyazaki S. A survey of the stable marriage problem and its variants. In: Kurohashi S, editor. International Conference on Informatics Education and Research for Knowledge-Circulating Society (ICKS 2008); 2008 Jan 17; Kyoto, Japan. Piscataway: IEEE; 2008. p. 131–6.

[34] Minieka E. The Chinese postman problem for mixed networks. Manage Sci 1979;25(7):643–8.

[35] Tarjan RE, Trojanowski AE. Finding a maximum independent set. SIAM J Comput 1977;6(3):537–46.

[36] Bomze IM, Budinich M, Pardalos PM, Pelillo M. The maximum clique problem. In: Du DZ, Pardalos PM, editors. Handbook of combinatorial optimization. Boston: Springer; 1999. p. 1–74.

[37] Henning MA, Löwenstein C, Rautenbach D, Southey J. Disjoint dominating and total dominating sets in graphs. Discrete Appl Math 2010;158(15):1615–23.

Journal Pre-proofs

[38] Sunil Chandran L, Das A, Rajendraprasad D, Varma NM. Rainbow connection number and connected dominating sets. J Graph Theory 2012;71(2):206–18.

[39] Zhou Z, Das S, Gupta H. Connected *K*-coverage problem in sensor networks. In: Proceedings of the 13th International Conference on Computer Communications and Networks; 2004 Oct 11–13; Chicago, IL, USA. Piscataway: IEEE; 2004. p. 373–8.

[40] Chekuri C, Kumar A. Maximum coverage problem with group budget constraints and applications. In: Jansen K, Khanna S, Rolim JDP, Ron D, editors. Approximation, randomization and combinatorial optimization. Algorithms and techniques. Berlin: Springer; 2004. p. 72–83.

[41] Liu S, Yan X, Jin Y. End-to-end Pareto set prediction with graph neural networks for multi-objective facility location. In: Emmerich M, Deutz A, Wang H, Kononova AV, Naujoks B, Li K, et al., editors. Evolutionary multi-criterion optimization: 12th international conference; 2023 Mar 20–24; Leiden, the Netherlands. Berlin: Springer; 2023. p. 147–61.

[42] Heydaribeni N, Anastasopoulos A. Distributed mechanism design for network resource allocation problems. IEEE Trans Netw Sci Eng 2020;7(2):621–36.

[43] Mohamaddiah MH, Abdullah A, Subramaniam S, Hussin M. A survey on resource allocation and monitoring in cloud computing. Int J Mach Learn Comput 2014;4:31–8.

[44] Xie J, Gao L, Peng K, Li X, Li H. Review on flexible job shop scheduling. IET Coll Intel Manuf. 2019;1(3):67–77.

[45] Dhiflaoui M, Nouri HE, Driss OB. Dual-resource constraints in classical and flexible job shop problems: a state-of-the-art review. Procedia Comput Sci 2018;126:1507–15.

[46] Lodi A, Martello S, Monaci M, Vigo D. Two-dimensional bin packing problems. In: Paschos VT, editors. Paradigms of combinatorial optimization: problems and new approaches. Hoboken: John Wiley & Sons, Inc.; 2014. p. 107–29.

[47] Martello S, Pisinger D, Vigo D. The three-dimensional bin packing problem. Oper Res 2000;48(2):256–67.

[48] Kang J, Park S. Algorithms for the variable sized bin packing problem. Eur J Oper Res 2003;147(2):365–72.

[49] Kellerer H, Pferschy U, Pisinger D. The bounded knapsack problem. In: Knapsack problems. Berlin: Springer; 2004. p. 185–209.

[50] Andonov R, Poirriez V, Rajopadhye S. Unbounded knapsack problem: dynamic programming revisited. Eur J Oper Res 2000;123(2):394–407.

[51] Cherri AC, Arenales MN, Yanasse HH, Poldi KC, Vianna ACG. The one-dimensional cutting stock problem with usable leftovers–a survey. Eur J Oper Res 2014;236(2):395–402.

[52] Oliveira Ó, Gamboa D, Silva E. An introduction to the two-dimensional rectangular cutting and packing problem. Int Trans Oper Res 2023;30(6):3238–66.

[53] Wu F, Duan J, Ai P, Chen Z, Yang Z, Zou X. Rachis detection and three-dimensional localization of cut off point for vision-based banana robot. Comput Electron Agric 2022;198:107079.

[54] Golab L, Korn F, Li F, Saha B, Srivastava D. Size-constrained weighted set cover. In: Proceedings of 2015 IEEE 31st International Conference on Data Engineering; 2015 Apr 13–17; Seoul, Republic of Korea. Piscataway: IEEE; 2015. p. 879–90.

[55] Indyk P, Mahabadi S, Rubinfeld R, Ullman J, Vakilian A, Yodpinyanee A. Fractional set cover in the streaming model. In: Approximation, randomization, and combinatorial optimization. Algorithms and techniques. Wadern: Dagstuhl Publishing; 2017. p. 12.

[56] De Moura L, Bjørner N. Satisfiability modulo theories: introduction and applications. Commun ACM 2011;54(9):69–77.

[57] Narodytska N, Bacchus F. Maximum satisfiability using core-guided MaxSAT resolution. In: Proceedings of the 28th AAAI Conference on Artificial Intelligence; 2014 Jul 27–31; Québec City, QC, Canada. Palo Alto: AAAI Press; 2014. p. 2717–23.

[58] Beyersdorff O, Janota M, Lonsing F, Seidl M. Quantified Boolean formulas. In: Biere A, Heule M, van Maaren H, Walsh T, editors. Handbook of satisfiability. 2nd ed. Amsterdam: IOS Press; 2021. p. 1177–1221.

[59] Galbrun E, Pelechrinis K, Terzi E. Urban navigation beyond shortest route: the case of safe paths. Inf Syst 2016;57:160–71.

[60] Dang T, Mascarich F, Khattak S, Papachristos C, Alexis K. Graph-based path planning for autonomous robotic exploration in subterranean environments. In: Proceedings of 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems; 2019 Nov 3–8; Macao, China. Piscataway: IEEE; 2019. p. 3105–12.

Journal Pre-proofs

[61] Lu H, Li J, Guizani M. Secure and efficient data transmission for cluster-based wireless sensor networks. IEEE Trans Parallel Distrib Syst 2013;25:750–61.

[62] Kumar K. Technology for supporting supply chain management: introduction. Commun ACM 2001;44(6):58–61.

[63] Zetina CA, Contreras I, Fernández E, Luna-Mota C. Solving the optimum communication spanning tree problem. Eur J Oper Res 2019;273(1):108–17.

[64] Gamayanti N, Alkaff A, Mangatas R. Optimisasi multi depot vehicle routing problem (MDVRP) dengan variabel travel time menggunakan algoritma particle swarm optimization. JAVA J Electr Electron Eng 2015;13(1):18–22. Indonesian.

[65] Kucukoglu I, Dewil R, Cattrysse D. The electric vehicle routing problem and its variations: a literature review. Comput Ind Eng 2021;161:107650.

[66] Schrijver A. On the history of the transportation and maximum flow problems. Math Program 2002;91(3):437–45.

[67] Ciciriello P, Mottola L, Picco GP. Efficient routing from multiple sources to multiple sinks in wireless sensor networks. In: Langendoen K, Voigt T, editors. Wireless sensor networks. EWSN 2007. Berlin: Springer; 2007. p. 34–50.

[68] Azad A, Buluç A, Pothen A. Computing maximum cardinality matchings in parallel on bipartite graphs via tree-grafting. IEEE Trans Parallel Distrib Syst 2017;28(1):44–59.

[69] Ma X, Yan X, Liu J, Zhong G. Simultaneous multi-graph learning and clustering for multiview data. Inf Sci 2022;593:472–87.

