IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 19 March 2023, accepted 14 April 2023, date of publication 18 April 2023, date of current version 24 April 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3268212

RESEARCH ARTICLE

Applying Graph Neural Networks to the Decision Version of Graph Combinatorial Optimization Problems

RAKA JOVANOVIC[1], MICHAEL PALK[2], SERTAC BAYHAN[1], (Senior Member, IEEE), AND STEFAN VOSS[2]

[1]Qatar Environment and Energy Research Institute, Hamad Bin Khalifa University, Doha, Qatar
[2]Institute of Information Systems, University of Hamburg, 20146 Hamburg, Germany

Corresponding author: Raka Jovanovic (rjovanovic@hbku.edu.qa)

Open Access funding provided by the Qatar National Library.

ABSTRACT In recent years, there has been a significant increase in the application of graph neural networks on a wide range of different problems. A specially promising direction of research is on graph convolutional neural networks (GCN). This work focuses on the application of GCNs to graph combinatorial optimization problems (GCOPs), specifically on their decision versions. GCNs are applied on the family of GCOPs in which the objective function is directly related to the number of vertices in a graph. The selected problems are the minimum vertex cover, maximum clique, minimum dominating set and graph coloring. In this work, a framework is proposed for solving problems of this type. The performance of this approach is explored for standard multi-layer GCNs using different types of convolutional layers. On top of this, we propose a new structure that is based on graph isomorphism networks. Computational experiments show that this new structure is significantly more efficient than basic structures. To further improve the performance of this method, the use of GCN ensembles is also explored. When using GCNs to generate solution values for GCOPs, they often correspond to non-feasible solutions because they violate the problem boundaries. This issue is addressed by defining an asymmetric loss function that is used during the GCN training. The proposed method is evaluated on a large set of training data consisting of graph solution pairs that can also be accessed online.

INDEX TERMS Combinatorial optimization, NP-hard problem, graph neural networks, loss function, deep learning.

I. INTRODUCTION

In recent years there has been an extensive research on the topic of artificial neural networks (ANN). They have managed to outperform many of the standard methods for a wide range of practical problems [1], [2], [3], [4], [5]. One of the most significant advances in this field has come from the use of convolutional neural networks (CNNs) and the availability of enormous data sets used for their training [6].

One of the issues with the use of ANNs and, consequently, CNNs is that they, in the general case, are most effectively applied on grid structured data (single or multidimensional). To be more precise, a CNN is applied to some data in the form of vectors, matrices or tensors, over which the convolutional filter can "scan" through; exploiting the grid or ordered structure. A typical example would be a convolutional filter that detects edges in an image, which can be represented by a matrix or tensor. For many real-world problems, the corresponding mathematical models have a more complex structure that cannot be well presented using n-dimensional arrays or similar structures. In these models, the relation between data elements has a graph structure without any specific ordering of the elements. To be more precise, there can be arbitrary relations (connections) between data elements. Such complex structures frequently occur in problems related to logistics, communication systems, social networks

The associate editor coordinating the review of this manuscript and approving it for publication was Diego Oliva.

and many others. For example, in a social network, the graph consists of nodes representing people and edges representing relationships between them. The relationships could be friendships, family connections, or work relationships, among others. In this graph, the order of the nodes is not important because the relationships between people are not dependent on the order in which they are added to the graph. Thus, even a rearrangement of the nodes would preserve the graph structure. In contrast, switching pixel elements of an image would completely change the original structure, as the matrix or tensor representing the image would be modified.

This problem was addressed by generalizing ANNs into graph neural networks (GNNs), for which a comprehensive review can be found in [7] and [8]. The use of GNNs has proven its effectiveness on a wide range of practical problems; some examples are social recommendations [9], [10], text classification [11], physics [12], traffic [13], [14], estimation of arrival time [15], power systems [16], etc. The application of GNNs to practical problems frequently corresponds to some type of graph classification, i.e. molecular property prediction [17]. Another type of problem is the classification of the graph's vertices or edges, i.e. detecting fraudulent entities (corresponding to vertices of a graph). A very promising direction of GNN research is the use of graph convolutional networks (GCNs), for which a recent review can be seen in [18]. There is a high level of similarities between GCNs and CNNs, but there is a significant difference in the method for message passing between elements and the network structures. Except for the original GCN structure [19], several other ones have been developed, i.e.: GraphSAGE [20], graph attention network (GAT) [21] graph isomorphism network (GIN) [22], topology adaptive graph convolutional networks (TAGConv) [23], and many others.

ANNs have been applied for finding approximate solutions for a variety of combinatorial optimization problems; recent reviews can be found in [24] and [25]. Similar to this, GNNs have been extensively researched for solving graph combinatorial optimization problems (GCOP). References [26], [27]. For instance, GNNs have been applied to the traveling salesman problem using supervised learning [28], [29], [30]. The minimum vertex cover problem (MVCP) has been solved using GCNs as a part of more complex algorithms [31], [32]. GNNs have also been used for finding high quality heuristics for the weighted version of the MVCP [33]. The decision version of the graph coloring problem (GCP) has been solved using a specialized GCN-based method [34]. A GNN has been combined with a Q-learning-based method and applied to the max cut problem and the MVCP [35]. Other GCOPs, that have been solved using GCNs, are maximum independent set [32], the maximum clique problem (MCP) [32], [35], and others.

Most of these methods focus on using GNNs as a part of more complex algorithms to find solutions. One approach is using a trained GNN instead of a heuristic in a constructive or similar method when solving a problem of interest [33]. Another direction is using the probabilities, provided by the GNN, that a node or edge is a part of the solution to guide a tree search [32] or within a Q-learning method [35]. This type of approach is often part of a metaheuristic method that itself has a good performance. Although the use of GNNs has proven effective for designing heuristics for GCOPs, it has been shown that there are limitations for specific groups of problems [36]. It is important to point out that the message passing system in GNNs, specially GCNs, has limits on the graph properties it can learn [37].

For a wide range of problems, ANNs have been applied more directly, in the sense that for a set of input parameters the corresponding solutions are provided. In case of GCOPs, the direct application of GNNs for finding solutions is often complex and not very effective. In this paper, we explore the use of GNNs, more precisely GCNs for GCOPs. To be more precise, the proposed use of GCNs solves a somewhat simpler version related to the decision version of the problem. In it, the exact solution is not found but its value. As an example, consider the number of nodes in a minimum vertex cover based on the input graph. It should be noted that for many real-world problems modeled using graphs, the specific solution is not important but only its value. An example would be deciding if it is possible to deliver goods to a group of customers with an electric vehicle with a given range, modeled using the decision version of the traveling salesman problem. In the proposed approach, the input to the problem is the graph and the output is the approximation to the value of the optimal solutions. The method is applied for several standard GCOPs, with a focus on a subset of GCOPs, where the solution is closely related to the total number of nodes in the graph. In this work, a general framework is provided on how GNNs, more specifically GCNs, could be applied to problems of this type. In the conducted computational experiments, a comparison is done on several commonly used GCNs with different properties. In addition, a more complex GCN structure based on GINs is provided. The computational experiments show that this GCN structure is more effective than the basic ones. The performance of the GIN-based GCN structure is further enhanced by the use of GCN ensembles. One of the issues with using GCN for generating solution values for GCOPs is that they often correspond to non-feasible solutions in the sense that they violate the bounds of the problem. This problem is addressed by defining an asymmetric loss function that is used during the GCN training. A large training/test data set is used, generated for the computational experiments consisting of graph solution pairs which are also made available online [38].

The main contribution of this work is proposing a general framework for using GCNs to approximate the solution values of decision versions of GCOPs which is highly independent of the specific problem being solved. One of the main issues in developing this framework is addressing different constraints of GCOPs without losing the generality of the framework. To address this problem, which corresponds to non-feasible solutions, a parameterized asymmetric loss function for GCN training is introduced. The use of this loss

function can be used to control the probability of generating solution values that correspond to non-feasible solutions.

A. PAPER STRUCTURE

The paper is organized as follows. Section II provides the definitions of the set of GCOPs that are solved using GCNs. The next section gives the details on the use of GCNs for solving the decision version of these GCOPs. This section is divided into several parts providing information on the data use, GCN structure, loss function and ensembles. Section IV provides details of the conducted computational experiments and the analysis of the results. In Section V, the paper is finalized with some concluding remarks.

II. GRAPH COMBINATORIAL OPTIMIZATION PROBLEMS

In this section, the set of GCOPs used for evaluating the proposed GCN approach are presented. In this work, the focus is on problems defined on a graph $G(V, E)$ with a set of vertices V and a set of edges E. The selected problems have an objective function whose value is directly related to the cardinality $|V|$ of the vertex set V. To be more precise, the objective $O(G)$, for graph G, has an integer value between 1 and $|V|$. The goal of the proposed work is to develop a GCN-based method that is highly independent of the specific GCOP being solved. The trained GCN should receive as input a problem instance (graph G) and provide as output, an approximation to the solution value of the underlying GCOP.

