GNN-based Conceptual Model Modularization: Approach and GA-based Comparison

Syed Juned Ali[1][0000−0003−1221−0278], MohammadHadi Dehghani[2][0009−0002−5540−5841], Manuel Wimmer[2][0000−0002−1124−7098], and Dominik Bork[1][0000−0001−8259−2297]

[1] TU Wien, Business Informatics Group, Vienna, Austria
{syed.juned.ali,dominik.bork}@tuwien.ac.at
[2] Johannes Kepler University Linz, CDL-MINT, Linz, Austria
{mohammadhadi.dehghani,manuel.wimmer}@jku.at

Abstract. Due to the crucial role conceptual models play in explicitly representing a subject domain, it is imperative that they are comprehensible and maintainable by humans. Modularization, i.e., decomposing an overarching, monolith model into smaller modules, is an established technique to make the model comprehensible and maintainable. Genetic Algorithms (GA) have been applied to modularize conceptual models by formulating desired structural model characteristics as multiple objectives. Recently, Graph Neural Networks (GNN)-based methods have shown promising performance in graph processing tasks, including graph clustering but outside the conceptual modeling domain. In this paper, we present a novel approach for GNN-based conceptual model modularization and comparatively analyze our approach against an existing multi-objective GA-based one. Furthermore, we provide a comparative analysis of our novel GNN model against two existing GNN-based graph clustering approaches. We investigate the dependence of the quality of the modularized solutions on the model size. We discuss the comparative results of our novel GNN-based approach and the existing GA-based approach to derive future research lines. Furthermore, our results show, that our proposed GNN-based modularization outperforms the existing GNN-based graph clustering approaches and provides a suitable alternative compared to the GA-based modularization.

Keywords: Conceptual modeling · Model Modularization · Graph neural networks · Genetic algorithms · ER · Data modeling.

1 Introduction

Conceptual modeling allows to understand complex domains and supports communication among stakeholders, and thus, is essential for enterprise and information systems engineering and beyond [30]. Due to their crucial role, comprehension of conceptual models is a prerequisite for value creation. For instance, Entity Relationship (ER) models are a prominent approach to develop

and analyze abstractions as conceptual representation of data models used by humans. By this, ER models abstract the technical details of database models and implementations, providing dedicated views of conceptual structures hidden in implementation-oriented data schemas [8].

It has been shown that the usefulness of conceptual models for humans is inversely proportional to the models' sizes [37]. Models having already more than 30 nodes are considered challenging for human comprehension. The more relationships, the less comprehension is given due to the accompanying increase in complexity [37]. Therefore, the increased size and complexity can make models *cognitively intractable* [12]. Clustering or modularizing conceptual models[‡] into smaller chunks enables humans to communicate better, validate, and maintain very large models [37].

Existing modularization approaches mostly leverage the topological properties of conceptual models [12], i.e., characteristics that relate to the graph's structure and the arrangement of its elements (vertices and edges) such as degree, connectivity, cyclicity. In the past, Genetic Algorithms (GAs) have been used to transform the modularization of graphs into an optimization problem to partition a model into modules, such that the nodes within each module are closer related to each other than the nodes in the other clusters. These approaches allow exploring a solution space more efficiently than exhaustive search methods. They can be particularly useful when dealing with large and complex graphs where traditional clustering methods might struggle to find optimal or near-optimal solutions. GAs proved valuable for tasks such as community detection in social networks [4], protein function prediction in bioinformatics [35], and modular decomposition in software engineering [8].

Graph Neural Networks (GNN) are a variant of neural networks that can apply deep learning methods on graph-based inputs and train the deep learning model for a specific task. GNN-based solutions learn a vector-based representation, i.e., node embedding that is supposed to reflect the graph structural information captured by a given node in the context of the entire graph. While GNNs have been successfully applied, among many others, in applications such as medical diagnosis and electronic health records modeling [21], drug discovery and chemical compounds synthesis [40], recommender systems [39] and text classification [23], GNNs also showed promising results for unsupervised learning-based graph clustering [10, 27, 41, 42]. Conceptual Model Modularization (CMM) can be transformed into a graph clustering task, however, GNNs have not yet been applied to modularize conceptual models. Therefore, in this paper, we present a novel GNN-based conceptual model modularization approach, a deep conceptual model modularizer (DCMM).

We provide a comparative evaluation with an existing GA-based approach to assess the quality of our GNN-based approach. For this, we extend the GA-based modularization approach ModuleER [8] initially developed for ER models for being able to modularize Ecore-based UML Class Diagrams (CD). We then evaluate the performance of our novel GNN-based approach on CDs and compare

[‡]We use clustering, partitioning, and modularization interchangeably in this paper.

it with the extended ModuleER approach using *cohesion* and *coupling* as the modularization quality metrics. Furthermore, we also adapt the existing GNN-based graph clustering approaches Deep Graph Infomax (DGI) [41] and Deep Modularity Network (DMoN) [27] for the CMM task to provide a comparative analysis of the state-of-the-art GNN models with our DCMM GNN model.

The main contributions of this paper comprise: (*i*) DCMM: a novel GNN-based approach for CMM; (*ii*) the adaptation of existing GNN-based graph clustering approaches for CMM; (*iii*) a comparative evaluation of GNN-based approaches with GA-based ones for CMM; (*iv*) a comparative evaluation of several GNN-based graph clustering approaches for CMM; and, finally, (*v*) an analysis of the effect of model size on the different approaches. Note that in this paper we do not use the semantic aspects of conceptual models as graphs and focus on the structural aspects of the conceptual models as graphs.

The remainder of this paper is structured as follows. Section 2 provides background information to the different conceptual and technological aspects of our work. Section 3 presents the details of DCMM. Section 4 comprises the experimental setup for the comparative evaluation of GNN and GA-based modularization and the experimental results. In Section 5, we provide insights gained from our results. Section 6 discusses related work before we conclude this paper in Section 7 with an outlook to future work.

2 Background

We now introduce the relevant background for this work. In particular, we discuss the existing GA and GNN approaches as well as the metrics to be used for the comparative evaluation.

2.1 Conceptual Model Modularization

Modularization of conceptual models defined in a specific modeling language (e.g., ER, UML) aims to determine a suitable set of elements to form a module. The module elements depend on the intended purpose of modularization while fulfilling the definition of a module. For e.g, if the modularization aims to determine modules optimized to answer individual queries, then the generated modules should be composed of all elements necessary to answer a considered query [20]. Software clustering is a modularization technique for source code elements such as classes or functions. These elements are grouped into sets, so-called modules, in such a way that elements residing in the same module are more similar (to a given definition) to each other than to those in other modules [29].