[70] Zhong G, Shu T, Huang G, Yan X. Multi-view spectral clustering by simultaneous consensus graph learning and discretization. Knowl Base Syst 2022;235:107632.

[71] Maus Y. Distributed graph coloring made easy. ACM Trans Parallel C 2023;10(4):21.

[72] Kierstead HA. A simple competitive graph coloring algorithm. J Comb Theory Ser B 2000;78(1):57–68.

[73] Marx D. Complexity of clique coloring and related problems. Theor Comput Sci 2011;412(29):3487–500.

[74] Rossi RA, Ahmed NK. Coloring large complex networks. Soc Netw Anal Min 2014;4(1):228.

[75] Hajiaghayi M, Wijting C, Ribeiro C, Hajiaghayi MT. Efficient and practical resource block allocation for LTE-based D2D network via graph coloring. Wirel Netw 2014;20(4):611–24.

[76] Bodlaender HL, Kolay S, Pieterse A. Parameterized complexity of conflict-free graph coloring. SIAM J Discrete Math 2021;35(3):2003–38.

[77] Mohar B. Circular colorings of edge-weighted graphs. J Graph Theory 2003;43(2):107–16.

[78] Jothiraj R, Jayakumar S, Venugopal P. Cyclic open shop scheduling in a paired triple connected dominator coloring of circular-arc graph. J Appl Sci Comput. 2019;6:675–80.

[79] Tabi Z, El-Safty KH, Kallus Z, Hága P, Kozsik T, Glos A, et al. Quantum optimization for the graph coloring problem with space-efficient embedding. In: Proceedings of 2020 IEEE International Conference on Quantum Computing and Engineering; 2020 Oct 12–16; Denver, CO, USA. Piscataway: IEEE; 2020. p. 56–62.

[80] Szachniuk M, Cristina De Cola M, Felici G, Blazewicz J. The orderly colored longest path problem—a survey of applications and new algorithms. RAIRO-Oper Res 2014;48(1):25–51.

[81] Wang R, Yan J, Yang X. Neural graph matching network: learning Lawler's quadratic assignment problem with extension to hypergraph and multiple-graph matching. IEEE Trans Pattern Anal Mach Intell 2021;44:5261–79.

[82] Gaspars-Wieloch H. The assignment problem in human resource project management under uncertainty. Risks 2021;9(1):25.

[83] Xu X, Hao J, Yu L, Deng Y. Fuzzy optimal allocation model for task–resource assignment problem in a collaborative logistics network. IEEE Trans Fuzzy Syst 2019;27(5):1112–25.

[84] Grötschel M, Yuan YX. Euler, Mei-ko Kwan, Königsberg, and a Chinese postman. In: Grötschel M, editor. Optimization stories: 21st International Symposium on Mathematical Programming; 2012 Aug 19–24; Berlin, Germany. Berlin: EMS Press; 2012. p. 43–50.

[85] Çomaklı Sökmen Ö, Emeç Ş, Yilmaz M, Akkaya G. An overview of Chinese postman problem. In: Proceedings of the 3rd International Conference on Advanced Engineering Technologies; 2019 Sep 19–21; Erzurum, Turkey. Bayburt: Bayburt University; 2019. p. 1219–28.

Journal Pre-proofs

[86] Jiang H, Talebi AA, Shao Z, Sadati SH, Rashmanlou H. New concepts of vertex covering in cubic graphs with its applications. Mathematics 2022;10(3):307.

[87] Hedar AR, Ismail R, El-Sayed GA, Khayyat KMJ. Two meta-heuristics designed to solve the minimum connected dominating set problem for wireless networks design and management. J Netw Syst Manage 2019;27(3):647–87.

[88] Yu J, Wang N, Wang G, Yu D. Connected dominating sets in wireless ad hoc and sensor networks—a comprehensive survey. Comput Commun 2013;36(2):121–34.

[89] Fagerholt K, Christiansen M. A combined ship scheduling and allocation problem. J Oper Res Soc 2000;51(7):834–42.

[90] Bouajaja S, Dridi N. A survey on human resource allocation problem and its applications. Oper Res 2017;17:339–69.

[91] Applegate D, Cook W. A computational study of the job-shop scheduling problem. ORSA J Comput 1991;3(2):149–56.

[92] Katoh N, Ibaraki T. Resource allocation problems. In: Du DZ, Pardalos PM, editors. Handbook of combinatorial optimization. Boston: Springer; 1998. p. 905–1006.

[93] Monma CL, Schrijver A, Todd MJ, Wei VK. Convex resource allocation problems on directed acyclic graphs: duality, complexity, special cases, and extensions. Math Oper Res 1990;15(4):736–48.

[94] Destouet C, Tlahig H, Bettayeb B, Mazari B. Flexible job shop scheduling problem under Industry 5.0: a survey on human reintegration, environmental consideration and resilience improvement. J Manuf Syst 2023;67:155–73.

[95] Leao AA, Toledo FM, Oliveira JF, Carravilla MA, Alvarez-Valdés R. Irregular packing problems: a review of mathematical models. Eur J Oper Res 2020;282(3):803–22.

[96] Lodi A, Martello S, Vigo D. Recent advances on two-dimensional bin packing problems. Discrete Appl Math 2002;123(1–3):379–96.

[97] Chu PC, Beasley JE. A genetic algorithm for the multidimensional knapsack problem. J Heuristics 1998;4(1):63–86.

[98] Munien C, Ezugwu AE. Metaheuristic algorithms for one-dimensional bin-packing problems: a survey of recent advances and applications. J Intell Inf Syst 2021;30(1):636–63.

[99] Seiden SS. On the online bin packing problem. J Assoc Comput Mach 2002;49(5):640–71.

[100] Ross KW, Tsang DH. The stochastic knapsack problem. IEEE Trans Commun 1989;37(7):740–7.

[101] Fréville A. The multidimensional 0–1 knapsack problem: an overview. Eur J Oper Res 2004;155(1):1–21.

[102] Liu J, Zhang K, Yan X, Zhang Q. A configuration space evolutionary algorithm with local minimizer for weighted circles packing problem. Expert Syst Appl 2024;238:121768.

[103] Chen Y, Song X, Ouelhadj D, Cui Y. A heuristic for the skiving and cutting stock problem in paper and plastic film industries. Int Trans Oper Res 2019;26(1):157–79.

[104] Delorme M, Iori M. Enhanced pseudo-polynomial formulations for bin packing and cutting stock problems. INFORMS J Comput 2020;32(1):101–19.

[105] Caprara A, Toth P, Fischetti M. Algorithms for the set covering problem. Ann Oper Res 2000;98(1–4):353–71.

[106] Marchiori E, Steenbeek A. An evolutionary algorithm for large scale set covering problems with application to airline crew scheduling. In: Cagnoni S, editor. Real-world applications of evolutionary computing. EvoWorkshops 2000. Berlin: Springer; 2000. p. 370–84.

[107] Liu H, Xing K, Zhou M, Han L, Wang F. Transition cover-based design of Petri net controllers for automated manufacturing systems. IEEE Trans Syst Man Cybern Syst 2014;44(2):196–208.

[108] Owais M, Osman MK, Moussa G. Multi-objective transit route network design as set covering problem. IEEE Trans Intell Transp Syst 2016;17(3):670–9.

[109] Zamani H, Dumais S, Craswell N, Bennett P, Lueck G. Generating clarifying questions for information retrieval. In: Huang Y, King I, Liu TY, van Steen M, editors. Proceedings of the Web Conference 2020; 2020 Apr 20–24; Taipei, China. New York City: Association for Computing Machinery; 2020. p. 418–28.