We focus on problems which are defined on undirected simple graphs without vertex or edge weights. Although GCN have been applied to problems of edge classification or objectives related to edge properties, in their basic form the message passing is focused on vertices of the graph. To avoid additional complexity, problems whose objective value is related to the edge set are avoided.

The proposed approach is tested on several graph problems, namely the MVCP, the MCP, the minimum dominating set problem (MDSP) and the GCP, which were chosen to represent a diverse range of graph problems. In the case of the MVCP and MDSP, the emphasis is on the constraints related to the adjacency or incidence between a group of nodes and the remaining nodes or edges in the graph. For the MCP, the constraints are centered on the adjacency between nodes in a subset of the vertex set and the relationship between all such subsets in the graph. Finally, concerning the GCP, although the constraints are local, the objective is a highly global property that cannot be easily solved by evaluating subgraphs of the original graph. Although the selected problems are well known, for the sake of completeness, their definitions and corresponding integer programming models (IP)s are provided.

A. MINIMUM VERTEX COVER PROBLEM

The first GCOP that is used is the MVCP. A vertex cover C is a subset of the set of vertices V that has the property that each edge $uv \in E$ is incident to at least one vertex in $v \in C$. The goal is to find the vertex cover C having minimal cardinality $|C|$. The MVCP can be represented as an IP with a set of binary decision variables x_v for $v \in V$. The value of $x_v = 1$ indicates that $v \in C$. The corresponding IP is given as follows.

$$\text{Minimize} \sum_{v \in V} x_v \tag{1}$$

$$x_u + x_v \geq 1 \; uv \in E \tag{2}$$

$$x_v \in \{0, 1\} \; v \in V \tag{3}$$

The goal (see (1)) is to minimize the number of vertices in the vertex cover. The constraints (2) guarantee that for each edge $uv \in E$ at least one of its vertices is in the vertex cover.

B. MAXIMUM CLIQUE PROBLEM

The second chosen GCOP is the MCP. A subset C of the set of vertices V is a clique if all vertices in C are adjacent to each other. The goal is to find the clique C having maximal cardinality. The IP for the MCP uses a set of binary decision variables x_v for $v \in V$. The value of $x_v = 1$ indicates that $v \in C$. The corresponding IP is given as follows.

$$\text{Maximize} \sum_{v \in V} x_v \tag{4}$$

$$x_u + x_v \leq 1 \; uv \notin E \tag{5}$$

$$x_v \in \{0, 1\} \; v \in V \tag{6}$$

Eq. (4) states that the goal is to maximize the number of vertices in the clique. The constraints (5) guarantee that there are no two vertices $u, v \in C$ which do not have a connecting edge.

C. MINIMUM DOMINATING SET PROBLEM

The third GCOP of interest is the MDSP. A subset C of the set of vertices V is a dominating set if all the vertices in V are either an element of C or are adjacent to at least one vertex in C. The goal is to find the dominating set C having minimal cardinality. The IP for the MDSP uses a set of binary decision variables x_v for $v \in V$. The value of $x_v = 1$ indicates that $v \in C$. The corresponding IP is given as follows.

$$\text{Minimize} \sum_{v \in V} x_v \tag{7}$$

$$\sum_{v \in N[u]} x_v \geq 1 \; u \in V \tag{8}$$

$$x_v \in \{0, 1\} \; v \in V \tag{9}$$

According to (7) the goal is to minimize the number of vertices in the dominating set. Constraint (8) uses the notation $N[u]$ for the closed neighborhood (contains u) of node u. The constraints in this equation guarantee that for each vertex $u \in V$ at least one of the vertices in its closed neighborhood are in the dominating set.

D. GRAPH COLORING PROBLEM

The last problem that is addressed is the GCP, more precisely a vertex coloring. In this problem, the objective is to label all

the vertices with a specific ''color''. The labeling must satisfy the constraint that no two adjacent vertices are of the same color. The goal is to find the minimal number of colors to achieve this. There is a wide range of IPs for the GCP, which could be found in [39]. In this work, the representative model, proposed in [40] and [41], is used. This model introduces binary variables x_{uv} for each non-adjacent pair of vertices $u, v \in V$ which is equal to 1 if and only if the color of v is represented by u. Additional binary variables x_{uu} are defined to indicate if u is the representative of its color class. In the IP, $D = \{(u, v) \mid (u, v) \notin E \vee u = v\}$ is used for the set of all decision variable indices. Next, the notation $\bar{N}(u)$ is used for the set of vertices that are not adjacent to node u. Using these definitions, the IP has the following form.

$$\text{Minimize} \quad \sum_{u \in V} x_{uu} \tag{10}$$

$$\sum_{u \in \bar{N}(v) \cup v} x_{uv} \geq 1 \quad v \in V \tag{11}$$

$$x_{uv} + x_{uw} \leq x_{uu} \; u \in V, vw \; \in \mathcal{G}[\bar{N}(u)] \tag{12}$$

$$x_{uv} \in \{0, 1\} \; (u, v) \in D \tag{13}$$

The goal is to minimize the number of vertices that are color representatives; see (10). Inequalities Eq. (11) state that for any vertex $v \in V$, there must exist a color representative which can be v itself or is a non-adjacent node $u \in \bar{N}(v)$. The constraints given in (12) guarantee that neighboring vertices cannot have the same representative. Note that the notation $\mathcal{G}[A]$ is used for the subgraph of G containing all the vertices in vertex set A and corresponding edges.

III. METHOD OVERVIEW

In this section, the basic components of a method that uses GCNs for solving the decision version of GCOPs are presented. Firstly, the standard message passing systems of GCNs are presented. Next, the structure of the proposed GCNs for this type of problem are provided using the standard GCN layers. After that, the method for assigning the values of node features based on the input graph and the problem of interest are provided. In addition, the method for normalizing the objective values for a specific problem for different graphs is given. The last steps are introducing the used loss functions and ensembles.

A. GRAPH CONVOLUTIONAL NETWORKS

In this subsection, we provide an overview of the application of GNN on GCOPs. More precisely on the application of GCNs. The basic idea of a GCN is to have smart message passing between neighboring vertices. To be more precise, the method for aggregating information (vertex features) from a vertex and its neighbors. The message passing is performed using convolutional layers and, in the general case, several such layers are stacked one after another, see Fig. 1. It is assumed that each node u has an associated feature vector $h_u^{(l)} \in \mathbb{R}^m$, where m is the number of features, the same for all $u \in V$, at layer l, for $l = 0..L$, where L is the number of layers of the GCN. The standard graph convolutional layer-wise propagation rule is given in the following equation.

$$h_u^{(l+1)} = \sigma\left(b^{(l)} + \sum_{v \in \mathcal{N}[u]} \frac{1}{c_{uv}} h_v^{(l)} W^{(l)}\right) \tag{14}$$

Eq. (14) provides the information on how the feature vector $h_u^{(l)}$ of a node u in layer l, is transformed to its feature vector $h_u^{(l+1)}$ in layer $l+1$. In Eq. (14), σ represents an activation function, such as the ReLU$(\cdot)$ [42]. $W^{(l)}$ is the layer-specific trainable weight matrix and $b^{(l)}$ is a layer-specific trainable bias vector. As before, the notation $N[u]$ is used for the closed neighborhood of node u. c_{uv} is a normalization factor (i.e. $c_{uv} = \sqrt{|\mathcal{N}(u)|}\sqrt{|\mathcal{N}(v)|}$).

There have been several ways to improve the propagation rule given in Eq. (14) using more advanced approaches like aggregation and dropout rates. The idea of aggregation is that in addition to using the features of a neighboring node $v \in N(u)$ is to also use aggregated data about its neighbors. One of the most successful approaches of this type is the use of the SAGE convolutional layer, for more details see [20]. Its layer-wise propagation rule is given in the following equations

$$h_{\mathcal{N}(u)}^{(l+1)} = \text{aggregate}\left(\{h_v^l, \forall v \in \mathcal{N}(u)\}\right) \tag{15}$$

$$h_u^{(l+1)} = \sigma\left(b^{(l)} + W^{(l)} \cdot \text{concat}(h_u^{(l)}, h_{\mathcal{N}(u)}^{(l+1)})\right) \tag{16}$$

$$h_u^{(l+1)} = \text{normalize}\left(h_u^{(l+1)}\right) \tag{17}$$

The first part of the propagation rule is the aggregation given in Eq. (15). In it, the aggregated information for the neighboring nodes is stored in a vector $h_{\mathcal{N}(u)}^{(l+1)}$. The typical aggregator functions are *pooling*, *mean*, and *lstm*. Eq. (16) is similar to the one in the case of the basic *GCN*, given in Eq. (14), except that it uses the concatenation of vector $h_u^{(l)}$ and the aggregated values of the features of neighboring nodes of u instead of the features of these nodes. Finally, Eq. (17) includes normalization of the output vector $h_u^{(l+1)}$. Note also that the SAGE convolution can include dropout rates in the same way as standard CNNs.