2.2 Graph-based modularization

Graph-based modularization uses the structural aspects of a model for identifying module candidates [29]. It produces modules by interpreting a conceptual model as a graph and applying algorithms to extract related (or sufficiently related) concepts. Graph-based modularization approaches are often intuitive and employ statistical methods to determine similarities of concepts or clusters and require the model to have a graph-based representation (i.e., a set of vertices

and edges) [20]. The model is generally represented as an *adjacency matrix* for graph-based clustering tasks. In this matrix, each element captures if an edge between the nodes denoted by the indices of the element exists. Each node can denote an entity of a model, and each edge can represent a dependency between the entities, e.g., the composition relationship.

2.3 Search-based Modularization

Search techniques, such as Genetic Algorithms (GA), use one or more objective functions to guide the modularization process, i.e., search for the optimal solutions, i.e., the solutions that provide optimal scores using the objective functions [26]. These functions quantify the quality of a candidate solution. In case of CMM, objective functions evaluate the quality of the modules and the modularization achieved for a given set of modules during the process of search the optimal modules. Frequently, the objective functions' intuition is maximizing cohesion within and minimizing coupling across the modules [29]. ModuleER [8,14] uses a GA-based approach for the modularization of ER models. In [2], ModuleER is extended to support graph modularization of multiple different modeling languages using a Generic GA-based Modularization Framework.

2.4 Graph Neural Networks-based Graph Representation Learning

The nodes of a graph and their structural properties can be represented in dense fixed-sized vectors [36]. Recently, there have been approaches that can learn node representations using the structural aspects of the graphs. Graph Neural Networks (GNNs) are neural models that learn graph representations via *message passing* between graph nodes by information aggregation of a node from its neighborhood [16]. The fundamental paradigm of these node embeddings is, that "similar" nodes have close embeddings. The similarity of nodes is often defined based on their distance in a graph, e.g., based on their co-occurrence probability in a random walk [15,16,42]. It is also argued that two nodes should be similar if they are similar to a graph summary representation [36]. *Node2vec* learns node embeddings by maximizing the likelihood of preserving network neighborhoods of nodes by exploring the network through graph traversal [15]. In recent years, variants of GNNs such as Graph Convolutional Networks (GCN) [17], GraphSage [17], and Graph Attention Networks (GAT) [36] have demonstrated good performance on deep learning tasks such as link prediction, node classification, graph classification, and graph mining [43]. Particularly GraphSage [17] learns a function that generates embeddings by sampling and aggregating features from a node's local neighborhood. This inductive approach enables the model to generalize to unseen nodes or entirely new graphs.

2.5 GNN-based Modularization

The node embeddings from GNNs can be obtained as a result of optimizing a GNN model to minimize an objective function i.e., the loss function. The loss

minimization of a GNN model can be used to steer the GNN toward optimization of graph modularization metrics like cohesion and coupling. Existing approaches i.e., DGI [41] and DMoN [27] use modularity [28] (see Eq. 1) as a modularization metric that approaches graph clustering from a statistical perspective.

$$Q = \frac{1}{2m} \sum_{i,j} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta(c_i, c_j) \tag{1}$$

Modularity measures Q as the divergence between the intra-cluster edges from the expected one where m is the total edges, k_i and c_i denotes the degree and cluster of node i and δ function is defined as $\delta(c_i, c_j) = 1$ if nodes i and j are in the same cluster c, and 0 otherwise. Modularity remains one of the most commonly used graph clustering metrics in literature [13].

2.6 Modularization Quality Metrics

Once a modularized solution is available, its cohesion and coupling can be evaluated as given by Eq. 2 and Eq. 3, respectively.

$$cohesion = \frac{1}{c} \sum_{i}^{c} \frac{E_{C_i}}{N_{C_i} * (N_{C_i} - 1) * 0.5} \tag{2}$$

$$coupling = \sum_{C_i, C_j} E_{C_i, C_j} \tag{3}$$

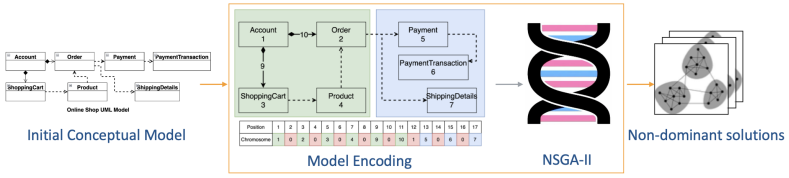
In cohesion, c is the number of cluster, E_{C_i} denotes the number of edges in cluster C_i, and N_{C_i} is the number of nodes in C_i. In coupling, E_{C_i,C_j} denotes the number of edges between the cluster C_i and C_j.

3 Deep Conceptual Model Modularizer

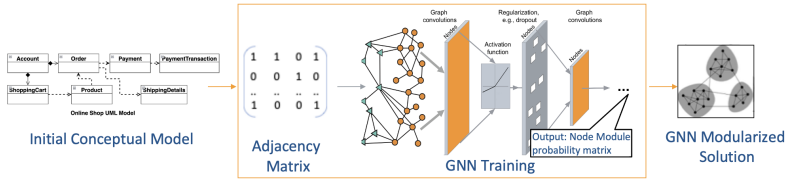
We now describe our end-to-end unsupervised GNN-based conceptual model modularization approach which is applied to UML class diagrams. We sketch the major steps involved in GNN-based CMM in general in Fig. 1b and our node2vec-based structural embedding augmented GNN-based CMM in Fig. 1c and further contrast it with the Genetic Algorithms-based approach ModulER [8] adapted for UML class diagrams in Fig. 1a that we use for a comparative evaluation of our approach. Furthermore, we provide the details about the adaptations of the existing GNN-based graph clustering approaches for our usecase of CMM so as to evaluate the performance of DCMM with existing GNNs.

3.1 Model to Graph Transformation

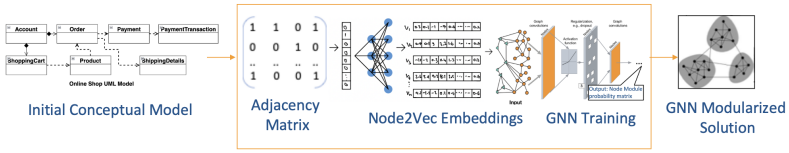
In order for a UML class diagram (CD) to be modularized by graph-based algorithms, each CD undergoes a model to graph transformation in a preprocessing step. The resulting graph is used as an input to both approaches. We use the generic conceptual model to knowledge graph transformation methods introduced in [1, 32], which transform models into directed graphs. Each class in the class diagram is considered as a node and each reference and an inheritance relation is treated as an edge in the resulting graph.