Journal Pre-proofs

[110] Marques-Silva J. Practical applications of Boolean Satisfiability. In: Proceedings of 2008 9th International Workshop on Discrete Event Systems; 2008 May 28–30; Gothenburg, Sweden. Piscataway: IEEE; 2008. p. 74–80.

[111] Nguyen AHN, Aono M, Hara-Azumi Y. FPGA-based hardware/software co-design of a bio-inspired SAT solver. IEEE Access 2020;8:49053–65.

[112] Lynce I, Marques-Silva J. SAT in bioinformatics: making the case with haplotype inference. In: Biere A, Gomes CP, editors. Theory and applications of satisfiability testing—SAT 2006. Berlin: Springer; 2006. p. 136–41.

[113] Cambria E, White B. Jumping NLP curves: a review of natural language processing research. IEEE Comput Intell Mag 2014;9(2):48–57.

[114] Li Z, Liu F, Yang W, Peng S, Zhou J. A survey of convolutional neural networks: analysis, applications, and prospects. IEEE Trans Neural Netw Learn Syst 2021;33(12):6999–7019.

[115] Park J, Chun J, Kim SH, Kim Y, Park J. Learning to schedule job-shop problems: representation and policy learning using graph neural network and reinforcement learning. Int J Prod Res 2021;59(11):3360–77.

[116] Wang X, Huang T, Wang D, Yuan Y, Liu Z, He X, et al. Learning intents behind interactions with knowledge graph for recommendation. In: Proceedings of the Web Conference 2021; 2021 Apr 19–23; Ljubljana, Slovenia. New York City: Association for Computing Machinery; 2021. p. 878–87.

[117] La Gatta V, Moscato V, Postiglione M, Sperli G. An epidemiological neural network exploiting dynamic graph structured data applied to the COVID-19 outbreak. IEEE Trans Big Data 2021;7(1):45–55.

[118] Cui P, Wang X, Pei J, Zhu W. A survey on network embedding. IEEE Trans Knowl Data Eng 2019;31(5):833–52.

[119] Zhang D, Yin J, Zhu X, Zhang C. Network representation learning: a survey. IEEE Trans Big Data 2020;6(1):3–28.

[120] Perozzi B, Al-Rfou R, Skiena S. DeepWalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 2014 Aug 24–27; New York City, NY, USA. New York City: Association for Computing Machinery; 2014. p. 701–10.

[121] Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. 2013. arXiv:1301.3781

[122] Grover A, Leskovec J. node2vec: scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 2016 Aug 13–17; San Francisco, CA, USA. New York City: Association for Computing Machinery; 2016. p. 855–64.

[123] Li J, Zhu J, Zhang B. Discriminative deep random walk for network classification. In: Erk K, Smith NA, editors. Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics; 2016 Aug 7–12; Berlin, Germany. Kerrville: Association for Computational Linguistics; 2016. p. 1004–13.

[124] Chen J, Zhang Q, Huang X. Incorporate group information to enhance network embedding. In: Proceedings of the 25th ACM International Conference on Information and Knowledge Management; 2016 Oct 24–28; Indianapolis, IN, USA. New York City: Association for Computing Machinery; 2016. p. 1901–4.

[125] Tang J, Qu M, Wang M, Zhang M, Yan J, Mei Q. LINE: large-scale information network embedding. In: Proceedings of the 24th International Conference on World Wide Web; 2015 May 18–22; Florence, Italy. New York City: Association for Computing Machinery; 2015. p. 1067–77.

[126] Zhang D, Yin J, Zhu X, Zhang C. Collective classification via discriminative matrix factorization on sparsely labeled networks. In: Proceedings of the 25th ACM International Conference on Information and Knowledge Management; 2016 Oct 24–28; Indianapolis, IN, USA. New York City: Association for Computing Machinery; 2016. p. 1563–72.

[127] Lü L, Zhou T. Link prediction in complex networks: a survey. Physica A 2011;390(6):1150–70.

[128] Zhang C, Zhang K, Yuan Q, Peng H, Zheng Y, Hanratty T, et al. Regions, periods, activities: uncovering urban dynamics via cross-modal representation learning. In: Proceedings of the 26th International Conference on World Wide Web; 2017 Apr 3–7; Perth, WA, Australia. New York City: Association for Computing Machinery; 2017. p. 361–70.

[129] Tang J, Liu J, Zhang M, Mei Q. Visualizing large-scale and high-dimensional data. In: Proceedings of the 25th International Conference on World Wide Web; 2016 Apr 11–15; Montréal, QC, Canada. New York City: Association for Computing Machinery; 2016. p. 287–97.

[130] Scarselli F, Gori M, Tsoi AC, Hagenbuchner M, Monfardini G. The graph neural network model. IEEE Trans Neural Netw 2009;20(1):61–80.

[131] Micheli A. Neural network for graphs: a contextual constructive approach. IEEE Trans Neural Netw 2009;20(3):498–511.

Journal Pre-proofs

[132] Gallicchio C, Micheli A. Graph Echo State Networks. In: Proceedings of the 2010 International Joint Conference on Neural Networks; 2010 Jul 18–23; Barcelona, Spain. Piscataway: IEEE; 2010. p. 1–8.

[133] Li Y, Tarlow D, Brockschmidt M, Zemel R. Gated graph sequence neural networks. 2015. arXiv:1511.05493.

[134] Kipf TN, Welling M. Semi-supervised classification with graph convolutional networks. In: Proceedings of the 5th International Conference on Learning Representations; 2017 Apr 24–26; Toulon, France; 2017.

[135] Defferrard M, Bresson X, Vandergheynst P. Convolutional neural networks on graphs with fast localized spectral filtering. In: Lee DD, von Luxburg U, Garnett R, Sugiyama M, Guyon I, editors. Proceedings of the 30th International Conference on Neural Information Processing Systems; 2016 Dec 5–10; Barcelona, Spain. Red Hook: Curran Associates, Inc.; 2016. p. 3844–52.

[136] Hamilton W, Ying Z, Leskovec J. Inductive representation learning on large graphs. In: von Luxburg U, Guyon I, Bengio S, Wallach H, Fergus R, editors. Proceedings of the 31st International Conference on Neural Information Processing Systems; 2017 Dec 4–9; Long Beach, CA, USA. Red Hook: Curran Associates, Inc.; 2017. p. 1025–35.

[137] Gilmer J, Schoenholz SS, Riley PF, Vinyals O, Dahl GE. Neural message passing for quantum chemistry. In: Precup D, Teh YW, editors. Proceedings of the 34th International Conference on Machine Learning; 2017 Aug 6–11; Sydney, NSW, Australia. New York City: PMLR; 2017. p. 1263–72.

[138] Veličković P, Cucurull G, Casanova A, Romero A, Lio P, Bengio Y. Graph attention networks. 2018. arXiv:1710.10903.

[139] Levie R, Monti F, Bresson X, Bronstein MM. CayleyNets: graph convolutional neural networks with complex rational spectral filters. IEEE Trans Signal Process 2019;67(1):97–109.

[140] Chami I, Ying Z, Ré C, Leskovec J. Hyperbolic graph convolutional neural networks. In: Wallach HM, Larochelle H, Beygelzimer A, d'Alché-Buc F, Fox EB, editors. Proceedings of the 33rd International Conference on Neural Information Processing Systems; 2019 Dec 8–14; Vancouver, BC, Canada; Red Hook: Curran Associates, Inc.; 2019. p. 4868–79.

[141] Xu K, Hu W, Leskovec J, Jegelka S. How powerful are graph neural networks? 2019. arXiv:1810.00826.

[142] Bianchi FM, Grattarola D, Livi L, Alippi C. Graph neural networks with convolutional ARMA filters. IEEE Trans Pattern Anal Mach Intell 2021;44:3496–507.