By applying a series of trained convolutional layers L-times like in Eq. (14) or in Eq. (15)-(17), the initial feature vectors h_u^0 of all nodes u are transformed to h_u^L in the last layer. These transformed feature vectors are aggregated via a readout function like *max*, *min*, *sum* and *mean*, to provide a single resulting value for the corresponding input graph. In some applications, it is possible to enhance the learning capability of the GCN by applying additional transformations (i.e. by using a CNN) to the readout. A graphical illustration of a general GCN that can be used for GCOPs can be seen in Fig. 1.

B. METHOD STRUCTURE

The goal of this work is to evaluate the use of different structured GCNs on GCOPs. Therefore, two types of multi-layer structures are considered. The first one is the standard simple

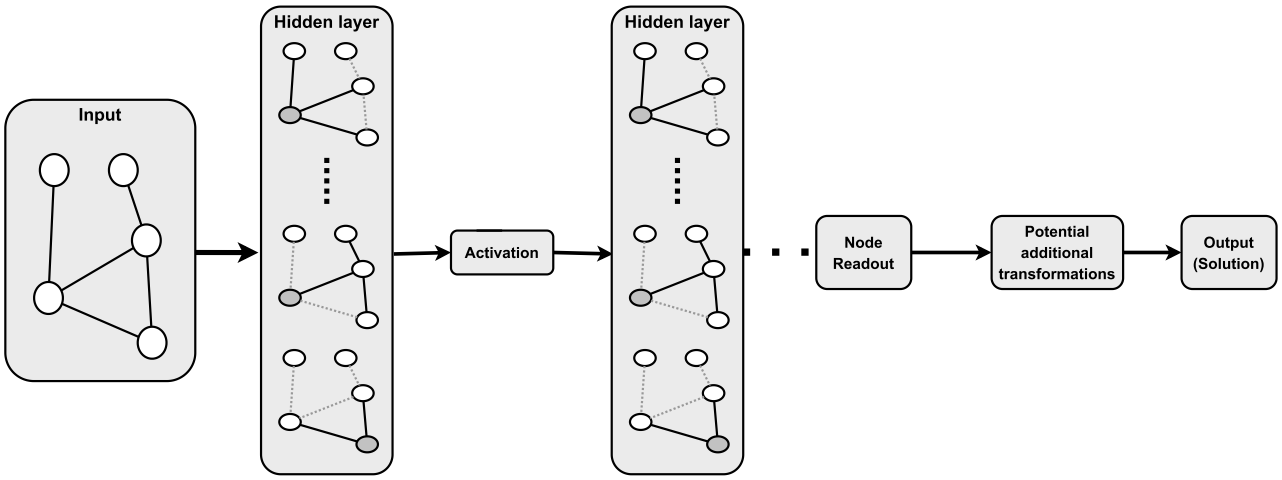


FIGURE 1. Illustration of a standard GCN for calculating global graph properties.

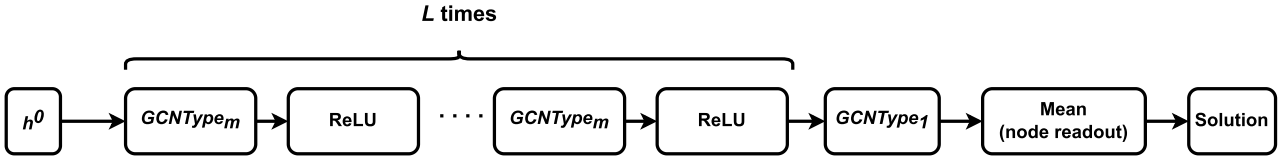


FIGURE 2. Illustration of the basic GCN structure. It consists of L layers of type *GCNType* with the output having m values.

multi-layer structure. The second one, proposed in this paper, is based on the GIN approach. In the further text, details of both structures are given.

In the simple approach, layers are stacked one after another. The input for the GCN are input feature vectors $h_u^{(0)}$ of nodes u. The next part are L layers having one of the layer-wise propagation rules given in Eq. (14) or Eqs. (15)-(17). In the further text, the notation *GCNType* is used for a convolutional layer that is one of these types. All these layers use *ReLU* as the activation function and the dimension of the output feature vector $h_u^{(l+1)}$ is m. After L such layers, an additional one, $GCNType_1$, is added that has an output feature vector with dimension equal to one. Finally, the used node readout function is *mean*, which returns the average value of all the nodes' feature vectors in the last layer, resulting in a single solution value for the whole graph. A graphical illustration of this multi-layer GCN structure can be seen in Fig. 2. It should be noted that the use of the *mean* node readout function has been selected empirically, based on our computational experiments, since it produced the best results.

The second structure uses the concepts proposed in the GIN paper by Xu et al. [22]. This structure is a modification of the simple GCN architecture, where the outputs of the layers in the multi-layer GCN are reused. The proposed structure is similar to the simple one in that it uses L stacked layers with an output vector of dimension m. However, the key difference is that it concatenates the output features $h^{(1)}$ to $h^{(L)}$ of the stacked layers into a single vector. This concatenated vector is then used as an input for a multi-layer perceptron (MLP) network. The MLP network consists of two fully connected layers. The first layer is a fully connected layer with an output size of m and a ReLU activation function. The second layer is a fully connected layer with an output size of 1. Finally, the *mean* node readout function is performed to collect global information from all nodes. This function takes the output of the MLP network and computes the mean of the output over all nodes in the graph. The resulting output is a scalar that summarizes the graph-level information.

In summary, the proposed structure uses a modified GCN architecture that concatenates the outputs of the stacked layers and feeds them into an MLP network. The resulting output is then summarized using the mean node readout function to capture global information from the graph. A graphical illustration of this multi-layer GIN-based structure can be seen in Fig. 3.

C. TRAINING SETUP

The training set up is presented in this subsection. It is divided into two parts. Part one discusses how training data is used, and part two discusses loss functions.

A training procedure must be designed so that knowledge can be effectively generalized to graphs of different sizes. In the case of applying GCNs to standard graph classification problems, this is not an issue since the classification is done to a fixed number of types [7], [8]. In the case of the problems of interest, although the value of the objective function is discrete, there is a very large number of potential classes. On the other hand, it is evident that such fixed classes would not be

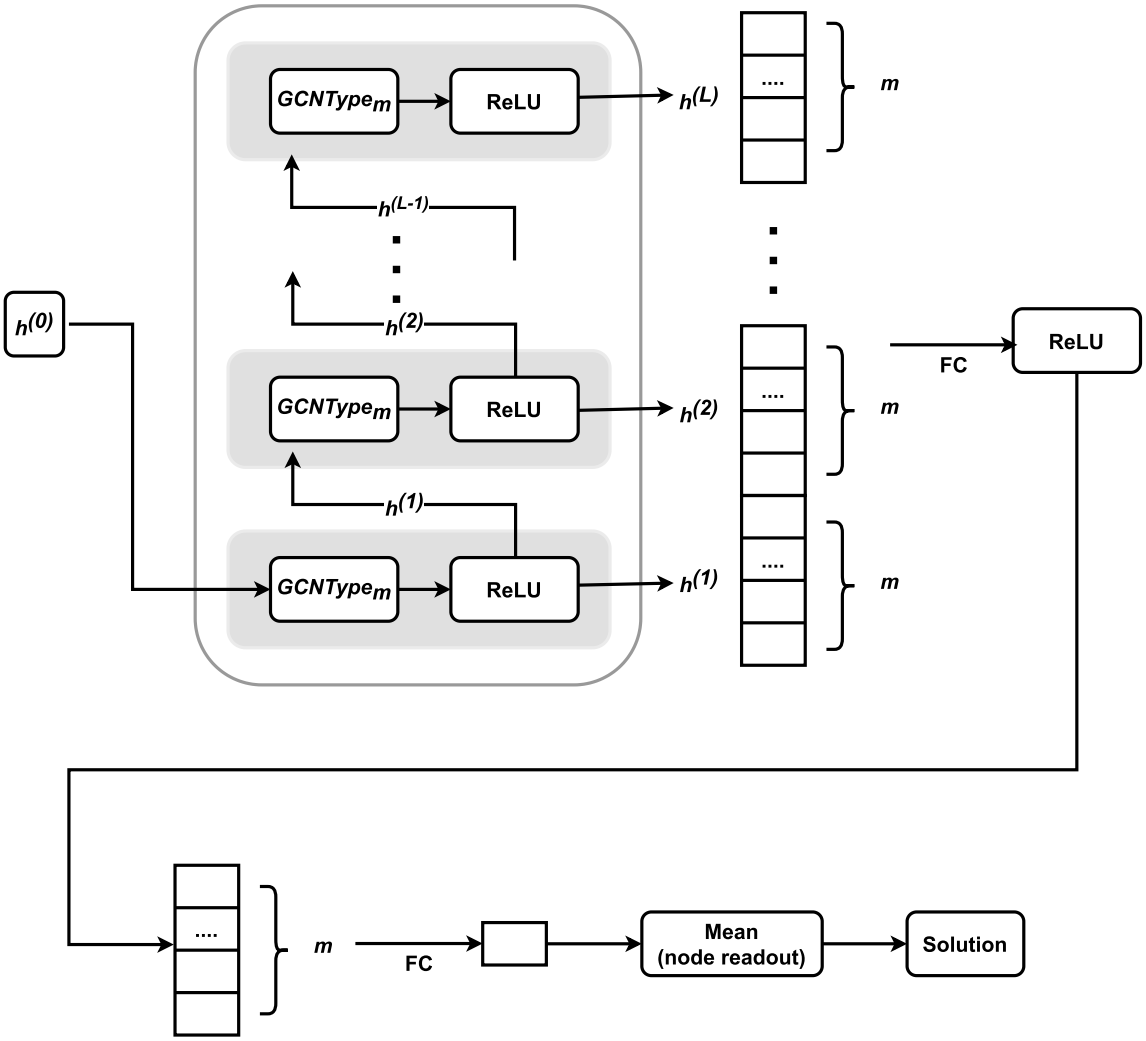


FIGURE 3. Illustration of the basic GIN-based GCN structure. It consists of *L* layers of type *GCNType* with the output having *m* values. The output of each layer is concatenated to an *nm*-sized vector. The abbreviation *FC* is used for fully connected layer.