(a) Genetic Algorithms-based CMM [2]



(b) General GNN-based CMM



(c) Deep Conceptual Model Modularizer

Fig. 1: Conceptual Model Modularization Approaches

3.2 Deep Conceptual Model Modularization

Once we have generated the graph, we transform the graph into an *adjacency matrix* A (cf. Section 2) as shown in Fig. 1. Next, we generate node embeddings for each node in the graph using A. The embeddings-based representation of the nodes is generated by running the *node2vec* algorithm that captures the structural properties of nodes in a way that preserves their neighborhood relationships. Node2vec transforms each node into a dense fixed-sized vector X in R^{n*d} (see Eq. 4.1) where d is the dimension of the embedding.

$$X = \Phi(A) \quad \text{where} \quad \Phi = \text{node2vec} \quad \text{and} \quad X \in \mathbb{R}^{n \times d} \tag{4.1}$$

$$H^{(0)} = X$$

$$H^{(l)} = \text{ReLU}\left(\sum_{j \in \mathcal{N}(i)} \frac{1}{|\mathcal{N}(i)|} W^{(l)} H_j^{(l-1)}\right) \quad \text{for} \quad l = 1, \ldots, L \tag{4.2}$$

$$\text{where} \quad \mathcal{N}(i) \quad \text{is the set of neighbors of node} \quad i$$

$$Y = \text{softmax}\left(\Omega(X, A)\right) \quad \text{where} \quad \Omega = H^{(L)} \quad \text{and} \quad Y \in \mathbb{R}^{n \times C} \tag{4.3}$$

$$\text{Cohesion} = \sum_{c=1}^{k} \frac{\sum_{i,j} A_{ij} \cdot Y_{ic} \cdot Y_{jc}}{\sum_{i,j} Y_{ic} \cdot Y_{jc}} \tag{4.4}$$

$$\text{Coupling} = \sum_{i,j} A_{ij} \cdot \left(1 - \sum_{c=1}^{k} Y_{ic} \cdot Y_{jc}\right) \tag{4.5}$$

$$\mathcal{L}_{\text{dcmm}} = -\text{Cohesion} + \text{Coupling} \tag{4.6}$$

Next we define DCMM in Eq. 4.2 and 4.3 where DCMM takes the node embeddings X and the adjacency matrix A and maps each node embedding to a particular cluster out of C clusters thereby transforming X in R^{n*d} matrix to Y in R^{n*C} matrix (see. Eq. 4.3). DCMM is built using l layers of the GraphSage GNN model as shown in Eq. 4.2. The GraphSage layers apply ReLU [11] activation which adds the crucial non-linearity required to learn the complex mapping C that provides high modularization scores. Using C, the node is assigned to the module with the highest probability, and the modularization result quality is evaluated using a modularization metric. In general, a GNN model learns to maximize the modularization quality by minimizing the error in the difference between the evaluated modularization and the ground truth values. In unsupervised learning, however, we do not have ground truth values. Therefore, we use metrics that reflect the modularization quality such as cohesion (Eq. 4.4) and coupling (Eq. 4.5). We consider a combination of such metrics as the measure of error during training and aim to minimize such error (see Eq. 4.6).

3.3 Adaptation of Existing GNN-based Approaches

To compare with existing GNN approaches, we adapt *i*) DMoN, an unsupervised pooling method inspired by the modularity measure of clustering quality [27] and, *ii*) Community Deep Graph Infomax (CommDGI), a GNN designed to handle community detection problems by using a mechanism to capture neighborhood and community information in graphs [41] for CMM. Both DMoN and CommDGI use the modularity metric in their loss function; however, DMoN adds a regularization parameter to prevent trivial solutions to the optimization problem. CommDGI further adds graph and cluster information exchange loss to their loss functions. The modularity loss is given in Eq. 5.

$$\mathcal{L}_{\text{modularity}} = -\frac{1}{2m}\text{Tr}(Y^T(A - \frac{dd^T}{2m}))Y) \tag{5}$$

where Tr is trace of matrix, i.e., sum of the elements in the left diagonal and d is the degree of a node. We provide the code for all the three GNN models, namely DCMM as well as adapted DMoN and DGI for CMM publicly available[§].

4 Comparative Experimental Evaluation

To evaluate our GNN approach, we compare it with the state-of-the-art ModuleER approach introduced in Section 2, which uses NSGA-II and existing GNNs — DMoN and DGI. With this evaluation, we aim to systematically answer the following research questions:

[RQ1] How do GNNs perform compared to GA for CMM?
- **[RQ1.1]** How do GNNs perform compared to GA on hypervolume, cohesion, and coupling metrics?

[§]https://github.com/junaidiiith/dcmm

Table 1: Model to Graph Transformation Mappings

Size Category	Num. of Models	Min/Max Nodes	Avg±Std Nodes	Min/Max Edges	Avg±Std Edges	Min/Max Combined	Avg±Std Combined
Small	132	30/64	38 ± 7	41/128	80 ± 22	71/163	118 ± 26
Medium	104	31/132	70 ± 19	95/305	178 ± 56	164/420	249 ± 66
Large	27	55/226	130 ± 50	291/1552	591 ± 273	425/1778	722 ± 303
Total	263	30/226	60 ± 34	41/1552	171 ± 177	71/1778	232 ± 206

[RQ1.2] How do GNNs perform compared to GA for different model sizes?

[RQ2] How does DCMM perform compared to existing GNNs?

[RQ2.1] How does DCMM perform compared to DMoN and DGI on hypervolume, cohesion, and coupling metrics?

[RQ2.2] How do DCMM perform compared to DMoN and DGI for different model sizes?

4.1 Experimental Setup

In the following, we elaborate on the experimental set up the dataset used, the evaluation method and the metrics used for a comparative analysis.

Dataset Used - To evaluate our approach and train the GNN models, we use the Modelset dataset [22] that contains more than 5000 Ecore models. We applied the following exclusion criteria to filter out models of insufficient quality or models which are not appropriate inputs for an modularization approach: *i*) models with less than 30 nodes as this is the limit of cognitive intractability [12]; *ii*) models with edges to nodes ratio of less than 1.2 to reject poorly connected models such as models with a large number of nodes with only a few edges; *iii*) duplicate models; and *iv*) models that have very few nodes and edges uncommon. For e.g, given two graphs, one with 126 nodes and 469 edges and another graph with 127 nodes and 475 edges, if both graphs have more than 90% nodes in common, we consider them as duplicates and remove the one with the lower number of nodes. These exclusion criteria left us with 263 models of various sizes. We categorize these models in different sizes as show in Table 1. To compare the size of two models, we consider the combined number of nodes and edges and then chose the first 50 percentile of models as *small models*, *50-90 percentile as medium*, and *above 90 percentile as large models*. The table shows some descriptive statistics of the models in each model size category.

