Localized recommendation in assembly modeling: employing GNNs for targeted part placement

Carola Lenzen, Wolfgang Reif

Angaben zur Veröffentlichung / Publication details:

Lenzen, Carola, and Wolfgang Reif. 2024. "Localized recommendation in assembly modeling: employing GNNs for targeted part placement." In 23rd International Conference on Machine Learning and Applications (ICMLA'24), December 18-20, 2024, Miami, Florida, USA, edited by M. Arif Wani, Plamen Angelov, Feng Luo, and Mitsunori Ogihara, 1804–9. Piscataway, NJ: IEEE. https://doi.org/10.1109/ICMLA61862.2024.00278.

Localized Recommendation in Assembly Modeling: Employing GNNs for Targeted Part Placement

1st Carola Lenzen
Insitute for Software & Systems Engineering
University of Augsburg
Augsburg, Germany
0000-0002-1678-6795

2nd Wolfgang Reif
Insitute for Software & Systems Engineering
University of Augsburg
Augsburg, Germany
0000-0002-4086-0043

***Abstract*—Assembly modeling in computer-aided design (CAD) refers to designing new products based on a collection of preexisting individual parts. To streamline this process, designers would benefit from recommendations for parts needed next, tailored to a specific extension point within the design. By their nature, assemblies can be represented as undirected graphs over parts. As parts of an assembly can be inserted in any order, we employ graph neural networks (GNNs) that are invariant to permutations. In terms of graph machine learning, the problem of localized part recommendation does not match traditional formulations such as link prediction or purely generative tasks that mostly focus on generating graphs with specific statistical properties on a macro-level. Instead, a novel approach is required that integrates the prediction of parts along with their connection to the existing graph at a specific node. In this problem setting, we investigate two distinct use cases: predicting new parts for a given partial design and user-selected extension point, as well as recommending both a new part and its extension point within the existing design. Our experiments indicate that our approaches significantly reduce the cognitive burden for designers: When recommending ten potential next parts, they included the needed part in up to 97.5% of cases for the first and both the part and its location in up to 92.0% for the second use case.**

***Index Terms*—Graph Machine Learning, Recommendation, Assembly Modeling**

I. Introduction

Assembly modeling constitutes a core task of computer-aided design (CAD). It defines an assembly as a collection of parts governed by pairwise constraints defined relative to the parts' topology, commonly known as *mates* or *mating conditions* [1]. Instead of creating designs from basic geometric shapes, assembly modeling entails the selection and placement of preexisting parts taken from common part catalogs. Choosing the appropriate next parts during design is a time-consuming challenge for design engineers due to the extensive amount of options they must sift through. Assuming a designer has previously only built very robust cabinets consisting of heavy doors and sturdy hinges, they may have to find the corresponding counterparts in completely different catalogs from other manufacturers when designing a lighter-weight variant – a situation commonly occurring, e.g., in contract manufacturing or special machine construction where designed products frequently vary. Additionally, companies typically use several part catalogs with proprietary formats, each providing different technical specifications and metadata (e.g., material properties), such that design engineers cannot easily compare similar parts from different manufacturers.

In preliminary work [2], we addressed part recommendation for a whole assembly, aiming at supporting experienced designers by reducing the selection of parts to be searched. In this approach, the part placement must be fully carried out by the designer. This may be acceptable for small assemblies, but becomes unwieldy for large ones with plenty of possible connection options. Furthermore, it is desirable to more specifically target the recommendations, for example to receive suitable suggestions only for a selected part of the assembly. This "filtering" on an existing part could be particularly helpful for inexperienced designers.

Consequently, this paper aims to support designers not only in the selection but also in the placement of the selected parts. The existing part of an assembly to which a recommended part is to be assembled is referred to below as the *extension point* or *extension node*. We investigate the localized recommendation of parts needed next in two use cases:

I. users specify the desired part of the design that should be extended and provide it as input to the model, and
II. this extension node of the existing design also needs to be predicted in addition to the next part type.

Within a CAD system, the first use case can be envisioned as follows: designers select an existing part to receive recommended parts to attach specifically to it instead of recommendations unrelated to what the designers currently focus on. This can support especially inexperienced design engineers who know which *kind* of part they want to connect at a specific location, but don't know which specific part fits best. On the other hand, if the designers don't even know that adding further parts to unused extension points (e.g., covering for screw holes) in their assembly can be beneficial for the actual application, they need recommendations for those together with the extension node – this motivates the latter use case.