suitable for the generalization of knowledge. For instance, a dominating set of n nodes for a graph with $|V| = 2n$ nodes has completely different properties than for a graph having $|V| = n^2$ nodes. In the case of the former, each node in the dominating set covers a small number of nodes, while in the latter it covers a large number of them.

To avoid these types of issues, the model uses normalization. As a matter of fact, the normalization is performed on both the initial features of nodes and the objective function In both cases, the normalization is performed based on the size $|V|$ of the vertex set V. This is done using the following equations.

$$h_u^{(0)} = \frac{1}{|V|} \tag{18}$$

$$O_n(G) = \frac{O(G)}{|V|} \tag{19}$$

Eq. (18) states that the value of the initial feature $h_u^{(0)}$ is equal to one over $|V|$ for all nodes u of the graph that is being evaluated. The idea is that the initial feature provides some information about the size of the graph. Eq. (19) states that the normalization of the objective is also done using the number of nodes in the graph. Note that the selected GCOPs have an objective function $O(G)$ whose value is between 1 and $|V|$, both values included. A potential natural alternative to the use of normalized values is the use of the original ones in combination with the *sum* node readout function. This type of approach is mathematically highly similar and should preserve the same level of information. In our initial test this combination had a very poor convergence and consequently has not been used.

In the training, two loss functions are considered. Firstly, this is the standard mean square error (MSE).

$$MSE(o, \hat{o}) = \frac{1}{n} \sum_{i=1}^{n} (o_i - \hat{o}_i)^2 \tag{20}$$

In Eq. (20), o are the normalized training values (optimal solution values) corresponding to input graphs G and $\hat{o}$ is the estimate acquired by applying the GNN, $GCN(G)$. The issue with the MSE loss function is that it does not consider the feasibility of having a solution with the corresponding value. To avoid this, the following asymmetric loss function, for the case of minimization problems, is proposed.

$$Er_c(o, \hat{o}) = \begin{cases} (o - \hat{o})^2 & o \leq \hat{o} \\ c(o - \hat{o})^2 & o > \hat{o} \end{cases} \tag{21}$$

$$asMSE_c(o, \hat{o}) = \frac{1}{n} \sum_{i=1}^{n} Er_c(o_i, \hat{o}_i) \tag{22}$$

The loss function uses the error function given in Eq. (21). In the case the predicted value $\hat{o}$ is larger than the objective value the error is equal to the squared difference. In case the predicted value is lower, or in other words there is no corresponding feasible solution, the error is scaled using a coefficient $c > 1$. Let us use the term out-of-bounds solution values (OBSV) for solution values of this type. In this way, such non-feasible solutions are additionally ''penalized''. Finally, the loss function is equal to the average of all errors of this type. Note that in the case of a maximization problem the coefficient c is used for predictions that are greater than the value of the optimal solution, corresponding to OBSVs, so the conditions in Eq. (21) should be changed to $o \geq \hat{o}$ and $o < \hat{o}$.

D. ENSEMBLES

A common approach to improve the performance of ANNs is the use of ensembles; see, e.g. [43]. Ensembles use multiple ANNs jointly to solve a problem, this concept has also been successfully extended to GCNs [44], [45]. It has been shown that the generalization capabilities of such systems can outperform those of single networks. The use of ensembles is most effective if the component networks are as accurate and diverse as possible [43], [46]. The diversity of the used ANNs is generally achieved in two ways: by using different training sets and by having ANNs with different structures. Although there is a wide range of methods of combining the results of several ANNs when creating ensembles, even the use of simple averaging produces a high level of improvement [47]. In our implementation, this simple approach is used to enhance the performance of the method, based on the following equation

$$GCN_e(G) = \frac{1}{n} \sum_{i=1}^{n} GCN_i(G) \tag{23}$$

In Eq. (23), the notation GCN_i is used for a GCN acquired having a specific structure and training data. The estimate acquired using the ensemble GCN_e for graph G is equal to the average value of $GCN_i(G)$.

IV. NUMERICAL EXPERIMENTS

In this section, the results of the conducted computational experiments are presented. They have several objectives. The first is to evaluate the effectiveness of different structures of the GCN. The next is to assess the effect of the use of ensembles. The last objective is to compare the use of different loss functions for avoiding the generation of objective values that do not satisfy the bounds, or, in other words, for which there are no feasible solutions.

The proposed optimization methods have been implemented in Python in PyCharm 2022.2.3 (Community Edition). The GCNs have been implemented using Deep Graph Library (DGL) version 0.9.x and PyTorch version 1.9. The training data has been generated based on the IPs from Section II using CPLEX. The computational experiments have been performed on a PC running Windows 10 having an Intel(R)Xeon(R) Gold 6244 CPU @3.60 GHz processor with 128 GB memory.

A. TRAINING/TEST DATA

In this subsection, the method for generating the training/test data is presented. The training/test data consists of a large number of pairs, graph G and the value of an optimal solution. For each GCOP, a total of 105 000 such pairs have been generated. The procedure for generating the training/test pairs is the following. Firstly, graphs of varying sizes and densities have been generated. To be specific, the number of vertices in the graph is randomly selected from a range (N_{min}, N_{max}) and the density from a range $(\delta_{min}, \delta_{max})$, using a uniform distribution. For all the problems the following values are used: $\delta_{min} = 0.05$, $\delta_{max} = 0.5$ and $N_{min} = 15$. In the case of GCP, the maximal number of vertices has the value $N_{max} = 60$ and for the other problems $N_{max} = 150$. Note that the graphs in the generated training/test data have the following statistical properties. The mean value of the number of nodes is 37.5 and 82.5, with a standard deviation of 12.99 and 38.97, for the GCP instances and the other problems, respectively. The mean value for the graph density is 0.275 with a standard deviation of 0.13 for all the problems.

For each generated graph, the optimal solution is acquired using the presented IPs utilizing CPLEX. The size of the graphs that are used for training is constrained by the size of problem instances that could be solved by CPLEX within a reasonable time. The generated training data can be found at [38]. Note that this data is used in all the computational experiments that are presented in this section for training and testing.

B. TRAINING SETUP

The training setup is the following. The training of GCNs is stochastic in nature due to the use of the stochastic gradient descent. Because of this, for each of the tests, multiple GCNs are trained, which makes it possible to evaluate the method in more depth. To be more precise, 20 GCNs have been trained for each pair of selected structure and problem. Each of them has been trained using 5000 training pairs, distinct for each of the GCNs. The used test set is a separated set of 5000 graph and optimal solution pairs. Each of the layers has been implemented using the built-in methods of the DGL library. To be

specific, the DGL methods *GraphConv* and *SAGEConv* have been used for the basic and SAGE layers. The methods have been used with the default set of parameters, except for the ones related to the structure (number of layers and their size) of the proposed GCNs. In the case of *SAGEConv*, the drop rate is set to 0.5. In all the tests, the training is limited to 150 epochs. The size of a training batch is 50 and the value of the learning rate is 0.005. The value for the learning rate and batch size are selected based on the discussion provided in [48].

C. EVALUATING GCN STRUCTURES

The goal of the first set of experiments is to evaluate the performance of different GCN structures. The evaluation is done for the two structures using stacked layers, as in Fig. 2 with different layer-wise propagation rules. The notations *bGCN* and *SAGE* are used if the layer-wise propagation rule is specified using Eq. (14) and (15) - (17), respectively. The notation *sageGIN* is used for the GCNs corresponding to the structure given in Fig. 3. As it will be seen in the further text, the use of the *SAGE* layers had a better performance than the use of the basic GCN layer. Due to this fact, the proposed, more advanced structure *sageGIN*, uses *SAGE*-based convolutional layers.