[143] Veličković P, Fedus W, Hamilton WL, Liò P, Bengio Y, Hjelm RD. Deep graph infomax. 2018. arXiv:1809.10341.

[144] Gao H, Ji S. Graph U-Nets. In: Proceedings of the 36th International Conference on Machine Learning; 2019 Jun 9–15; Long Beach, CA, USA. New York City: PMLR; 2019. p. 2083–92.

[145] Zeng H, Zhou H, Srivastava A, Kannan R, Prasanna V. GraphSAINT: graph sampling based inductive learning method. 2019. arXiv:1907.04931.

[146] Liu M, Wang Z, Ji S. Non-local graph neural networks. IEEE Trans Pattern Anal Mach Intell. 2022;44(12):10270–6.

[147] Zhong Z, Li CT, Pang J. Hierarchical message-passing graph neural networks. Data Min Knowl Discov 2023;37(1):381–408.

[148] Zhu Z, Zhang Z, Xhonneux LP, Tang J. Neural Bellman–Ford networks: a general graph neural network framework for link prediction. Adv Neural Inf Process Syst 2021;34:29476–90.

[149] Fan S, Wang X, Shi C, Cui P, Wang B. Generalizing graph neural networks on out-of-distribution graphs. IEEE Trans Pattern Anal Mach Intell 2023;46(1):322–37.

[150] Luan S, Hua C, Lu Q, Zhu J, Zhao M, Zhang S, et al. Revisiting heterophily for graph neural networks. Adv Neural Inf Process Syst 2022;35:1362–75.

[151] Feng J, Chen Y, Li F, Sarkar A, Zhang M. How powerful are *K*-hop message passing graph neural networks. Adv Neural Inf Process Syst 2022;35:4776–90.

[152] Bouritsas G, Frasca F, Zafeiriou S, Bronstein MM. Improving graph neural network expressivity via subgraph isomorphism counting. IEEE Trans Pattern Anal Mach Intell 2023;45(1):657–68.

[153] Wu Z, Pan S, Long G, Jiang J, Chang X, Zhang C. Connecting the dots: multivariate time series forecasting with graph neural networks. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 2020 Aug 23–27; San Diego, CA, USA. New York City: Association for Computing Machinery; 2020. p. 753–63.

Journal Pre-proofs

[154] Li M, Zhu Z. Spatial–temporal fusion graph neural networks for traffic flow forecasting. Proc Conf AAAI Artif Intell 2021;35(5):4189–96.

[155] Chen J, Zheng L, Hu Y, Wang W, Zhang H, Hu X. Traffic flow matrix-based graph neural network with attention mechanism for traffic flow prediction. Inf Fusion 2024;104:102146.

[156] Jin W, Ma Y, Liu X, Tang X, Wang S, Tang J. Graph structure learning for robust graph neural networks. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 2020 Aug 23–27; San Diego, CA, USA. New York City: Association for Computing Machinery; 2020. p. 66–74.

[157] Gasteiger J, Becker F, Günnemann S. GemNet: universal directional graph neural networks for molecules. Adv Neural Inf Process Syst 2021;34:6790–802.

[158] Réau M, Renaud N, Xue LC, Bonvin AM. DeepRank-GNN: a graph neural network framework to learn patterns in protein–protein interfaces. Bioinformatics 2023;39(1):btac759.

[159] Truong-Quoc C, Lee JY, Kim KS, Kim DN. Prediction of DNA origami shape using graph neural network. Nat Mater 2024;23(7):984–2.

[160] Chen H, Bei Y, Shen Q, Xu Y, Zhou S, Huang W, et al. Macro graph neural networks for online billion-scale recommender system. 2024. arXiv:2401.14939.

[161] Gu Y, Peng S, Li Y, Gao L, Dong Y. FC-HGNN: a heterogeneous graph neural network based on brain functional connectivity for mental disorder identification. Inf Fusion 2025;113:102619.

[162] Chen LY, Li YP. Uncertainty quantification with graph neural networks for efficient molecular design. Nat Commun 2025;16:3262.

[163] Chen S, Liu Y, Zhang Q, Shao Z, Wang Z. Multi-distance spatial-temporal graph neural network for anomaly detection in blockchain transactions. Adv Intell Syst 2025;7(8):2400898.

[164] Simonovsky M, Komodakis N. GraphVAE: towards generation of small graphs using variational autoencoders. In: Kůrková V, Manolopoulos Y, Hammer B, Iliadis L, Maglogiannis I, editors. Artificial neural networks and machine learning—ICANN 2018. Cham: Springer; 2018. p. 412–22.

[165] You J, Ying R, Ren X, Hamilton W, Leskovec J. GraphRNN: generating realistic graphs with deep auto-regressive models. In: Proceedings of the 35th International Conference on Machine Learning; 2018 Jul 10–15; Stockholm, Sweden. New York City: PMLR; 2018. p. 5708–17.

[166] Chen Z, Mao H, Li H, Jin W, Wen H, Wei X, et al. Exploring the potential of large language models (LLMs) in learning on graphs. SIGKDD Explor 2024;25(2):42–61.

[167] Zhu J, Rossi RA, Rao A, Mai T, Lipka N, Ahmed NK, et al. Graph neural networks with heterophily. In: Proceedings of the 35th AAAI Conference on Artificial Intelligence; 2021 Feb 2–9; online conference. Palo Alto: AAAI Press; 2021. p. 11168–76.

[168] Yang X, Yan M, Pan S, Ye X, Fan D. Simple and efficient heterogeneous graph neural network. In: Proceedings of the 37th AAAI Conference on Artificial Intelligence; 2023 Feb 7–14; Washington, DC, USA. Palo Alto: AAAI Press; 2023. p. 10816–24.

[169] Fu C, Zheng G, Huang C, Yu Y, Dong J. Multiplex heterogeneous graph neural network with behavior pattern modeling. In: Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining; 2023 Aug 6–10; Long Beach, CA, USA. New York City: Association for Computing Machinery; 2023. p. 482–94.

[170] Melton J, Krishnan S. muxGNN: multiplex graph neural network for heterogeneous graphs. IEEE Trans Pattern Anal Mach Intell 2023;45(9):11067–78.

[171] Wang S, Li Y, Wang D, Zhang W, Chen X, Dong D, et al. Echo state graph neural networks with analogue random resistive memory arrays. Nat Mach Intell 2023;5(2):104–13.

[172] Xue J, Jiang N, Liang S, Pang Q, Yabe T, Ukkusuri SV, et al. Quantifying the spatial homogeneity of urban road networks via graph neural networks. Nat Mach Intell 2022;4(3):246–57.

[173] Wang Y, Wang J, Cao Z, Barati Farimani A. Molecular contrastive learning of representations via graph neural networks. Nat Mach Intell 2022;4(3):279–87.

[174] Longa A, Azzolin S, Santin G, Cencetti G, Liò P, Lepri B, et al. Explaining the explainers in graph neural networks: a comparative study. ACM Comput Surv 2024;57(5):1–37.

[175] Wang X, Yan X, Jin Y. A graph neural network with negative message passing and uniformity maximization for graph coloring. Complex Intell Syst 2024;10(3):4445–55.

Journal Pre-proofs

[176] Li X, Sun L, Ling M, Peng Y. A survey of graph neural network based recommendation in social networks. Neurocomputing 2023;549:126441.

[177] Jiang W, Luo J. Graph neural network for traffic forecasting: a survey. Expert Syst Appl 2022;207:117921.

[178] Zheng X, Liu Y, Pan S, Zhang M, Jin D, Yu PS. Graph neural networks for graphs with heterophily: a survey. 2022. arXiv:2202.07082.