Technical Setup - In case of GNN-based modularization, we use an embedding dimension of 64 for *node2vec*, a learning rate of $1e-3$ and use the Adam optimizer [19] during the training phase. It is common to run the non-deterministic algorithm many times on each input instance and then perform the statistical tests. Therefore, in case of GA, in order to get robust results, we run GA $n = 30$ times and calculate the *score* values for each run. In case of GNN, even though the weights of the GNN model are initialised randomly, however, as a GNN optimizes the loss function and not searches for the best solution like in of GA, we get the same results in terms of modularization quality for each GNN run.

In [8], ModuleER uses five fitness functions: cohesion, coupling, the number of modules, the average number of elements per module, and the standard deviation of module sizes. We have modified the fitness functions to conform to the modularization quality metrics introduced in Section 2, i.e., only cohesion and coupling, as we observed that these two can capture the effects of the remaining metrics as well. The GA algorithm is run 30 times for each input model. In this setting, the population size is twice the size of each model's nodes, i.e., it varies between 60 and 452, and the number of iterations is fixed to 2000 because we investigated, that even for the largest models it takes less than 2000 iterations to reach a stable optimum point where no further improvements were observed.

Evaluation Methodology - In the case of GA, we get a set of non-dominating solutions for a multi-objective optimization problem, i.e., a Pareto Set. In case of multiple objectives, ModuleER [8] gives a Pareto Set of solutions for each problem, offering a spectrum of trade-offs between conflicting objectives. However, in case of GNN, we get a single solution. Therefore, in order to compare a single solution from GNN versus a pareto set from GA, we use the hypervolume metric to compare a multi-objective solution with a single solution based on Ishibuchi et al. [18]. NSGA-II runs in a non-deterministic way and it is common to run the non-deterministic algorithm many times on each input instance.

Fig. 2 shows a hypervolume metric-based comparison of a pareto set solution from a GA versus a GNN solution. Given a GA solution P_k of a k^{th} UML model with $k \in 1..N$ with P_k having n pareto sets $PS_j \in P_k$ from each run n with $j \in 1..n$ and PS_j having points $p_i \in PS_j$ where each p_i denoted a single modularized result, we define V_p in Eq. 6.1 as the hypervolume of a solution p using cohesion and coupling scores, both of which lie between 0 and 1. Coupling is a minimizing metric, therefore, we inverse coupling by plotting $1 - coupling$. The hypervolume of GNN is V_g where g is the GNN solution.

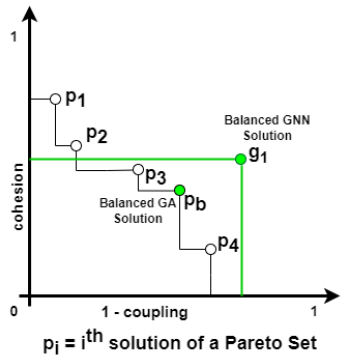


Fig. 2: Hypervolume-based GA and GNN solution comparison

Note, that if V_p of a given point is larger than the other, then the larger V_p solution balances both cohesion and coupling, (see p_3, p_b, g_1 in Fig. 2) whereas a smaller area indicates a preference towards one of the objectives (see p_1, p_4 in Fig. 2). The hypervolume of PS_j is the set union of V_{p_i} for all $p_i's \in PS_j$ (see. Eq. 6.2). We define the score s as given in Eq. 6.3 to calculate the average number of times V_g is higher than the hypervolume of the j^{th} pareto set V_{PS_j} $\forall PS_j$. Then, we define a $HP(.)$ in Eq. 6.4 that determines the better algorithm between GA and GNN, given s. Finally, for a given set of N models, we calculate $PHV(.)$ according to Eq. 6.5 to capture the fraction of models for which one approach outperforms the other. In other words, $P_{GNN} = 0.67$ implies that for a given set

of N models, GNN provides better hypervolume score, i.e., $PHV(s) = \text{GNN}$ for 67% of the models.

$$s = \frac{\sum_{j=1}^{n} \mathbf{1}(V_{p_g} \geq V_{PS_j})}{n} \tag{6.3}$$

$$V_p = cohesion * (1 - coupling) \tag{6.1}$$

$$V_{PS} = \bigcup_{p=1}^{n} V_p \tag{6.2}$$

$$HP(s) = \begin{cases} \text{GNN} & \text{if } s > 0.5, \\ \text{GA} & otherwise \end{cases} \tag{6.4}$$

$$PHV_{\text{x}} = \frac{\sum_{j=1}^{N} \mathbf{1}(HP(s_j) = \text{x})}{N} \quad \text{x} \in \{GNN, GA\} \tag{6.5}$$

$$M(p_b, g, m) = \begin{cases} \text{GNN} & \text{if } m_g > m_{p_b} \\ \text{GA} & otherwise \end{cases} \tag{6.6}$$

$$\overline{M}(P, g, m, x) = \frac{\sum_{j=1}^{n} \mathbf{1}(M(p_{b_j}, g, m) = x)}{n} \tag{6.7}$$

$$\sigma(PS, g, m) = \begin{cases} \text{GNN} & \text{if } \overline{M}(PS, g, m, GNN) > 0.5 \\ \text{GA} & \text{if } \overline{M}(PS, g, m, GA) \geq 0.5 \end{cases}, \text{m} \in \{cohesion, 1 - coupling\} \tag{6.8}$$

$$PM_{m,x} = \frac{\sum_{i=1}^{N} \mathbf{1}(\sigma(P_j, g_j, m) = \text{x})}{N} \text{ x} \in \{GNN, GA\}, m \in \{cohesion, 1 - coupling\} \tag{6.9}$$

Next, we further compare the cohesion and coupling scores of the solutions of GA and GNN. In case of GA, for a given pareto set of n non-dominating solutions, we select p_b to compare with the GNN solution and p_{b_j} as the trade-off point of each pareto set in j^{th} run. Then, given a GNN solution g and a metric m, we define $M(p_b, g, m)$ which checks if V_g is larger than V_{p_b}. If yes, then we consider GNN as the better trade-off solution over GA on metric m. Next, for a given model with n pareto sets, we take the average $\overline{M}$ that captures the average number of runs for which the approach x is better than the other approach in Eq. 6.7. Then, given the average score, we extract the better approach out of GA and GNN in Eq. 6.8. Finally, we calculate the percentage of models for which a given approach x dominates the other on a metric m in Eq. 6.9.