The parameters specifying *bGCN*, *SAGE* and *sageGIN* are the size of each layer (m) and the number of such layers (L). As it is discussed in [49] for GCNs, in the general case, having a large number of layers is not beneficial due to message smoothing, although recently some research has been conducted on deep GCNs [50]. Note that this is in high contrast to CNNs, where a high number of convolutional layers is, generally, highly beneficial [51]. Because of this, the tested number of layers ranged from 2 to 4. The initially conducted computational experiments have empirically shown that a higher number of layers did not produce improvement in performance using the available hardware. The tested layer sizes are 16, 32 and 64.

A comparison of the three different GCN structures can be observed in Fig. 4. These figures provide aggregated information over the 20 GCNs generated for each GCN structure with a specified number of layers and layer sizes. Fig. 4 shows the average root mean squared relative error (RMSRE). The RMSRE corresponds to the following equation, which uses the same notation as Eq. (20) except that the original optimal solution values are used for o (not the normalized values) and the prediction $\hat{o}$ is equal to the GCN prediction scaled by $|V|$ and rounded to the closest integer.

$$RMSRE(o, \hat{o}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{o_i - \hat{o}_i}{o_i}\right)^2} \tag{24}$$

Note that the discussion in this subsection focuses on the general trends while the next subsection is dedicated to evaluating the quantitative performance. The first observation that can be made from these results is that *bGCN* has a significantly worse performance than *SAGE* and *sageGIN*. The advantage of *SAGE* compared to *bGCN* is most significant in case of using only two layers. The expected reason for this is that the information from more distant nodes is important for finding the objective values. In essence, *SAGE* considers nodes one level more distant than *bGCN* due to the use of aggregation. Another possible reason is that the *SAGE* layer uses a drop rate and *bGCN* does not. The advantage of *SAGE* becomes less significant in case of a higher number of layers. The more advanced *sageGIN* consistently outperforms the other two methods for all the tested combinations of number of layers and layer sizes. An increase in the layer size improved the performance for the same number of layers for *bGCN*, *SAGE* and *sageGIN* for all the tests.

Another observation that can be made for the best performing *sageGIN* is that an increase in layer size has a more significant effect on the performance of the method than the number of layers. The use of 3 layers instead of 2 provides a significant improvement. On the other hand, the further increase of layers to 4 does not result in a consistent improvement, which coincides with the conclusions in [49]. It is important to point out that, although the values of the RMSRE differ for the different GCOPs used for evaluation, the trends are consistent.

With the goal of providing additional insight for selecting the suitable structure of a GCN for application on a GCOP, the training time is also evaluated. The related information is depicted in Fig. 5. The first thing that the conducted computational experiments have shown, is that the training time is not dependent on the specific problem but only on the size of the graphs in the training set. Because of this, Fig. 5 only provides the average training time for GCP and MDSP, since MDSP used the same graph sizes as MVCP and MCP. It can be seen that *bGCN* has generally a lower training time than the other two methods, with the difference increasing with the number of layers and layer sizes. It needed approximately 50%-70% of the time of *SAGE* and *bGCN*. It can be observed that *SAGE* and *sageGIN* have similar training times, with *sageGIN* being slightly lower, even though *sageGIN* has a more complex structure. As expected, the increase in training time is directly related to the number and size of the used layers. In the case of a training set containing larger graphs (MDSP), there is a drastic increase in training time with the increase in layer size, which is nonlinear. For example, the training times for the layer sizes of 16, 32 and 64, for the MDSP in the case of *sageGIN* with 4 layers, were 5.2, 7.1 and 21.9 minutes, respectively. An increase in the number of layers has a less drastic effect but it is still highly significant. For the same GCOP and GCN structure, with 2, 3 and 4 layers with layer size of 64, the training times were 9.1, 15.1 and 21.9 minutes, respectively.

D. QUANTITATIVE EVALUATION

In this subsection, a quantitative evaluation of the use of GCNs for approximating the value of the optimal solution for GCOP is presented. The focus is on the *sageGIN* structure, since it significantly outperforms the two basic ones. In the

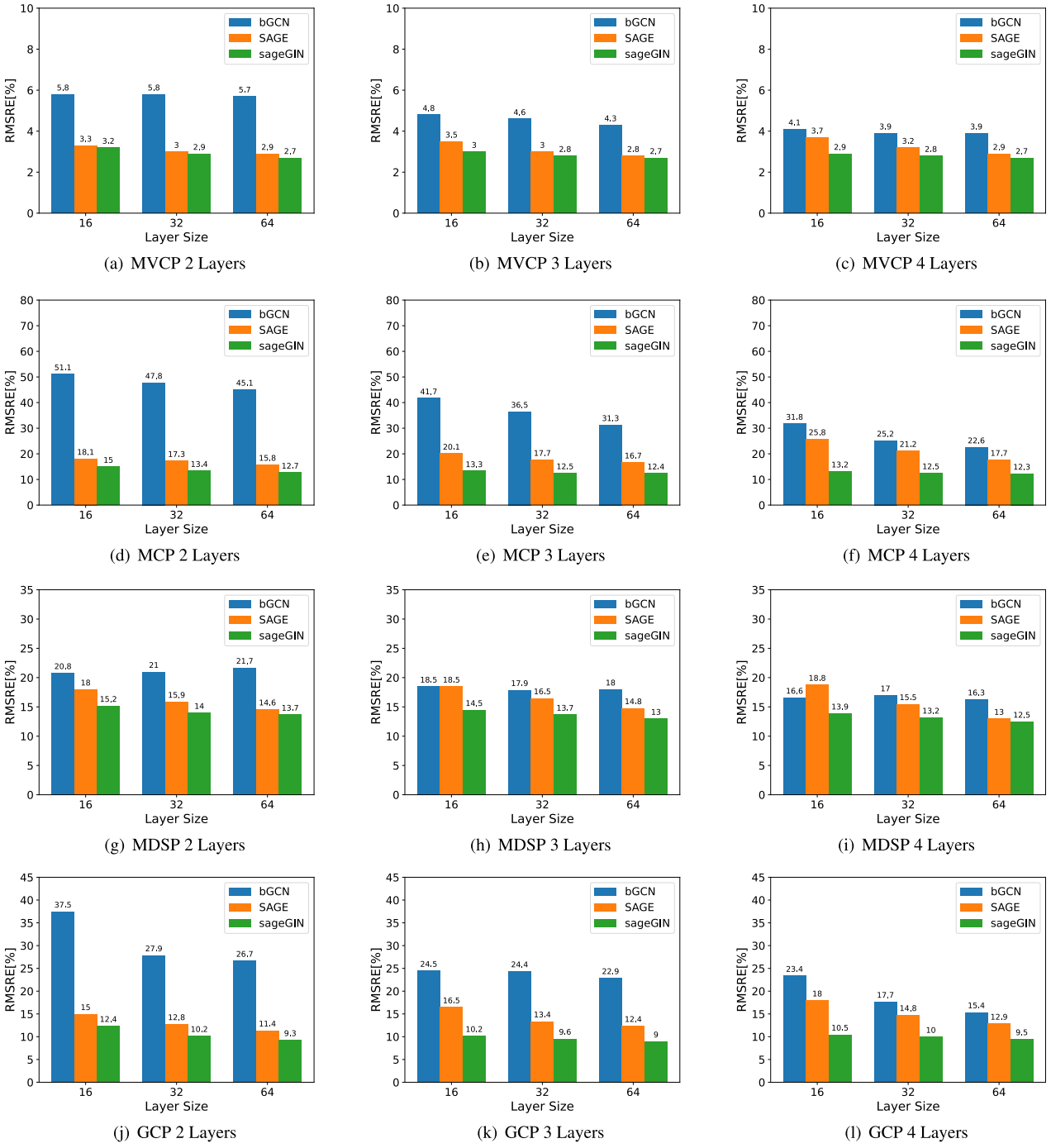


FIGURE 4. **Average RMSRE over 20 runs of different GCN structures. Each subfigure provides information for a different problem (MVCP, MCP, MDSP, and GCP) and number of layers. In each of the subfigures, the average RMSRE can be seen for different layer sizes and GCN structures.**

conducted experiments, the *sageGIN* has a layer size $m = 64$ and the number of used layers $L = 3$. The reason for selecting these values is that the computational experiments related to GCN structure, presented in the previous subsection, have shown that these values provide the best balance between training time and approximation quality.

The performance of the *sageGIN* structure is done for trained GCNs using different loss functions. The first used loss function is the MSE given in Eq. (20). In addition, the asymmetric loss function given in Eqs. (21), (22) is evaluated for parameter c having values 10 or 100.