[179] Wu Z, Pan S, Chen F, Long G, Zhang C, Philip SY. A comprehensive survey on graph neural networks. IEEE Trans Neural Netw Learn Syst 2021;32(1):4–24.

[180] Wu F, Souza A, Zhang T, Fifty C, Yu T, Weinberger K. Simplifying graph convolutional networks. In: Proceedings of the 36th International Conference on Machine Learning; 2019 Jun 9–15; Long Beach, CA, USA. New York City: PMLR; 2019. p. 6861–71.

[181] Bo D, Wang X, Shi C, Shen H. Beyond low-frequency information in graph convolutional networks. In: Proceedings of the 35th AAAI Conference on Artificial Intelligence; 2021 Feb 2–9; online conference. Palo Alto: AAAI Press; 2021. p. 3950–57.

[182] Wei L, Zhao H, He Z. Learn layer-wise connections in graph neural networks. 2021. arXiv:2112.13585.

[183] Pei H, Wei B, Chang KCC, Lei Y, Yang B. Geom-GCN: geometric graph convolutional networks. 2020. arXiv:2002.05287.

[184] Chen Z, Wu Z, Lin Z, Wang S, Plant C, Guo W. AGNN: alternating graph-regularized neural networks to alleviate over-smoothing. IEEE Trans Neural Netw Learn Syst 2024;35(10):13764–76.

[185] Wu Z, Jain P, Wright M, Mirhoseini A, Gonzalez JE, Stoica I. Representing long-range context for graph neural networks with global attention. Adv Neural Inf Process Syst 2021;34:13266–79.

[186] Lin D, Lin J, Zhao L, Wang ZJ, Chen Z. Multilabel aerial image classification with a concept attention graph neural network. IEEE Trans Geosci Remote Sens 2021;60:1–12.

[187] Zhang H, Bai L. Few-shot link prediction for temporal knowledge graphs based on time-aware translation and attention mechanism. Neural Netw 2023;161:371–81.

[188] Wu W, Song C, Zhao J, Xu Z. Physics-informed gated recurrent graph attention unit network for anomaly detection in industrial cyber–physical systems. Inf Sci 2023;629:618–33.

[189] Kong Z, Jin X, Xu Z, Zhang B. Spatio-temporal fusion attention: a novel approach for remaining useful life prediction based on graph neural network. IEEE Trans Instrum Meas 2022;71:1–12.

[190] Sun Z, Yang Y. DIFUSCO: graph-based diffusion solvers for combinatorial optimization. Adv Neural Inf Process Syst 2024;36:3706–31.

[191] Lemos H, Prates M, Avelar P, Lamb L. Graph colouring meets deep learning: effective graph neural network models for combinatorial problems. In: Proceedings of 2019 IEEE 31st International Conference on Tools with Artificial Intelligence; 2019 Nov 4–6; Portland, OR, USA. Piscataway: IEEE; 2019. p. 879–85.

[192] Prates M, Avelar PH, Lemos H, Lamb LC, Vardi MY. Learning to solve NP-complete problems: a graph neural network for decision TSP. In: Proceedings of the 33rd AAAI Conference on Artificial Intelligence; 2019 Jan 27–Feb 1; Honolulu, HI, USA. Palo Alto: AAAI Press; 2019. p. 4731–8.

[193] Li Y, Gu C, Dullien T, Vinyals O, Kohli P. Graph matching networks for learning the similarity of graph structured objects. In: Proceedings of the 36th International Conference on Machine Learning; 2019 Jun 9–15; Long Beach, CA, USA. New York City: PMLR; 2019. p. 3835–45.

[194] Fey M, Lenssen JE, Morris C, Masci J, Kriege NM. Deep graph matching consensus. 2020. arXiv:2001.09621.

[195] Bai Y, Ding H, Gu K, Sun Y, Wang W. Learning-based efficient graph similarity computation via multi-scale convolutional set matching. In: Proceedings of the 34th AAAI Conference on Artificial Intelligence; 2020 Feb 7–12; New York City, NY, USA. Palo Alto, CA: AAAI Press; 2020. p. 3219–26.

[196] Joshi CK, Laurent T, Bresson X. An efficient graph convolutional network technique for the travelling salesman problem. 2019. arXiv:1906.01227.

[197] Xie Y, Xu Z, Zhang J, Wang Z, Ji S. Self-supervised learning of graph neural networks: a unified review. IEEE Trans Pattern Anal Mach Intell 2023;45(2):2412–29.

[198] Mitton J, Senn HM, Wynne K, Murray-Smith R. A graph VAE and graph transformer approach to generating molecular graphs. 2021. arXiv:2104.04345.

Journal Pre-proofs

[199] Samanta B, De A, Jana G, Gómez V, Chattaraj PK, Ganguly N, et al. NEVAE: a deep generative model for molecular graphs. J Mach Learn Res 2020;21(114):4556–88.

[200] Wang C, Pan S, Long G, Zhu X, Jiang J. MGAE: marginalized graph autoencoder for graph clustering. In: Proceedings of the 2017 ACM Conference on Information and Knowledge Management; 2017 Nov 6–10; Singapore, Singapore. New York City: Association for Computing Machinery; 2017. p. 889–98.

[201] Park J, Lee M, Chang HJ, Lee K, Choi JY. Symmetric graph convolutional autoencoder for unsupervised graph representation learning. In: Proceedings of the IEEE/CVF International Conference on Computer Vision; 2019 Oct 27–Nov 2; Seoul, Republic of Korea. Piscataway: IEEE; 2019. p. 6519–28.

[202] Li W, Li R, Ma Y, Chan SO, Pan D, Yu B. Rethinking graph neural networks for the graph coloring problem. 2022. arXiv:2208.06975.

[203] Schuetz MJ, Brubaker JK, Zhu Z, Katzgraber HG. Graph coloring with physics-inspired graph neural networks. Phys Rev Res 2022;4(4):043131.

[204] Wang X, Yan X, Jin Y. A graph neural network with negative message passing for graph coloring. 2023. arXiv:2301.11164.

[205] Schuetz MJ, Brubaker JK, Katzgraber HG. Combinatorial optimization with physics-inspired graph neural networks. Nat Mach Intell 2022;4(4):367–77.

[206] Tönshoff J, Ritzert M, Wolf H, Grohe M. Graph neural networks for maximum constraint satisfaction. Front Artif Intell 2021;3:580607.

[207] Duan H, Vaezipoor P, Paulus MB, Ruan Y, Maddison C. Augment with care: contrastive learning for combinatorial problems. In: Proceedings of the 39th International Conference on Machine Learning; 2022 Jul 17–23; Baltimore, MD, USA. New York City: PMLR; 2022. p. 5627–42.

[208] Xu H, Hui KH, Fu CW, Zhang H. TilinGNN: learning to tile with self-supervised graph neural network. ACM Tran Graphic 2020;39(4):129.

[209] Kool W, Van Hoof H, Welling M. Attention, learn to solve routing problems! 2018. arXiv:1803.08475.

[210] Nazari M, Oroojlooy A, Takác M, Snyder LV. Reinforcement learning for solving the vehicle routing problem. In: Bengio S, Wallach HM, Larochelle H, Grauman K, Cesa-Bianchi N, editors. Proceedings of the 32nd International Conference on Neural Information Processing Systems; 2018 Dec 3–8; Montréal, QC, Canada. Red Hook: Curran Associates, Inc.; 2018. p. 9861–71.

[211] Dai H, Khalil EB, Zhang Y, Dilkina B, Song L. Learning combinatorial optimization algorithms over graphs. In: von Luxburg U, Guyon I, Bengio S, Wallach H, Fergus R, editors. Proceedings of the 31st International Conference on Neural Information Processing Systems; 2017 Dec 4–9; Long Beach, CA, USA. Red Hook: Curran Associates, Inc.; 2017. p. 6351–61.