Finally, to compare the three different GNN solutions, i.e., DCMM(p_{dcmm}), DGI(p_{dgi}), DMoN(p_{dmon}), we consider the V_p, *cohesion* and *coupling* scores (see Eq. 7.1). Then, similar to Eq. 6.4, we get the best performing GNN using Eq. 7.2 and then we calculate the percentage of N models for which a given GNN approach is the best out of the three approaches in Eq. 7.3.

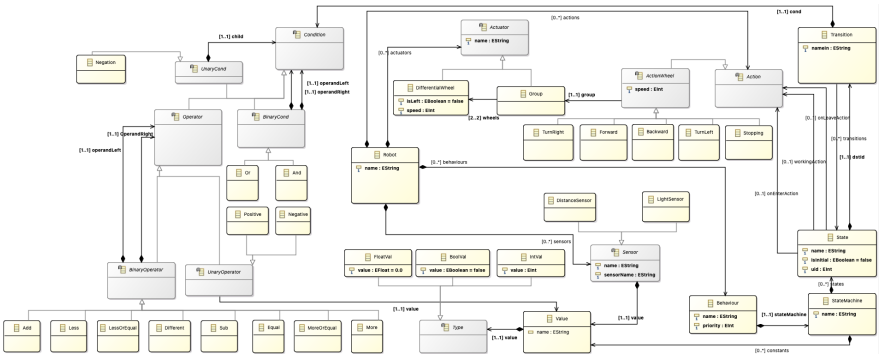
$$GNN_{max}(P, m) = \max(m(p_{\text{dcmm}}), m(p_{\text{dgi}}), m(p_{\text{dmon}})), \text{ m} \in \{V_p, cohesion, 1 - coupling\} \tag{7.1}$$

$$IGNN(P, m) = \begin{cases} \text{DCMM} & \text{if } p_{\text{dcmm}_m} = GNN_{max}(P, m), \\ \text{DGI} & \text{if } p_{\text{dgi}_m} = GNN_{max}(P, m), \\ \text{DMoN} & \text{if } p_{\text{dmon}_m} = GNN_{max}(P, m). \end{cases} \tag{7.2}$$

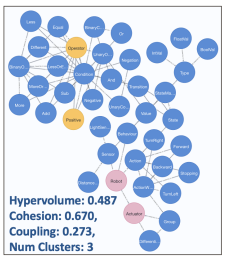
$$PGNN_{m,x} = \frac{\sum_{j=1}^{N} \mathbf{1}(IGNN(P_j, m) = \text{x})}{N} \quad \text{x} \in \{\text{DCMM, DGI, DMoN}\} \tag{7.3}$$

4.2 Results

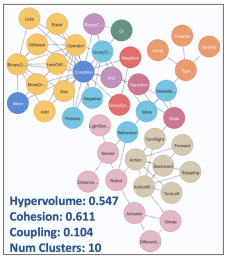
This comparative analysis sheds light on the efficacy of solutions provided by each approach, contributing to a deeper understanding of their strengths and limitations in addressing complex modularization tasks. Therefore, we respond to each RQ in the following. Fig. 3 shows the result of CMM using all four of the discussed approaches on an example input class diagram given in the Fig. 3a. The figure shows the hypervolume cohesion, coupling and the number of modules values. The figure shows DCMM with the highest hypervolume for the input class diagram. DGI provides better cohesion but worse coupling compared to DMoN.



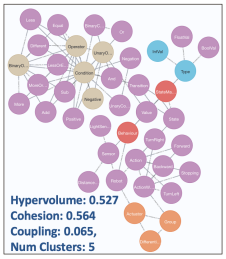
(a) A Robot Behaviour State Machine Class Diagram taken from the ModelSet [22]



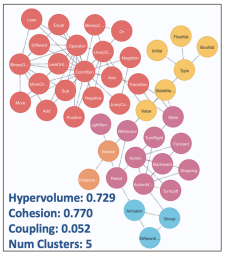
(b) ModuleER



(c) DGI



(d) DMoN



(e) DCMM

Fig. 3: Example modularization results computed by the evaluated approaches

Response to RQ1 - GA verus GNN. In this RQ, we provide a comparative analysis of GA approach with GNN. In case of GNN, we take the best performing GNN approach out of the three and compare it with GA. The results for RQ.1 are provided in Table 2 where each value shows the $P_{approach}$ scores across different metrics.

The overall results show that GA outperforms GNN-based approach for hypervolume and cohesion metrics, however GNN outperforms GA for coupling scores. In case of hypervolume, GA outperforms in more than two-third of the

Table 2: Overall Performance of GNN over GA

	PHV		$PM_{cohesion}$		$PM_{coupling}$	
Model Size	**GA**	**GNN**	**GA**	**GNN**	**GA**	**GNN**
Small	**63.63%**	36.36%	**68.18%**	31.82%	36.36%	**63.64%**
Medium	**79.80%**	20.20%	**94.23%**	5.77%	44.23%	**55.77%**
Large	**55.55%**	44.44%	**85.18%**	14.82%	48.14%	**51.86%**
Overall	**69.20%**	30.80%	**80.22%**	19.78%	40.68%	**59.32%**

models and provides higher cohesion scores in more than 80% of the models. We observe that while GNN approaches outperform GA in almost 60% of the cases for coupling, however, GA still provides a higher hypervolume, a metric which captures the combination of cohesion and coupling, for almost 70% of the cases. An explanation for this can relate to the fact that GA outperforms GNN in cohesion by a larger amount, so the contribution of even better (lower) coupling scores cannot balance out the contribution of higher cohesion to the hypervolume scores. Overall, this indicates that GA is more suitable for optimizing the cohesion scores, whereas GNN is suitable for optimizing coupling.

We further evaluate the effect of different model sizes on the comparative performance of GA and GNN. Interestingly, for hypervolume and cohesion, the performance of GNNs goes down for medium-sized models and then increases again for larger models. Furthermore, we see that the performance of GNNs decreases with larger model sizes for the coupling scores; however, it still provides better coupling scores than GA. Overall, it is interesting to note that GNN does provide better performance than GA in some of the metrics and setups. Moreover, esp. for a large models, we see that GNN produced modularizations of comparable quality or even of better quality with respect to coupling. Thus, we consider GNN-based CMM a promising alternative to GA which can be further investigated for improvements.