The assessment is based on the average values over the 20 different trained GCNs for each problem type (MVCP, MCP, MDSP, and GCP) and selected loss function of the following values. Firstly, the root mean square error (RMSE) and RMSRE are used to provide information on the quality of

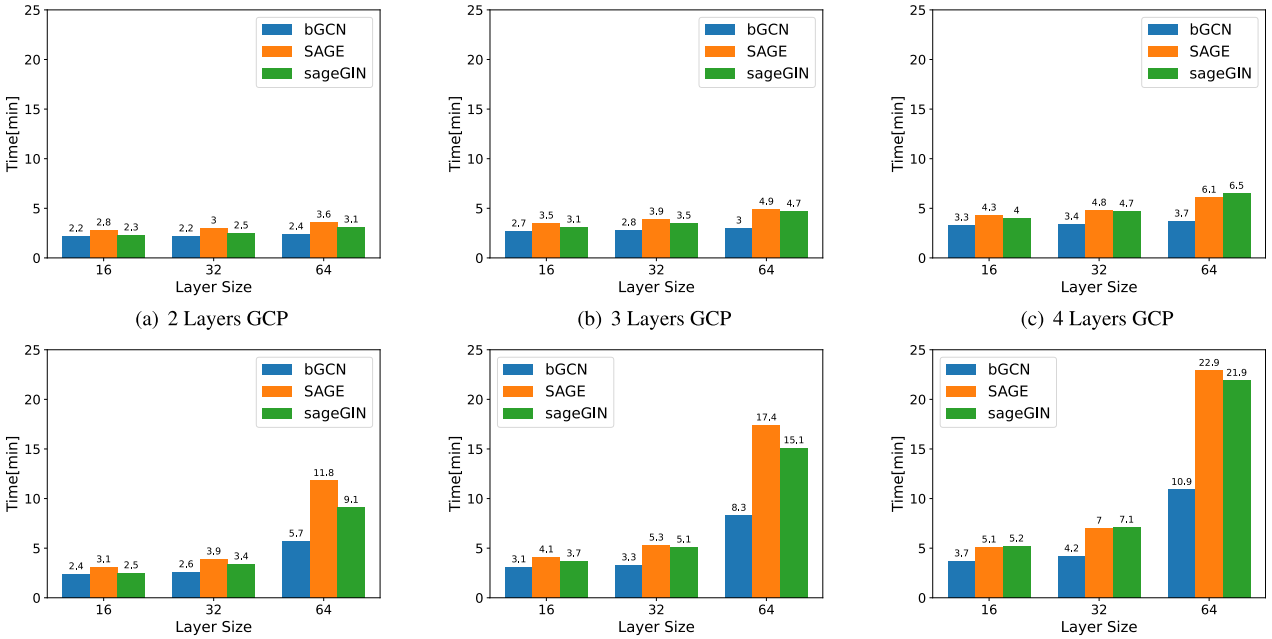


FIGURE 5. Average training times of different GCN structures. Each subfigure provides information for a different problem (GCP or MDSP) and number of layers. In each of the subfigures, the execution time can be seen for different layer sizes and GCN structures.

TABLE 1. Comparison of the average results over 20 trained *sageGIN* GCN structures with different loss functions (*MSE*, $asMSE_{10}$ and $asMSE_{100}$). In each run, the test set had 5000 graph solution pairs. The table also provides information on the use of ensembles. The presented aggregated values are RMSE and RMSRE, percentage of found optimal solution values and the percentage of solution values for which there are no feasible solutions (OBSV).

Loss function	RMSE		RMSRE [%]		Optimal [%]		OBSV [%]	
	Single	Ensemble	Single	Ensemble	Single	Ensemble	Single	Ensemble
Minimum Vertex Cover Problem								
MSE	1.06	0.86	2.78	2.54	39.88	49.10	32.23	28.70
$asMSE_{10}$	1.40	1.21	3.65	3.42	30.84	33.10	10.06	3.30
$asMSE_{100}$	2.61	2.49	6.08	5.97	10.21	3.20	1.46	0.20
Maximum Clique Problem								
MSE	0.60	0.51	12.39	11.59	67.18	74.40	16.98	13.00
$asMSE_{10}$	0.82	0.75	14.73	13.59	50.42	53.10	3.19	0.20
$asMSE_{100}$	1.09	1.07	19.68	19.51	30.05	24.60	0.36	0.10
Minimum Dominating Set Problem								
MSE	0.76	0.67	12.98	11.42	59.53	66.70	18.77	15.30
$asMSE_{10}$	1.09	1.03	18.42	17.49	39.92	40.50	4.39	1.80
$asMSE_{100}$	1.83	1.77	29.46	28.87	12.75	8.60	0.37	0.10
Graph Coloring Problem								
MSE	0.46	0.40	9.04	8.08	78.96	84.30	12.49	9.50
$asMSE_{10}$	0.65	0.63	14.18	14.20	58.71	60.20	1.49	0.40
$asMSE_{100}$	1.01	0.97	21.09	20.53	25.80	21.00	0.16	0.00

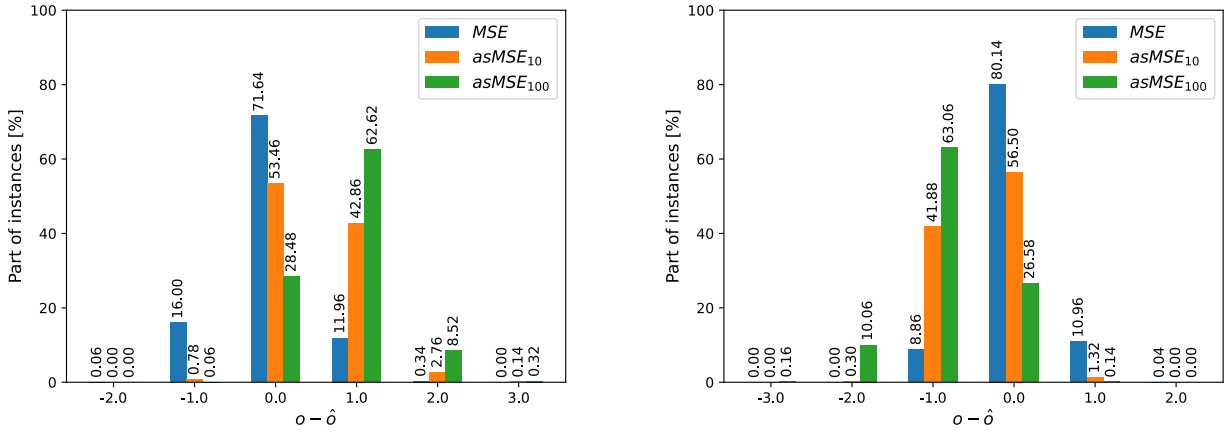
approximation of the solution values. The RMSE corresponds to the following equation, which uses the same notation as Eq. (24).

$$RMSE(o, \hat{o}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(o_i - \hat{o}_i)^2} \tag{25}$$

The next value is the average percentage of generated solution values that are equal to the value of the optimal solution. The final measure is the percentage of OBSV.

The following method is used for accessing the performance of ensembles. For each problem instance and selected loss function, 20 ensembles are generated and the previously listed aggregated values are used for evaluation. Each such ensemble uses 5 randomly selected GCNs from the 20 trained ones.

The results of the conducted computational experiments are illustrated in Table 1. First we discuss the results without the use of ensembles depicted in Table 1 as the "Single" columns. In the case of the GCP, the *sageGin* using the



(a) MCP (b) GCP

FIGURE 6. Graphical illustration of the percentage of test instances for which ensembles based on *sageGIN* had a specific difference between the GCN acquired value and the optimal value. For the MCP negative values correspond to OBSV and for the GCP positive ones, indicating a maximization and minimization problem, respectively.

MSE loss function managed to find solutions equal to the optimal solution in 78.96% of the instances in the test set. It should be noted that these results are comparable to the use of a specialized GCN for the GCP from [34] where for similar sized graphs 82% optimal solutions have been acquired. It is important to point out that this method uses a higher level of information than the proposed GCN. To be specific, the training data consists of graph instances and coloring information for all the nodes in the corresponding optimal solution. In addition, in the work in [34], the GCN does not directly provide a solution value but is a part of an iterative procedure. The RMSE error is 0.46 which corresponds to an average RMSRE of 9.04%. The number of OBSV is approximately half of the ones that are not equal to the optimal value. The use of the asymmetric function manages to significantly decrease the number of OBSV. In the case of $asMSE_{10}$, the percentage of OBSV has been decreased close to 10 times to 1.49%. It should be noted that this positive effect comes with an increase in RMSE and RMSRE of close to 50% and a decrease in the number of found optimal values. In the case of $asMSE_{100}$, the number of generated OBSV is less than 0.2%. The downside of using $asMSE_{100}$ is that it drastically decreases the quality of found solutions which is reflected in the values of RMSE, RMSRE and the number of found optimal solution values.

A similar behavior can be observed in the case of MVCP, MCP and MDSP. The percentage of solution values equal to the optimal solutions for the MVCP, MCP and MDSP is 39.10%, 67.18% and 59.53%, respectively. The percentage of found optimal solutions for a specific GCOP is closely related to the difference between the maximum and minimum values of the optimal solution in the training set. To be exact, in the case of MCP and MDSP, where the span of these values is lower, the percentage of found solution values equal to the optimal value is higher. On the other hand, the opposite is true for the RMSE and RMSRE.