For both use cases, we aim for an *interactive recommendation system* that provides auto-complete-like suggestions for parts rather than automatically completing or generating an assembly from scratch. The design engineer should be in full control of the system at all times, being free to choose whether to select from the part suggestions or search the catalogs themselves. This distinguishes the second use case in

particular from graph generation. Experienced designers can save time searching parts to insert; inexperienced designers can benefit from the knowledge inherent in the existing designs and localized recommendations. However, the task of part recommendation is inherently ambiguous as it depends on the designer's intent, who may want to connect two parts in an extraordinary way, e.g., to achieve innovative features for the product. Since such a combination apparently only occurs rarely – if at all – in previous designs, it would unlikely be recommended in a variable approach. Thus, we decided to recommend a fixed number of suggestions ordered by relevance, even if these might include misfit suggestions in obvious common cases.

We would like to emphasize that although the design sequence depends on the designer's preferences, this does not apply to the selection of the part types themselves. Determining a good part recommendation depends on the intended design itself, rather than simply relying on subjective preferences. We expect GNNs to be able to acquire the purpose of an assembly through the interplay of its parts via message passing.

II. Related Work

This work addresses two problems: the addition of new parts to an assembly and their location on the existing parts. The selection of new parts during the design process is commonly solved by designers using search engines based on keywords, shape similarity [3] or a combination of different search criteria [4], also referred to as assembly retrieval [5]. In contrast, we aim for a recommendation system where the designer does not have to specify any requirements or queries, but the next parts are inferred from parts already (partially) assembled. Transformer-based generative models have recently found application in CAD models, where they generate sequences of CAD-typical geometrical operations like extruding [6]. This method primarily addresses part design tasks, while our approach shifts the focus towards assessing the usage similarity of parts that frequently reoccur in assemblies.

The task of recommending parts shares similarities with graph generation, but approaches that generate the entire graph at once lack the suitability for interactive design. Autoregressive models, which predict nodes and edges incrementally, must accommodate external interactions, corresponding to a designer integrating self-selected parts searched in the catalogs. RNN-based methods like GRAN [7] or GraphRNN [8] do not incorporate permutation invariance by design, placing them at a disadvantage compared to GNNs. Moreover, they handle a single node type and assess only the final graph structure based on structural properties, neglecting intermediate states. However, the presence of different part types is a central property of assemblies. In order to capture the interplay between those parts, our approach must instead be capable of distinguishing them. The authors of [9] propose a generative approach learning a sequence of node and edge insertions. However, we focus on individual single-step part insertions, thus not needing recurrent structures that [9] employs.

The potential to support assembly modeling has already been recognized, resulting in more and more CAD datasets being published, e.g., [10], [11]. Jones et al. address a different aspect by AutoMate [12], predicting one of eight potential mate types for a connection between two parts. Instead of recommending parts to be connected, the approach relies on the designer to specify the two parts for mating. JoinABLe [11] also attempts to predict the relative geometric position of two parts in a CAD system (called joints instead of mates) using GNNs, but follows an entirely automated approach instead of involving users. Thus, neither approach matches the specific requirements of our use cases, but could be appended subsequent to ours to determine the mate type or specific position, respectively.

III. GNN-based Global Part Recommendation

We have initially introduced the problem of recommending parts for CAD assemblies using GNNs in [2], [13]. Both works focus on the recommendation of next parts in assembly modeling in a *global* context, i.e., without consideration or prediction of the localization. We employ a GNN which recommends a fixed-size ranked list of parts for a given partial assembly – however, designers have to know themselves where to place the recommendation to the current assembly. This paper draws on the main results of that work which is why we briefly revisit its approach in the following.

A. Assemblies as Graphs with Pretrained Embeddings

Following a data-driven approach, we analyze a set of assemblies over a given vocabulary of part types. Parts contained in an assembly are instantiations of these part types.