Response to RQ2 - DCMM versus DMoN and DGI. Now, we delve deeper into our DCMM's performance and compare it to the existing GNN models for CMM. We present an overall and a size-based performance comparison in Table 3. Each value in the table shows the percentage of models in a category for which a given GNN model outperforms the other two GNN approaches.

The results in Table 3 show that DCMM performs better than DGI and DMoN in most models for all three metrics. In case of hypervolume, DCMM outperforms DMoN and DGI in over 76% of the cases, outperforming DMoN (14.44%) and DGI (9.12%). This indicates DCMM's consistent superiority across all model sizes. In case of cohesion, DCMM scores highest 66.16%, compared to DMoN's 11.78% and DGI's 22.05%, showing its overall superiority in producing cohesive clusters. Finally, DCMM demonstrates overall dominance for coupling with 62.73%, compared to DMoN's 30.41% and DGI's 6.84%, confirming its consistent performance in reducing coupling.

We further evaluate the size-based comparative performance of the three GNN approaches. We see that in all the three metrics DCMM's performance improves in most cases or remains almost similar with an increase in model

Table 3: Comparative performance of DCMM with existing GNNs for CMM

	$PGNN_{V_p}$			$PGNN_{cohesion}$			$PGNN_{coupling}$		
Model Size	**DCMM**	**DMoN**	**DGI**	**DCMM**	**DMoN**	**DGI**	**DCMM**	**DMoN**	**DGI**
Small	**66.67%**	19.69%	13.63%	**55.30%**	15.90%	28.78%	**50.74%**	41.79%	7.46%
Medium	**84.62%**	10.57%	4.80%	**77.88%**	6.73%	15.38%	**73.52%**	20.58%	5.88%
Large	**92.59%**	3.70%	3.70%	**74.07%**	11.11%	14.81%	**81.48%**	11.11%	7.40%
Overall	**76.42%**	14.44%	9.12%	**66.16%**	11.78%	22.05%	**62.73%**	30.41%	6.84%

size. In case of hypervolume, the performance increases from DCMM being the best GNN model for 66.67% of the small models to DCMM being the best GNN model for 92.59% of the large models. In case of cohesion and coupling as well, DCMM still performs better than the other two GNNs for all the three model size categories. In case of hypervolume and coupling, DCMM performance shows an increase with model size. Overall, with this comparative evaluation we see, that our novel DCMM approach clearly outperforms the state-of-the-art for GNN-based graph clustering for conceptual models.

Based on our results we conclude as follows — *i*) GA-based approach is still a valuable approach for CMM, however, GNN-based approaches also turned out to be very useful alternatives for a large number of models providing better scores than GA for over 30% of the models and particularly for larger models providing better scores than GA for over 44% of the large sized models; *ii*) GNN-based approaches consistently perform better than GA for modularized solutions with lower coupling; and, finally, *iii*) the results underpin our novel DCMM model as a valuable GNN-based CMM approach, outperforming the state-of-the-art graph clustering models. DCMM further proves valuable as an alternative to GA-based approach, especially in case of minimizing coupling between modules.

5 Insights and Threats to Validity

In the following, we present further insights gained through the experiments conducted that yield additional responses to the outlined research questions.

Insights about RQ.1 - We observed that the model size affects the percentage of models for which either of GA and GNN perform better than. GA excels in achieving higher hypervolume and cohesion. This indicates that GA is more effective in maintaining tight, cohesive clusters. This is particularly evident in small and medium models where GA significantly outperforms GNN. However, GNN shows strengths in minimizing coupling, which is crucial for creating distinct, well-separated clusters and is therefore, advantageous for applications where minimizing inter-cluster dependencies is critical whereas GA may be better suited for tasks requiring tight clustering. GNN's performance is more competitive with GA for larger models, indicating its scalability and potential for handling more complex clustering tasks. Overall, our results show a significant potential in applying GNNs for conceptual model modularization; however there is no one size fits all solution and the choice between GA and GNN needs

to be guided by the specific requirements of the clustering task concerning a preference over cohesion, coupling, and a combination thereof. Our results do underpin GNN-based approaches as a suitable candidate for CMM, that can be investigated and improved further for a better performance by designing improved GNN architectures and loss functions that even better capture the graph clustering requirements of a conceptual modeler.

Insights about RQ.2 - The results of the comparative performance of DCMM and existing GNN models show, that DCMM consistently outperforms both DMoN and DGI across all model sizes and metrics. The performance of DCMM does not degrade with increasing model size. In fact, the performance gap between DCMM and the other models widens as the model size increases, showcasing its scalability. DCMM's high scores in cohesion and coupling metrics suggest that it effectively balances intra-cluster similarity and inter-cluster dissimilarity, which are crucial for high-quality clustering. This indicates that DCMM is a robust and scalable solution for GNN-based graph clustering in conceptual modeling. Furthermore, to the best of our knowledge, given that DCMM is the first GNN model for graph clustering applied in the domain of conceptual modeling, these results highlight DCMM's tailored effectiveness for this specific domain of conceptual modeling, addressing domain-specific challenges better than the more generic models like DMoN and DGI. The reason for a better DCMM performance can be attributed to the loss function used to train DCMM that directly aims to create solutions with high cohesion and low coupling, compared to the existing GNN solutions that maximize modularity that indirectly optimizes cohesion and coupling scores. This indicates that existing loss functions are not sufficient to produce good solutions and therefore, loss functions specific to CMM need to be developed.

We now elaborate on the threats to validity according to the widely accepted categories introduced by Wohlin et al. [38].

Conclusion validity regards issues that affect the ability to draw accurate conclusions about relations between the treatments and the outcome of an experiment. The selection of a point from a pareto set to compare with the GNN point falls under this category. There can be several other points in a pareto therefore the selection criteria of selecting the best trade-off point threatens the validity of our results. However, in this work, we focused on selecting the balanced solution and in our future work, we explore other criteria to compare a pareto set of a GA solution with a GNN solution.

Construct validity regards the ability to generalize the results of an experiment to the theory behind the experiment. We mitigated this threat by using a dataset of 263 UML models for our experiments. These 263 models are the filtered, non duplicated good quality models out of over 5000 models. However, our work still faces the construction validity threat for conceptual models of other modeling languages, which we aim to tackle as part of our future work.