[212] Tönshoff J, Kisin B, Lindner J, Grohe M. One model, Any CSP: graph neural networks as fast global search heuristics for constraint satisfaction. 2022. arXiv:2208.10227.

[213] Blum C, Puchinger J, Raidl GR, Roli A. Hybrid metaheuristics in combinatorial optimization: a survey. Appl Soft Comput 2011;11(6):4135–51.

[214] Dantzig G, Fulkerson R, Johnson S. Solution of a large-scale traveling-salesman problem. J Oper Res Soc Am 1954;2(4):393–410.

[215] Savelsbergh M. A branch-and-price algorithm for the generalized assignment problem. Oper Res 1997;45(6):831–41.

[216] Applegate DL, Bixby R E, Chvatál V, Cook WJ. The traveling salesman problem: a computational study. Princeton: Princeton University Press; 2011.

[217] Helsgaun K. An extension of the Lin–Kernighan–Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems. Roskilde: Roskilde University; 2017.

[218] Helsgaun K. General *k*-opt submoves for the Lin–Kernighan TSP heuristic. Math Program Comput 2009;1(2–3):119–63.

[219] Colorni A, Dorigo M, Maffioli F, Maniezzo V, Righini G, Trubian M. Heuristics from nature for hard combinatorial optimization problems. Int Trans Oper Res 1996;3(1):1–21.

[220] Lawler EL, Wood DE. Branch-and-bound methods: a survey. Oper Res 1966;14(4):699–719.

[221] Gauvin C, Desaulniers G, Gendreau M. A branch-cut-and-price algorithm for the vehicle routing problem with stochastic demands. Comput Oper Res 2014;50:141–53.

[222] Reinelt G. TSPLIB—a traveling salesman problem library. ORSA J Comput 1991;3(4):376–84.

Journal Pre-proofs

[223] Gonçalves JF, Resende MG. Biased random-key genetic algorithms for combinatorial optimization. J Heuristics 2011;17(5):487–525.

[224] Bertsimas D, Tsitsiklis J. Simulated annealing. Stat Sci 1993;8(1):10–5.

[225] Slavík P. A tight analysis of the greedy algorithm for set cover. In: Proceedings of the 28th Annual ACM Symposium on Theory of Computing; 1996 May 22–24; Philadelphia, PA, USA. New York City: Association for Computing Machinery; 1996. p. 435–41.

[226] Johnson DS. Local optimization and the traveling salesman problem. In: Paterson MS, editor. Automata, languages and programming. ICALP 1990. Berlin: Springer; 1990. p. 446–61.

[227] Helsgaun K. An effective implementation of the Lin–Kernighan traveling salesman heuristic. Eur J Oper Res 2000;126(1):106–30.

[228] Vinyals O, Fortunato M, Jaitly N. Pointer networks. In: Cortes C, Lee DD, Sugiyama M, Garnett R, editors. Proceedings of the 29th International Conference on Neural Information Processing Systems; 2015 Dec 7–12; Montréal, QC, Canada. Cambridge: MIT Press; 2015. p. 2692–700.

[229] Bello I, Pham H, Le QV, Norouzi M, Bengio S. Neural combinatorial optimization with reinforcement learning. 2016. arXiv:1611.09940.

[230] Ma Q, Ge S, He D, Thaker D, Drori I. Combinatorial optimization by graph pointer networks and hierarchical reinforcement learning. 2019. arXiv:1911.04936.

[231] Wu Y, Song W, Cao Z, Zhang J, Lim A. Learning improvement heuristics for solving routing problems. IEEE Trans Neural Netw Learn Syst 2022;33(9):5057–69.

[232] Grinsztajn N, Furelos-Blanco D, Surana S, Bonnet C, Barrett TD. Winner takes it all: training performant RL populations for combinatorial optimization. In: Oh A, Naumann T, Globerson A, Saenko K, Hardt M, Levine S, editors. Proceedings of the 37th International Conference on Neural Information Processing Systems; 2023 Dec 10–16; New Orleans, LA, USA. Red Hook: Curran Associates, Inc.; 2023. p. 48485–509.

[233] Selsam D, Lamm M, Bünz B, Liang P, de Moura L, Dill DL. Learning a SAT solver from single-bit supervision. 2018. arXiv:1802.03685.

[234] Ireland D, Montana G. LeNSE: learning to navigate subgraph embeddings for large-scale combinatorial optimisation. In: Proceedings of the 39th International Conference on Machine Learning; 2022 Jul 17–23; Baltimore, MD, USA. New York City: PMLR; 2022. p. 9622–38.

[235] Li Z, Chen Q, Koltun V. Combinatorial optimization with graph convolutional networks and guided tree search. In: Bengio S, Wallach HM, Larochelle H, Grauman K, Cesa-Bianchi N, editors. Proceedings of the 32nd International Conference on Neural Information Processing Systems; 2018 Dec 3–8; Montréal, QC, Canada. Red Hook: Curran Associates, Inc.; 2018. p. 537–46.

[236] Nowak A, Villar S, Bandeira AS, Bruna J. Revised note on learning quadratic assignment with graph neural networks. In: Proceedings of 2018 IEEE Data Science Workshop; 2018 Jun 4–6; Lausanne, Switzerland. Piscataway: IEEE; 2018. p. 1–5.

[237] Manchanda S, Mittal A, Dhawan A, Medya S, Ranu S, Singh A. Learning heuristics over large graphs via deep reinforcement learning. 2019. arXiv:1903.03332.

[238] Fu ZH, Qiu KB, Zha H. Generalize a small pre-trained model to arbitrarily large TSP instances. In: Proceedings of the 35th AAAI Conference on Artificial Intelligence; 2021 Feb 2–9; online conference. Palo Alto: AAAI Press; 2021. p. 7474–82.

[239] Deudon M, Cournut P, Lacoste A, Adulyasak Y, Rousseau LM. Learning heuristics for the TSP by policy gradient. In: van Hoeve WJ, editor. Integration of constraint programming, artificial intelligence, and operations research. CPAIOR 2018. Cham: Springer; 2018. p. 170–81.

[240] Joshi CK, Cappart Q, Rousseau LM, Laurent T. Learning the travelling salesperson problem requires rethinking generalization. 2020. arXiv:2006.07054.

[241] Dwivedi VP, Joshi CK, Luu AT, Laurent T, Bengio Y, Bresson X. Benchmarking graph neural networks. J Mach Learn Res 2023;24:1–48.

[242] Hudson B, Li Q, Malencia M, Prorok A. Graph neural network guided local search for the traveling salesperson problem. 2021. arXiv:2110.05291.

[243] Abe K, Xu Z, Sato I, Sugiyama M. Solving NP-hard problems on graphs with extended AlphaGo zero. 2019. arXiv:1905.11623.

[244] Barrett T, Clements W, Foerster J, Lvovsky A. Exploratory combinatorial optimization with reinforcement learning. In: Proceedings of the 34th AAAI Conference on Artificial Intelligence; 2020 Feb 7–12; New York City, NY, USA. Palo Alto: AAAI Press; 2020. p. 3243–50.

[245] Ye H, Wang J, Cao Z, Liang H, Li Y. DeepACO: neural-enhanced ant systems for combinatorial optimization. In: Oh A, Naumann T, Globerson A, Saenko K, Hardt M, Levine S, editors. Proceedings of the 37th International Conference on Neural Information Processing Systems; 2023 Dec 10–16; New Orleans, LA, USA. Red Hook: Curran Associates, Inc.; 2023. p. 43706–28.