The governing assumption is that parts utilized together frequently possess a causal relationship that is encapsulated within the data. As a collection of parts that are connected to each other, assemblies can be naturally represented as undirected, unweighted graphs over parts, where the edges can be read off from physical connections or the mates specified in the CAD system. However, due to the fact that different part catalogs provide different kinds of metadata (such as the type of mate or geometric features), we leverage solely the parts and their connections for this approach.

Furthermore, the graphs do not contain any temporal information about their creation: Although every assembly has been created by a sequence of part insertion and connection operations, the resulting design only represents its final state without its creation history. Indeed, various valid sequences can lead to the same assembly: For example, if a part p_1 is connected to parts p_2 and p_3, it is neither distinguishable from nor relevant for the final assembly if the designer first connected p_1 to p_2, or to p_3 instead. This permutation invariance is an important property of the assembly data.

In a preliminary step, embeddings serving as node features for the parts are derived from the designs. The used method *comp2vec* is a generalization of word2vec [14] in the Skip-gram variant for sequences to general graph structures, where n-hop distant nodes of a center node instead of n-distant words

of a center word serve as the basis for creating the samples on which the model is trained. The resulting embeddings capture the function of a part in the design, so that parts with similar uses exhibit similar embedding vectors.

B. Generating Recommendation Instances from Assemblies

For the part recommendation task, instances of partial assemblies together with attachable parts are required. As the original creation sequence leading to the final assembly remains unknown and no sequence is preferable, instances are generated for every conceivable creation sequence in a self-supervised manner. The instances are constructed by recursively cutting off nodes from assembly graphs until a minimal graph size is reached. A node may only be removed if the resulting partial graph is still one connected component. The partial graphs, together with the part types of the cut-off nodes as labels, make up the instances. The learned part type embeddings serve as node features for the partial graphs.

C. Recommendation Models and Framing the Task

The learning problem is formulated as a graph classification problem where each target class corresponds to a part type of the vocabulary. During inference, scores are computed for each part type, subsequently ranked in descending order, and lastly the desired number of recommendations is chosen according to the ranking. As the instances have single-label targets, there are several instances with the same input graph, but different target part – corresponding to situations where multiple part types would be valid extensions to the particular graph at hand. This way, parts that are frequently assembled to the graph are higher up in the ranking. GNN flavors were selected as processing models since they are permutation invariant by design. They internally represent all possible creation sequences of an assembly's intermediate state in the same way.

In addition to the GNN models, we also evaluated the problem using a simple baseline model and an upper bound to frame it: In order to evaluate whether the recommendation models indeed provide context-based recommendations based on the current assembly state, the baseline model *Evergreen* always recommends the most frequent part types occurring in the training data, independently of the actual input graph.

The upper bound model in turn memorizes the entire test set and thus the mapping from input graph to target parts. As there are typically multiple target parts which can be connected to a given partial graph, it always recommends the most frequent part types for an instance. Note that due to the single-target nature of the instances, the upper bound model can never achieve 100% accuracy if it provides fewer recommendations than the number of different target part types for a graph. Therefore, the upper bound of the performance measure is parameterized in the number of recommendations.

D. Experiments and Findings

The proposed approach was evaluated on three collections of assemblies based on three disjoint sets of part types. The evaluation metric is the top-k rate, referring to the percentage of the target part type being in a model's top-k predictions. By design, it incorporates the order of the recommendations. GNNs (and GATs [15] in particular) turned out to be well suited for the investigated task. When comparing different part representations, the 100-dimensional *comp2vec* embedding delivered the best results.

IV. Use Cases of Localized Recommendation

Existing approaches to part recommendation for CAD assemblies do not incorporate *where* to assemble the recommended parts to the partial design. In the following, we consider two use cases: for the first, localization is provided in the input, the second aims for its prediction.

In order to frame them properly, we define the problem setting of assemblies as graphs over parts according to [13]: we assume a set of part types $\mathcal{T}$ which serves as the vocabulary upon which a dataset of N assemblies $\{\mathcal{A}_i\}_{i=1}^{N}$ is built. The parts present within an assembly are instantiations of these defined part types. Each assembly $\mathcal{A}$ delineates its constituent parts as nodes $\mathcal{N}(\mathcal{A})$, and information regarding their interconnections as edges $\mathcal{E}(\mathcal{A}) \subseteq \mathcal{N}(\mathcal{A}) \times \mathcal{N}(\mathcal{A})$. As the connections between the parts are both-sided, the edges are bidirectional. An assembly can contain several instances of the same part type, which can be distinguished by the associated nodes. Resuming, assemblies are depicted as undirected, unweighted graphs in which the nodes exhibit heterogeneity (representing various part types), while the edges remain homogeneous expressing only one type of connectivity within the design. To provide support during design, we analyze an *intermediate state* of an assembly, referred to as *partial assembly*.