Internal Validity regards the influences that can affect the independent variables with respect to causality. In our work, we set the configuration parameters for all the different approaches using a trial-and-error strategy. It is possible

that other parameter settings might yield different results. In fact, parameter tuning of search algorithms is still considered an open research challenge [7]. We mitigated this issue by doing a grid search over the different parameter values possible by trying different possible combinations of the involved configuration parameters.

External Validity regards the extent to which the research elements (subjects, artifacts, etc.) are representative of actual element. We mitigated this threat by using a Modelset dataset [22] of UML models used and peer reviewed in the literature which comprises manually created and not synthetically generated UML models.

6 Related Work

Next, we discuss related work on CMM. We consider works that treat the models as graphs and separate them into *i*) works that focus on the *structural properties* of the model in general, and *ii*) works that utilize some kind of GNN to drive the modularization. We do not cover the works that utilize the *conceptual model semantics* because such works are not within the scope of this research.

Structural Modularization. Bork et al. [8] propose ModuleER, a genetic algorithm-based modularization approach for Entity Relationship (ER) models using the graph structure information. We use ModulER as a baseline for GA-based CMM. Stuckenschmidt and Klein [33] propose a structure-based method clustering models based on the structure of the class hierarchy for real-world ontologies like SUMO and the NCI cancer ontology. In [34], Stuckenschmidt and Schlicht demonstrate that modularization based on structural properties alone produces meaningful modules that intuitively make sense.

Saruladha et al. [31] propose two neighbor-based structural proximity measures, namely TNSP and DNSP, to decompose ontologies into disjoint clusters. They consider concept pairs with common neighbors for clustering. Doran et al. [9] present an approach to ontology module extraction. Furthermore, for software systems, Andritsos et al. [3] present LIMBO which is a hierarchical clustering algorithm based on the minimization of information loss when merging two nodes in a cluster. The authors in [24, 25] present a hierarchical clustering-based weighted linkage clustering (WLC) approach. In particular, they merge entities together to form clusters, where two entities are merged together based on their types, relationships and attributes. Hence, the new feature vector correctly reflects relationships between the entities. Pourasghar et al. [29] present a modularization technique named GMA (Graph-based Modularization Algorithm). In their work, they propose several metrics to evaluate the quality of modularization solutions by utilizing structural features of the models.

GNN-based Modularization. GNNs have been recently used for graph clustering. MinCutPool [6] proposes a graph clustering approach by continuously

relaxing the normalized minimum cut problem and training a GNN to compute cluster assignments that minimize this objective. DMoN [27] extends spectral clustering and presents an unsupervised pooling method inspired by the modularity clustering measure. In [41], a GNN-based approach to encode node structural and community-aware representation using mutual information maximization [5] is presented, which captures local and global structural information. In our work, we compared our proposed DCMM with DMoN and DGI.

Synopsis. We summarize, that in case of structural modularization, most approaches related to conceptual model modularization approaches apply search-based techniques such as genetic algorithm techniques for graph structural clustering. We further note that GNN models are used for modularization but are not yet adapted to the specific requirements and applied to conceptual models. In this context, we presented our GNN-based graph modularization i.e., DCMM and provided a comparative evaluation against the existing GA and GNN approaches. We aim to combine the semantics with the graph structural modularization as part of our future work.

7 Conclusion and Future Work

This paper presented a new GNN-based conceptual model modularization approach and evaluated its performance on a dataset of UML class diagrams and compared the results to a GA-based approach and existing GNN-based graph clustering approaches. Our results position our deep conceptual model modularizer approach as a valuable GNN-based CMM approach, outperforming the state-of-the-art GA-based approach in over 30% of the cases overall and in almost 60% of the cases for the objective of minimizing coupling using the balanced trade-off solution from the Pareto set. Our DCMM approach outperformed the existing GNN models for CMM thereby showing DCMM's effectiveness in the specific application to conceptual models, addressing CMM challenges better than the more generic GNN models for graph clustering like DMoN and DGI.

In our current work, we did not include the model's metamodel and natural language label-based semantics and we only focused on UML class diagrams. In the future, we plan to extend our approach to several other modeling languages and utilize further sources of semantics to train GNN models for CMM.

Acknowledgements. Work partially funded by the Austrian Federal Ministry for Digital and Economic Affairs and the National Foundation for Research, Technology and Development (CDG).

References

1. Ali, S.J., Guizzardi, G., Bork, D.: Enabling representation learning in ontology-driven conceptual modeling using graph neural networks. In: Advanced Information Systems Engineering - 35th International Conference, CAiSE 2023. pp. 278–294. Springer (2023). https://doi.org/10.1007/978-3-031-34560-9_17

2. Ali, S.J., Laranjo, J.M., Bork, D.: A generic and customizable genetic algorithms-based conceptual model modularization framework. In: Enterprise Design, Operations, and Computing - 27th International Conference, EDOC 2023. pp. 39–57. Springer (2023). https://doi.org/10.1007/978-3-031-46587-1_3
3. Andritsos, P., Tzerpos, V.: Information-theoretic software clustering. IEEE Transactions on Software Engineering **31**(2), 150–165 (2005)
4. Behera, R.K., Naik, D., Rath, S.K., Dharavath, R.: Genetic algorithm-based community detection in large-scale social networks. Neural Computing and Applications **32**, 9649–9665 (2020)
5. Belghazi, M.I., Baratin, A., Rajeshwar, S., Ozair, S., Bengio, Y., Courville, A., Hjelm, D.: Mutual information neural estimation. In: International conference on machine learning. pp. 531–540. PMLR (2018)
6. Bianchi, F.M., Grattarola, D., Alippi, C.: Spectral clustering with graph neural networks for graph pooling. In: International conference on machine learning. pp. 874–883. PMLR (2020)
7. Bill, R., Fleck, M., Troya, J., Mayerhofer, T., Wimmer, M.: A local and global tour on momot. Software & Systems Modeling **18**, 1017–1046 (2019)
8. Bork, D., Garmendia, A., Wimmer, M.: Towards a multi-objective modularization approach for entity-relationship models. In: ER Forum, Demo and Poster 2020. pp. 45–58. CEUR (2020)
9. Doran, P., Tamma, V.A.M., Iannone, L.: Ontology module extraction for ontology reuse: an ontology engineering perspective. In: Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management, CIKM 2007. pp. 61–70. ACM (2007). https://doi.org/10.1145/1321440.1321451
10. Duong, C.T., Nguyen, T.T., Hoang, T.D., Yin, H., Weidlich, M., Nguyen, Q.V.H.: Deep mincut: Learning node embeddings by detecting communities. Pattern Recognition **134**, 109–126 (2023)
11. Eckle, K., Schmidt-Hieber, J.: A comparison of deep networks with relu activation function and linear spline-type methods. Neural Networks **110**, 232–242 (2019)
12. Figueiredo, G., Duchardt, A., Hedblom, M.M., Guizzardi, G.: Breaking into pieces: An ontological approach to conceptual model complexity management. In: 12th International Conference on Research Challenges in Information Science, RCIS 2018. pp. 1–10. IEEE (2018). https://doi.org/10.1109/RCIS.2018.8406642
13. Fortunato, S., Hric, D.: Community detection in networks: A user guide. Physics reports **659**, 1–44 (2016)
14. Garmendia, A., Bork, D., Eisenberg, M., do Nascimento Ferreira, T., Kessentini, M., Wimmer, M.: Leveraging artificial intelligence for model-based software analysis and design. In: Optimising the Software Development Process with Artificial Intelligence, pp. 93–117. Natural Computing Series, Springer (2023). https://doi.org/10.1007/978-981-19-9948-2_4
15. 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. pp. 855–864 (2016)
16. Hamilton, W.L., Ying, R., Leskovec, J.: Representation learning on graphs: Methods and applications. IEEE Data Eng. Bull. **40**(3), 52–74 (2017)
17. Hamilton, W.L., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: Advances in Neural Information Processing Systems 30: Neural Information Processing Systems 2017. pp. 1024–1034 (2017)
18. Ishibuchi, H., Nojima, Y., Doi, T.: Comparison between single-objective and multi-objective genetic algorithms: Performance comparison and performance measures.