Journal Pre-proofs

[246] Drakulic D, Michel S, Mai F, Sors A, Andreoli JM. BQ-NCO: Bisimulation Quotienting for efficient neural combinatorial optimization. In: Oh A, Naumann T, Globerson A, Saenko K, Hardt M, Levine S, editors. Proceedings of the 37th International Conference on Neural Information Processing Systems; 2023 Dec 10–16; New Orleans, LA, USA. Red Hook: Curran Associates, Inc.; 2023. p. 77416–29.

[247] You J, Wu H, Barrett C, Ramanujan R, Leskovec J. G2SAT: learning to generate SAT formulas. In: Wallach HM, Larochelle H, Beygelzimer A, d'Alché-Buc F, Fox EB, editors. Proceedings of the 33rd International Conference on Neural Information Processing Systems; 2019 Dec 8–14; Vancouver, BC, Canada. Red Hook: Curran Associates, Inc.; 2019. p. 10553 - 10564

[248] Li Z, Si X. NSNet: a general neural probabilistic framework for satisfiability problems. Adv Neural Inf Process Syst 2022;35:25573–85.

[249] Sutskever I, Vinyals O, Le QV. Sequence to sequence learning with neural networks. Adv Neural Inf Process Syst 2014;2:3104–12.

[250] Khoshraftar S, An A. A survey on graph representation learning methods. ACM Trans Intell Syst Technol 2024;15(1):19.

[251] Guzman-Rivera A, Batra D, Kohli P. Multiple choice learning: learning to produce multiple structured outputs. Adv Neural Inf Process Syst 2012;2:1799–807.

[252] Bresson X, Laurent T. Residual gated graph convnets. 2017. arXiv:1711.07553.

[253] Dai H, Dai B, Song L. Discriminative embeddings of latent variable models for structured data. In: Balcan MF, Weinberger KQ, editors. Proceedings of the 33rd International Conference on International Conference on Machine Learning; 2016 Jun 19–24; New York City, NY, USA. New York City: PMLR; 2016. p. 2702–11.

[254] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need. In: von Luxburg U, Guyon I, Bengio S, Wallach H, Fergus R, editors. Proceedings of the 31st International Conference on Neural Information Processing Systems; 2017 Dec 4–9; Long Beach, CA, USA. Red Hook: Curran Associates, Inc.; 2017. p. 6000–8.

[255] Karalias N, Loukas A. Erdos goes neural: an unsupervised learning framework for combinatorial optimization on graphs. Adv Neural Inf Process Syst 2020;33:6659–72.

[256] Wang HP, Wu N, Yang H, Hao C, Li P. Unsupervised learning for combinatorial optimization with principled objective relaxation. Adv Neural Inf Process Syst 2022;35:31444–58.

[257] Saunshi N, Plevrakis O, Arora S, Khodak M, Khandeparkar H. A theoretical analysis of contrastive unsupervised representation learning. In: Chaudhuri K, Salakhutdinov R, editors. Proceedings of the 36th International Conference on Machine Learning; 2019 Jun 9–15; Long Beach, CA, USA. New York City: PMLR; 2019. p. 5628–37.

[258] Tosh C, Krishnamurthy A, Hsu D. Contrastive learning, multi-view redundancy, and linear models. In: Abernethy J, Agarwal S, editors. Proceedings of the 32nd International Conference on Algorithmic Learning Theory; 2021 Mar 15–17; online conference. New York City: PMLR; 2021. p. 1179–206.

[259] You Y, Chen T, Sui Y, Chen T, Wang Z, Shen Y. Graph contrastive learning with augmentations. Adv Neural Inf Process Syst 2020;33:5812–23.

[260] Hassani K, Khasahmadi AH. Contrastive multi-view representation learning on graphs. In: Daumé, H, Singh A, editors. Proceedings of the 37th International Conference on Machine Learning; 2020 Jul 13–18; online conference. New York City: PMLR; 2020. p. 4116–26.

[261] Jing M, Zhu Y, Zang T, Yu J, Tang F. Graph contrastive learning with adaptive augmentation for recommendation. In: Amini MR, Canu S, Fischer A, Guns T, Kralj Novak P, Tsoumakas G, editors. Machine learning and knowledge discovery in databases. ECML PKDD 2022. Cham: Springer; 2022. p. 590–605.

[262] Jiang Y, Huang C, Huang L. Adaptive graph contrastive learning for recommendation. In: Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining; 2023 Aug 6–10; Long Beach, CA, USA. New York City: Association for Computing Machinery; 2023. p. 4252–61.

[263] Maurya SK, Liu X, Murata T. Graph neural networks for fast node ranking approximation. ACM Trans Knowl Discov Data 2021;15(5):1–32.

[264] Ahmed R, Turja MA, Sahneh FD, Ghosh M, Hamm K, Kobourov S. Computing Steiner trees using graph neural networks. 2021. arXiv:2108.08368.

[265] Chen W, Yao X, Zhang X, Yu B. Efficient deep space filling curve. In: Proceedings of 2023 IEEE/CVF International Conference on Computer Vision; 2023 Oct 1–6; Paris, France. Piscataway: IEEE; 2023. p. 17525–34.

[266] Abboud R, Dimitrov R, Ceylan II. Shortest path networks for graph property prediction. In: Chien I, Chen J, Zhao L, editors. Proceedings of the 1st Learning on Graphs Conference; 2022 Dec 9–12; online conference. New York City: PMLR; 2022. p. 5.

Journal Pre-proofs

[267] Chen H, Luo Z, Zhang J, Zhou L, Bai X, Hu Z, et al. Learning to match features with seeded graph matching network. In: Proceedings of 2021 IEEE/CVF International Conference on Computer Vision; 2021 Oct 11–17; Montréal, QC, Canada. Piscataway: IEEE; 2021. p. 6301–10.

[268] Huoh TL, Luo Y, Li P, Zhang T. Flow-based encrypted network traffic classification with graph neural networks. IEEE Trans Netw Serv Manag 2022;20(2):1224–37.

[269] Hope O, Yoneki E. GDDR: GNN-based data-driven routing. In: Proceedings of 2021 IEEE 41st International Conference on Distributed Computing Systems; 2021 Jul 7–10; Washington, DC, USA. Piscataway: IEEE; 2021. p. 517–27.

[270] Yang Y, Zou D, He X. Graph neural network-based node deployment for throughput enhancement. IEEE Trans Neural Netw Learn Syst 2023;35(10):14810–24.

[271] Eisen M, Ribeiro A. Optimal wireless resource allocation with random edge graph neural networks. IEEE Trans Signal Process 2020;68:2977–91.

[272] He Z, Wang L, Ye H, Li GY, Juang BHF. Resource allocation based on graph neural networks in vehicular communications. In: Proceedings of 2020 IEEE Global Communications Conference; 2020 Dec 7–11; Taipei, China. Piscataway: IEEE; 2020. p. 1–5.

[273] Zhang C, Song W, Cao Z, Zhang J, Tan PS, Chi X. Learning to dispatch for job shop scheduling via deep reinforcement learning. Adv Neural Inf Process Syst 2020;33:1621–32.

[274] Song W, Chen X, Li Q, Cao Z. Flexible job-shop scheduling via graph neural network and deep reinforcement learning. IEEE Trans Industr Inform 2023;19(2):1600–10.

[275] Sato R, Yamada M, Kashima H. Approximation ratios of graph neural networks for combinatorial problems. In: Wallach HM, Larochelle H, Beygelzimer A, d'Alché-Buc F, Fox EB, editors. Proceedings of the 33rd International Conference on Neural Information Processing Systems; 2019 Dec 8–14; Vancouver, BC, Canada. Red Hook: Curran Associates, Inc.; 2019. p. 4081–90.