The use of ensembles proves to be highly effective. In the case of using the MSE loss function, the relative improvement achieved using ensembles is between 10%-20% for RMSE, RMSRE and number of found optimal values. The decrease in the number of OBSV provides a relative improvement of 5%-10%. It is interesting that the use of ensembles in combination with asymmetric loss functions manage to produce a relative decrease in the number of infeasible solution values by more than 70% in most cases. In the case of $asMSE_{10}$, the use of ensembles in almost all cases provides additional improvement to the values of RMSE, RMSRE and number of found optimal values. On the other hand, when $asMSE_{100}$ is used with the intention of removing almost all the OBSV, it has a negative effect on the percentage of found optimal solutions but has a positive effect on the RMSE and the RMSRE.

To have a better understanding of the effect of using different loss functions in the training of the GCN, a graphical representation of the distribution of the differences between the optimal solutions values and the ones acquired by *sageGIN* for ensembles is provided in Fig. 6 for the MCP and GCP. To be specific, the figure shows the percentage of instances in the test set where the acquired solution differs from the optimal solution by a certain integer value. It is important to point out that the error can be positive and negative, consequently it provides information on OBSV. The first issue that can be observed for the *MSE* loss function is that the percentage of solutions having the same absolute error value is similar. This indicates the significant drawback that GCNs trained in this way are not able to differentiate between feasible solution values and OBSV, which give impossible better solution values than the optimal solution. On the other hand for $asMSE_{10}$ and $asMSE_{100}$, this distribution is highly asymmetric, such that almost no OBSV are obtained. The issue is that this comes at the cost of the number of instances for which the optimal solution value is found. In case of $asMSE_{100}$, the number of feasible solutions having a small

error becomes even higher than the number of found optimal solution values, indicating that a too strict penalization leads to a decrease of the solution quality.

V. CONCLUSION

In this paper, a GCN approach has been proposed for solving the decision version of some classic GCOPs. Two common GCN structures have been evaluated based on the standard GCN layer and the SAGE layer. In addition, a more advanced structure was proposed based on the GIN. An effective method has been used to make training possible for differently sized graphs based on the normalization of the objective function and adapting the input graphs' node properties. One of the issues of using GCNs for approximating solution values of GCOPs is that there are maybe no corresponding feasible solutions for the generated value. In other words, the approximate solution values are lower or greater than the optimal solution for the corresponding minimization or maximization problems, respectively. To address this issue, an asymmetric loss function is proposed and used in the training of the GCN.

The proposed GCN approach has been evaluated based on computational experiments. To be able to conduct such experiments, a large set of training data has been generated for the MVCP, MCP, MDSP and GCP. Training pairs consisting of graphs of varying sizes and densities with corresponding optimal solutions have been generated and made available online. The performed experiments have shown that the proposed GIN-based structure of a GCN significantly outperforms the basic ones. In addition, it has been shown that the use of ensembles is highly effective when the proposed GCN structure is used, especially in the case of an asymmetric loss function. Another important observation that has been made is that the use of an asymmetric loss function can drastically decrease the number of OBSV for which there is no feasible solution. On the other hand, lowering the probability of generating such solution values below a certain level greatly decreased their quality. The proposed framework can be applied to a wide range of GCOPs without exploiting their specific properties, while managing to produce good quality approximations to optimal solution values.

The primary limitation of the method is that, in some cases, it may produce objective values that do not correspond to feasible solutions. The proposed approach addresses this issue by employing an asymmetric loss function that helps control the probability of generating non-feasible solutions. Another drawback is associated with the use of randomly generated graph instances. Many real-world applications of the GCOPs involve systems with underlying structures, such as planarity or other similar properties. The GCNs trained using randomly generated training data may not be suitable for such cases, and it may be necessary to generate training data with graphs that exhibit the appropriate structure.

There are several directions for potential extension of this research. Firstly, it is possible to use metaheuristics to optimize the hyperparameters of the proposed GCNs to enhance the performance, which is commonly done for ANNs, i.e. [52]. Another direction is to examine the use of GCNs for approximating solution values for variations of the used GCOPs having weights and more complex constraints like connectivity. It would be interesting to evaluate the potential improvement in performance of GCNs when larger training sets are used in combination with higher performance hardware. Exploring the use of the proposed approach in combination with feature-enriched core percolation in multiplex networks [53] for exploring more complex physics systems could also be of interest.

REFERENCES

[1] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, vol. 4, no. 11, Nov. 2018, Art. no. e00938.

[2] O. I. Abiodun, M. U. Kiru, A. Jantan, A. E. Omolara, K. V. Dada, A. M. Umar, O. U. Linus, H. Arshad, A. A. Kazaure, and U. Gana, "Comprehensive review of artificial neural network applications to pattern recognition," *IEEE Access*, vol. 7, pp. 158820–158846, 2019.

[3] A. H. Elsheikh, S. W. Sharshir, M. A. Elaziz, A. E. Kabeel, W. Guilan, and Z. Haiou, "Modeling of solar energy systems using artificial neural network: A comprehensive review," *Sol. Energy*, vol. 180, pp. 622–639, Mar. 2019.

[4] J. Runge and R. Zmeureanu, "Forecasting energy use in buildings using artificial neural networks: A review," *Energies*, vol. 12, no. 17, p. 3254, Aug. 2019.

[5] S. Ma, X. Zhang, C. Jia, Z. Zhao, S. Wang, and S. Wang, "Image and video compression with neural networks: A review," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 6, pp. 1683–1698, Jun. 2020.

[6] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: Analysis, applications, and prospects," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 12, pp. 6999–7019, Dec. 2021.

[7] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021.

[8] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, Jan. 2020.

[9] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, and D. Yin, "Graph neural networks for social recommendation," in *Proc. World Wide Web Conf.*, 2019, pp. 417–426.

[10] S. Wu, F. Sun, W. Zhang, X. Xie, and B. Cui, "Graph neural networks in recommender systems: A survey," *ACM Comput. Surv.*, vol. 55, no. 5, pp. 1–37, May 2023.

[11] L. Yao, C. Mao, and Y. Luo, "Graph convolutional networks for text classification," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 7370–7377.

[12] J. Shlomi, P. Battaglia, and J.-R. Vlimant, "Graph neural networks in particle physics," *Mach. Learn., Sci. Technol.*, vol. 2, no. 2, Jan. 2021, Art. no. 021001.

[13] C. Chen, K. Li, S. G. Teo, X. Zou, and Z. Zeng, "Gated residual recurrent graph neural networks for traffic prediction," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 485–492.

[14] W. Jiang and J. Luo, "Graph neural network for traffic forecasting: A survey," *Exp. Syst. Appl.*, vol. 207, Nov. 2022, Art. no. 117921.

[15] A. Derrow-Pinion et al., "Eta prediction with graph neural networks in Google maps," in *Proc. 30th ACM Int. Conf. Inf. Knowl. Manag.*, Oct. 2021, pp. 3767–3776.

[16] W. Liao, B. Bak-Jensen, J. R. Pillai, Y. Wang, and Y. Wang, "A review of graph neural networks and their applications in power systems," *J. Modern Power Syst. Clean Energy*, vol. 10, no. 2, pp. 345–360, 2022.

[17] O. Wieder, S. Kohlbacher, M. Kuenemann, A. Garon, P. Ducrot, T. Seidel, and T. Langer, "A compact review of molecular property prediction with graph neural networks," *Drug Discovery Today, Technol.*, vol. 37, pp. 1–12, Dec. 2020.

[18] S. Zhang, H. Tong, J. Xu, and R. Maciejewski, "Graph convolutional networks: A comprehensive review," *Comput. Social Netw.*, vol. 6, no. 1, pp. 1–23, Dec. 2019.

[19] M. Welling and T. N. Kipf, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Represent.*, 2016, pp. 1–14.