In: 2006 IEEE International Conference on Evolutionary Computation. pp. 1143–1150. IEEE (2006)
19. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: 3rd International Conference on Learning Representations, ICLR (2015)
20. LeClair, A., Marinache, A., El Ghalayini, H., MacCaull, W., Khedri, R.: A review on ontology modularization techniques-a multi-dimensional perspective. IEEE Transactions on Knowledge and Data Engineering **35**(5), 4376–4394 (2022)
21. Liu, Z., Li, X., Peng, H., He, L., Philip, S.Y.: Heterogeneous similarity graph neural network on electronic health records. In: 2020 IEEE International Conference on Big Data (Big Data). pp. 1196–1205. IEEE (2020)
22. López, J.A.H., Izquierdo, J.L.C., Cuadrado, J.S.: Modelset: a dataset for machine learning in model-driven engineering. Softw. Syst. Model. **21**(3), 967–986 (2022). https://doi.org/10.1007/S10270-021-00929-3
23. Malekzadeh, M., Hajibabaee, P., Heidari, M., Zad, S., Uzuner, O., Jones, J.H.: Review of graph neural network in text classification. In: IEEE 12th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON). pp. 84–91. IEEE (2021)
24. Maqbool, O., Babri, H.: Hierarchical clustering for software architecture recovery. IEEE Transactions on Software Engineering **33**(11), 759–780 (2007)
25. Maqbool, O., Babri, H.A.: The weighted combined algorithm: A linkage algorithm for software clustering. In: European Conference on Software Maintenance and Reengineering. CSMR 2004. pp. 15–24. IEEE (2004)
26. Mirjalili, S., Mirjalili, S.: Genetic algorithm. Evolutionary algorithms and neural networks: theory and applications pp. 43–55 (2019)
27. Müller, E.: Graph clustering with graph neural networks. Journal of Machine Learning Research **24**, 1–21 (2023)
28. Newman, M.E.: Modularity and community structure in networks. Proceedings of the national academy of sciences **103**(23), 8577–8582 (2006)
29. Pourasghar, B., Izadkhah, H., Isazadeh, A., Lotfi, S.: A graph-based clustering algorithm for software systems modularization. Information and Software Technology **133**, 106469 (2021)
30. Proper, H.A., Guizzardi, G.: Modeling for enterprises; let's go to rome via rime. In: Clark, T., Zschaler, S., Barn, B., Sandkuhl, K. (eds.) Proceedings of the Forum at Practice of Enterprise Modeling 2022 (PoEM-Forum 2022) co-located with PoEM 2022. CEUR Workshop Proceedings, vol. 3327, pp. 4–15. CEUR-WS.org (2022)
31. Saruladha, K., Aghila, G., Sathiya, B.: Neighbour based structural proximity measures for ontology matching systems. In: Proceedings of the International Conference on Advances in Computing, Communications and Informatics. pp. 1079–1085 (2012)
32. Smajevic, M., Bork, D.: From conceptual models to knowledge graphs: a generic model transformation platform. In: International Conference on Model Driven Engineering Languages and Systems Companion. pp. 610–614. IEEE (2021)
33. Stuckenschmidt, H., Klein, M.: Structure-based partitioning of large concept hierarchies. In: The Semantic Web–ISWC 2004: Third International Semantic Web Conference, 2004. pp. 289–303. Springer (2004)
34. Stuckenschmidt, H., Schlicht, A.: Structure-based partitioning of large ontologies. Modular ontologies: Concepts, theories and techniques for knowledge modularization pp. 187–210 (2009)
35. Unger, R.: The genetic algorithm approach to protein structure prediction. Applications of Evolutionary Computation in Chemistry pp. 153–175 (2004)

36. Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y., et al.: Graph attention networks. stat **1050**(20), 10–48550 (2017)
37. Villegas Niño, A.: A filtering engine for large conceptual schemas. Doctoral thesis (2013)
38. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in Software Engineering, Second Edition. Springer (2024). https://doi.org/10.1007/978-3-662-69306-3
39. Wu, S., Sun, F., Zhang, W., Xie, X., Cui, B.: Graph neural networks in recommender systems: a survey. ACM Computing Surveys **55**(5), 1–37 (2022)
40. Xiong, J., Xiong, Z., Chen, K., Jiang, H., Zheng, M.: Graph neural networks for automated de novo drug design. Drug Discovery Today **26**(6), 1382–1393 (2021)
41. Zhang, T., Xiong, Y., Zhang, J., Zhang, Y., Jiao, Y., Zhu, Y.: Commdgi: Community detection oriented deep graph infomax. In: CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, 2020. pp. 1843–1852. ACM (2020). https://doi.org/10.1145/3340531.3412042
42. Zhao, W., Xu, G., Cui, Z., Luo, S., Long, C., Zhang, T.: Deep graph structural infomax. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 37, pp. 4920–4928 (2023)
43. Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., Sun, M.: Graph neural networks: A review of methods and applications. AI Open **1**, 57–81 (2020)