[276] Wu Y, Song W, Cao Z, Zhang J, Gupta A, Lin M. Graph learning assisted multi-objective integer programming. Adv Neural Inf Process Syst 2022;35:17774–87.

[277] Wan X, Xu K, Liao X, Jin Y, Chen K, Jin X. Scalable and efficient full-graph GNN training for large graphs. Proc ACM Manag Data 2023;1(2):143.

[278] Kwon S, Choi H, Park S. Solving bilevel knapsack problem using graph neural networks. 2022. arXiv:2211.13436.

[279] Butti S, Dalmau V. The complexity of the distributed constraint satisfaction problem. Theory Comput Syst 2024;68(4):838–67.

[280] Liu Y, Zhang P, Gao Y, Zhou C, Li Z, Chen H. Combinatorial optimization with automated graph neural networks. 2024. arXiv:2406.02872.

[281] Chi C, Aboussalah A, Khalil EB, Wang J, Sherkat-Masoumi Z. A deep reinforcement learning framework for column generation. Adv Neural Inf Process Syst 2022;35:9633–44.

[282] Shafi Z, Miller BA, Eliassi-Rad T, Caceres RS. Graph-SCP: accelerating set cover problems with graph neural networks. 2023. arXiv:2310.07979.

[283] Yuan H, Jiang P, Song S. The neural-prediction based acceleration algorithm of column generation for graph-based set covering problems. In: Proceedings of 2022 IEEE International Conference on Systems, Man, and Cybernetics; 2022 Oct 9–12; Prague, Czech Republic. Piscataway: IEEE; 2022. p. 1115–20.

[284] Yolcu E, Póczos B. Learning local search heuristics for boolean satisfiability. In: Wallach HM, Larochelle H, Beygelzimer A, d'Alché-Buc F, Fox EB, editors. Proceedings of the 33rd International Conference on Neural Information Processing Systems; 2019 Dec 8–14; Vancouver, BC, Canada. Red Hook: Curran Associates, Inc.; 2019. p. 7992–8003.

[285] Bünz B, Lamm M. Graph neural networks and boolean satisfiability. 2017. arXiv:1702.03592.

[286] Yan Z, Li M, Shi Z, Zhang W, Chen Y, Zhang H. Addressing variable dependency in GNN-based SAT solving. 2023. arXiv:2304.08738.

[287] Goh YL, Cao Z, Ma Y, Dong Y, Dupty MH, Lee WS. Hierarchical neural constructive solver for real-world TSP scenarios. In: Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining; 2024 Aug 25–29; Los Angeles, CA, USA. New York City: Association for Computing Machinery; 2024. p. 884–95.

[288] Li M, Tu S, Xu L. Generalizing graph network models for the traveling salesman problem with Lin–Kernighan–Helsgaun heuristics. In: Luo B, Cheng L, Wu ZG, Li H, Li C, editors. Neural information processing. ICONIP 2023. Singapore: Springer; 2024. p. 528–39.

Journal Pre-proofs

[289] Chen T, Zhang X, You M, Zheng G, Lambotharan S. A GNN-based supervised learning framework for resource allocation in wireless IoT networks. IEEE Internet Things J 2022;9(3):1712–24.

[290] Cai C, Wang D, Wang Y. Graph coarsening with neural networks. 2021. arXiv:2102.01350.

[291] Awasthi P, Das A, Gollapudi S. Beyond GNNs: an efficient architecture for graph problems. In: Proceedings of the AAAI Conference on Artificial Intelligence; 2022 Feb 22–March 1; Vancouver, BC, Canada. Palo Alto: AAAI Press; 2022. p. 6019–27.

[292] Fan M, Wu Y, Cao Z, Song W, Sartoretti G, Liu H, et al. Conditional neural heuristic for multiobjective vehicle routing problems. IEEE Trans Neural Netw Learn Syst 2025;36(3):4677–89.

[293] Lin X, Yang Z, Zhang Q. Pareto set learning for neural multi-objective combinatorial optimization. In: Proceedings of the 10th International Conference on Learning Representations; 2022 Apr 25–29; online conference; 2022.

[294] Boulesnane A. Evolutionary dynamic optimization and machine learning. In: Valadi J, Singh KP, Ojha M, Siarry P, editors. Advanced machine learning with evolutionary and metaheuristic techniques. Singapore: Springer; 2024. p. 67–85.

[295] Lu H, Li Z, Wang R, Ren Q, Li X, Yuan M, et al. ROCO: a general framework for evaluating robustness of combinatorial optimization solvers on graphs. In: Proceedings of the 11th International Conference on Learning Representations; 2023 May 1–5; Kigali, Rwanda; 2023.

[296] Liu F, Tong X, Yuan M, Lin X, Luo F, Wang Z, et al. Evolution of heuristics: towards efficient automatic algorithm design using large language model. In: Salakhutdinov R, Kolter Z, Heller K, Weller A, Oliver N, Scarlett J, et al., editors. Proceedings of the 41st International Conference on Machine Learning; 2024 Jul 21–27; Vienna, Austria. New York City: PMLR; 2024. p. 32201–23.

[297] Ju W, Fang Z, Gu Y, Liu Z, Long Q, Qiao Z, et al. A comprehensive survey on deep graph representation learning. Neural Netw 2024;173:106207.

[298] Yan X, Zhong G, Jin Y, Ke X, Xie F, Huang G. Binary spectral clustering for multi-view data. Inf Sci 2024;677:120899.

[299] Yan X, Wang Z, Jin Y. Heterogeneity-aware federated graph neural networks for incomplete multi-view clustering. IEEE Trans Em Top Comput Intell 2026;10(1):369–81.

[300] Qiu H, Hancock ER. Graph simplification and matching using commute times. Pattern Recognit 2007;40(10):2874–89.

[301] Zhang T, Shan HR, Little MA. Causal GraphSAGE: a robust graph method for classification based on causal sampling. Pattern Recognit 2022;128:108696.

[302] Chen J, Ma T, Xiao C. FastGCN: fast learning with graph convolutional networks via importance sampling. 2018. arXiv:1801.10247.

[303] Bai Y, Li C, Lin Z, Wu Y, Miao Y, Liu Y, et al. Efficient data loader for fast sampling-based GNN training on large graphs. IEEE Trans Parallel Distrib Syst 2021;32(10):2541–56.

[304] Besta M, Hoefler T. Parallel and distributed graph neural networks: an in-depth concurrency analysis. IEEE Trans Pattern Anal Mach Intell 2024;46(5):2584–606.

[305] Liu S, Yan X, Jin Y. An edge-aware graph autoencoder trained on scale-imbalanced data for traveling salesman problems. Knowl Base Syst 2024;291:111559.

[306] Heydaribeni N, Zhan X, Zhang R, Eliassi-Rad T, Koushanfar F. Distributed constrained combinatorial optimization leveraging hypergraph neural networks. Nat Mach Intell 2024;6(6):664–72.

[307] Yan X, Huang H, Jin Y, Chen L, Liang Z, Hao Z. Neural architecture search via multi-hashing embedding and graph tensor networks for multilingual text classification. IEEE Trans Em Top Comput Intell 2023;8(1):350–63.

[308] Wang Z, Yan X, Jin Y. Fairness-oriented vertical federated GNNs with incomplete sensitive attributes. Inf Fusion 2026;127(Pt B):103871.

[309] Zhu H, Wang X, Jin Y. Federated many-task Bayesian optimization. IEEE Trans Evol Comput 2024;28(4):980–93.

Journal Pre-proofs

Declaration of Interest Statement

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐ The author is an Editorial Board Member/Editor-in-Chief/Associate Editor/Guest Editor for this journal and was not involved in the editorial review or the decision to publish this article.

☐ The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Journal Pre-proofs