[20] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–11.
[21] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–12. [Online]. Available: https://openreview.net/forum?id=rJXMpikCZ
[22] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" 2018, *arXiv:1810.00826*.
[23] J. Du, S. Zhang, G. Wu, J. M. F. Moura, and S. Kar, "Topology adaptive graph convolutional networks," 2017, *arXiv:1710.10370*.
[24] N. Mazyavkina, S. Sviridov, S. Ivanov, and E. Burnaev, "Reinforcement learning for combinatorial optimization: A survey," *Comput. Oper. Res.*, vol. 134, Oct. 2021, Art. no. 105400.
[25] Y. Bengio, A. Lodi, and A. Prouvost, "Machine learning for combinatorial optimization: A methodological tour d'horizon," *Eur. J. Oper. Res.*, vol. 290, no. 2, pp. 405–421, 2020.
[26] Q. Cappart, D. Chételat, E. Khalil, A. Lodi, C. Morris, and P. Velickovic, "Combinatorial optimization and reasoning with graph neural networks," 2021, *arXiv:2102.09544*.
[27] N. Vesselinova, R. Steinert, D. F. Perez-Ramirez, and M. Boman, "Learning combinatorial optimization on graphs: A survey with applications to networking," *IEEE Access*, vol. 8, pp. 120388–120416, 2020.
[28] A. Nowak, S. Villar, A. S. Bandeira, and J. Bruna, "Revised note on learning quadratic assignment with graph neural networks," in *Proc. IEEE Data Sci. Workshop (DSW)*, Jun. 2018, pp. 1–5.
[29] C. K. Joshi, T. Laurent, and X. Bresson, "An efficient graph convolutional network technique for the travelling salesman problem," 2019, *arXiv:1906.01227*.
[30] M. Prates, P. H. Avelar, H. Lemos, L. C. Lamb, and M. Y. Vardi, "Learning to solve NP-complete problems: A graph neural network for decision TSP," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 4731–4738.
[31] S. Manchanda, A. Mittal, A. Dhawan, S. Medya, S. Ranu, and A. Singh, "Learning heuristics over large graphs via deep reinforcement learning," 2019, *arXiv:1903.03332*.
[32] Z. Li, Q. Chen, and V. Koltun, "Combinatorial optimization with graph convolutional networks and guided tree search," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 1–10.
[33] K. Langedal, J. Langguth, F. Manne, and D. T. Schroeder, "Efficient minimum weight vertex cover heuristics using graph neural networks," in *Proc. 20th Int. Symp. Experim. Algorithms*, 2022, pp. 1–17.
[34] H. Lemos, M. Prates, P. Avelar, and L. Lamb, "Graph colouring meets deep learning: Effective graph neural network models for combinatorial problems," in *Proc. IEEE 31st Int. Conf. Tools Artif. Intell. (ICTAI)*, Nov. 2019, pp. 879–885.
[35] K. Abe, Z. Xu, I. Sato, and M. Sugiyama, "Solving NP-hard problems on graphs with extended AlphaGo zero," 2019, *arXiv:1905.11623*.
[36] S. Boettcher, "Inability of a graph neural network heuristic to outperform greedy algorithms in solving combinatorial optimization problems," *Nature Mach. Intell.*, vol. 5, pp. 24–25, Jan. 2022.
[37] A. Loukas, "What graph neural networks cannot learn: Depth vs width," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–17.
[38] R. Jovanovic. (2023). *Training Data for Decision Versions of Graph Combinatorial Optimization Problems (TrainGCOP)*. [Online]. Available: https://data.mendeley.com/datasets/3wkvfyb362/draft?a=6d79d4ec-a77b-43c1-8112-67d2ff429af3
[39] A. Jabrayilov and P. Mutzel, "New integer linear programming models for the vertex coloring problem," in *Proc. Latin Amer. Symp. Theor. Informat.* Cham, Switzerland: Springer, 2018, pp. 640–652.
[40] M. Campelo, R. Correa, and Y. Frota, "Cliques, holes and the vertex coloring polytope," *Inf. Process. Lett.*, vol. 89, no. 4, pp. 159–164, Feb. 2004.
[41] M. Campelo, V. A. Campos, and R. C. Correa, "On the asymmetric representatives formulation for the vertex coloring problem," *Discrete Appl. Math.*, vol. 156, no. 7, pp. 1097–1111, Apr. 2008.
[42] T. Szandała, "Review and comparison of commonly used activation functions for deep neural networks," in *Bio-inspired Neurocomputing*. Berlin, Germany: Springer, 2021, pp. 203–224.
[43] P. M. Granitto, P. F. Verdes, and H. A. Ceccatto, "Neural network ensembles: Evaluation of aggregation algorithms," *Artif. Intell.*, vol. 163, no. 2, pp. 139–162, Apr. 2005.
[44] E. Elson Kosasih, J. Cabezas, X. Sumba, P. Bielak, K. Tagowski, K. Idanwekhai, B. A. Tjandra, and A. R. Jamasb, "On graph neural network ensembles for large-scale molecular property prediction," 2021, *arXiv:2106.15529*.
[45] Q. Lin, S. Yu, K. Sun, W. Zhao, O. Alfarraj, A. Tolba, and F. Xia, "Robust graph neural networks via ensemble learning," *Mathematics*, vol. 10, no. 8, p. 1300, Apr. 2022.
[46] A. Krogh and J. Vedelsby, "Neural network ensembles, cross validation, and active learning," in *Proc. Adv. Neural Inform. Process.*, vol. 7, 1995, pp. 231–238.
[47] A. J. C. Sharkey, "On combining artificial neural nets," *Connection Sci.*, vol. 8, nos. 3–4, pp. 299–314, Dec. 1996.
[48] I. Kandel and M. Castelli, "The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset," *ICT Exp.*, vol. 6, no. 4, pp. 312–315, Jan. 2020.
[49] D. Chen, Y. Lin, W. Li, P. Li, J. Zhou, and X. Sun, "Measuring and relieving the over-smoothing problem for graph neural networks from the topological view," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 4, pp. 3438–3445.
[50] G. Li, M. Muller, B. Ghanem, and V. Koltun, "Training graph neural networks with 1000 layers," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 6437–6449.
[51] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, "Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions," *J. Big Data*, vol. 8, no. 1, pp. 1–74, Mar. 2021.
[52] I. Strumberger, E. Tuba, N. Bacanin, R. Jovanovic, and M. Tuba, "Convolutional neural network architecture design by the tree growth algorithm framework," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2019, pp. 1–8.
[53] Y. Shang, "Feature-enriched core percolation in multiplex networks," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 106, no. 5, Nov. 2022, Art. no. 054314.

RAKA JOVANOVIC received the B.S. and M.S. degrees in computer science from the Faculty of Mathematics, University of Belgrade, Serbia, and the Ph.D. degree from the University of Belgrade, Serbia. He is currently an Associate Research Professor with the University of Belgrade. He is also a Scientist with the Qatar Environment and Energy Research Institute (QEERI), Hamad Bin Khalifa University, with a background in the field of applied mathematics and operations research. Before joining QEERI, he was with the Institute of Physics, University of Belgrade, and Texas A&M University at Qatar. He has published more than 100 papers in journals, conference proceedings, and books. His research interests include developing mathematical models and applying optimization methods for real-world applications.

MICHAEL PALK received the M.Sc. degree (Hons.) in business mathematics from the University of Hamburg, where he is currently pursuing the Ph.D. degree with the Institute of Information Systems. His industry experiences include positions at Kühne+Nagel, Ginkgo Analytics, and Nordmetall. His current research interests include social media, machine learning, and graph theory. He serves as a reviewer for various journals and conferences, while his teaching experiences consist of several math and business classes, as well as the supervision of bachelor's and master's thesis.

SERTAC BAYHAN (Senior Member, IEEE) received the B.S. degree (Hons.) and the M.S. and Ph.D. degrees in electrical engineering from Gazi University, Ankara, Turkey, in 2008 and 2012, respectively.He joined Gazi University as a Lecturer, in 2008, where he was promoted as an Associate Professor and a Full Professor, in 2017 and 2022, respectively. He was an Associate Research Scientist with Texas A&M University at Qatar, from 2014 to 2018. Currently, he is a Senior Scientist with the Qatar Environment and Energy Research Institute (QEERI) and an Associate Professor with the Sustainable Division, Hamad Bin Khalifa University. He has acquired $13M in research funding and published more than 170 papers in mostly prestigious IEEE journals and conferences. He is also the coauthor of three books and six book chapters. His research interests include power electronics and their applications in next-generation power and energy systems, including renewable energy integration, electrified transportation, and demand-side management. He was a recipient of many prestigious international awards, such as the Teaching Excellence Award in recognition of outstanding teaching in Texas A&M University at Qatar, in 2022, the best paper awards in the 3rd International Conference on Smart Grid and Renewable Energy, Doha, Qatar, in March 2022, the 10th International Conference on Renewable Energy Research and Applications, Istanbul, Turkey, in September 2021, and the Research Fellow Excellence Award in recognition of his research achievements and exceptional contributions to the Texas A&M University at Qatar, in 2018. Because of the visibility of his research, he has been recently elected as the Chair of the IES Power Electronics Technical Committee. He currently serves as an Associate Editor for IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, IEEE JOURNAL OF EMERGING AND SELECTED TOPICS IN INDUSTRIAL ELECTRONICS, IEEE OPEN JOURNAL OF THE INDUSTRIAL ELECTRONICS SOCIETY, and *IEEE Industrial Electronics Technology News*, and a Guest Editor for the IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS.

STEFAN VOSS received the Diploma degree in mathematics and economics from the University of Hamburg and the Ph.D. and Habilitation degrees from the University of Technology Darmstadt. He is currently a Professor and the Director of the Institute of Information Systems, University of Hamburg. Moreover, he served as the Dean of the Hamburg Business School (School of Business Administration), until 2022. Previously, he was a Full Professor and the Head of the Department of Business Administration, Information Systems and Information Management, University of Technology Braunschweig, Germany, from 1995 to 2002. Furthermore, he is consulting with several companies. He is the author and coauthor of several books and several hundred papers in various journals. His research interests include quantitative/information systems approaches to supply chain management and logistics, including public mass transit and telecommunications. In the German Handelsblatt and Wirtschaftswoche rankings, he is continuously within the top ten professors in business administration within the German speaking countries. He serves on the editorial board of some journals, including *Public Transport*. He is frequently organizing workshops and conferences.