A. Generating Localization Instances

For both localization use cases, we stick to instances with "single-label targets", meaning that the target is a *single* possible answer (part type or tuple consisting of extension point and part type) instead of a collection of all possible answers to a query input. In contrast, the predictions of the models are ranked lists. This representation meets the requirement of ranked recommendations according to which more frequently seen combinations should appear higher up due to higher scores. The base instances are generated in the same way for both use cases, but aggregated differently in a subsequent step. The generation procedure presented in [2] only needs to be adapted slightly: An assembly is recursively decomposed into triple-instances consisting of partial graph, extension point, and expected part type, i.e., $\langle \mathcal{A},\ n \in \mathcal{N}(\mathcal{A}),\ \tau \in \mathcal{T} \rangle$. When a node m is removed, the extension points originate from its connected nodes $\{n \in \mathcal{N}(\mathcal{A}) \mid (n, m) \in \mathcal{E}(\mathcal{A})\}$ – if there are multiple neighboring nodes, multiple instances are created. The target part type τ is always the part type of the cut-off node m. Duplicates of the triple instances that originate from the same assembly are discarded. For the first use case, the partial graph and extension point are grouped to form the input, whereas for the second use case extension node and expected part type are grouped as targets.

B. Use Case I: Recommending Parts for a Given Assembly Graph and Extension Point

For the use case of given localization information in the input, we train a discriminative model $P(\mathcal{T} \mid \mathcal{A},\ n \in \mathcal{N}(\mathcal{A}))$. We model the learning problem as *node classification*, predicting next part types for every node, where each part type corresponds to one class. Afterwards, we retain only the recommendations for the extension node, and rank the part types according to their scores. Finally, we select the desired number of suggestions.

Framing the Task: In order to frame this novel localization use case, we developed a tailored variant of the Evergreen model from [2] as our baseline, which also ignores the structure of the input graph, but additionally takes the extension node into account. Its recommendations for next parts are therefore based on the parts most frequently assembled to this particular extension point. Further, we adjust the upper bound model of [2] in a similar way, such that it memorizes the mapping from graph *and* extension node to the target parts from the test set. Consequently, it determines the best possible ranking of part types based on their frequency in the test set for an extension point in an assembly. Note that since the model only memorized the test set, it can never process input graphs not occurring in the test set, and thus only provides an estimation of the maximum achievable performance.

C. Use Case II: Recommending Parts and Their Extension Points for A Given Assembly Graph

For this use case, we need a *multi-task model* that outputs suggestions for new part types as well as their placement on the existing assembly, representing $P(\mathcal{T},\ \mathcal{N}(\mathcal{A}) \mid \mathcal{A})$. As already mentioned, the part type prediction can be modeled as a classification of fixed size out of the part types in the vocabulary $\mathcal{T}$, whereas the prediction of the extension point is a classification of *variable* size, depending on the number of nodes in the current assembly graph. In theory, the two sub-tasks of the multi-task model can be performed in arbitrary order. We modeled and investigated both variants, however, we only present the one which consistently outperformed the other variant during experiments: The first model performs global part recommendation according to $P(\mathcal{T} \mid \mathcal{A})$, and we sample a part type for the given input assembly as input for the second model, which then suggests a placement conditioned on this part type $\big(P(\mathcal{N}(\mathcal{A}) \mid \mathcal{A},\ \tau \in \mathcal{T})\big)$. The overall architecture is depicted in Fig. 1.

The first task can be implemented via graph classification, as was done in [2]. However, the second sub-model is responsible for placing the predicted part and thus requires both the current assembly graph and the part to extend it. At first sight, this appears similar to link prediction. However, instead of predicting new connections between the nodes of an existing graph as in conventional link prediction, we need to predict a connection of the *new* part to extend the existing graph. To do this, we insert a new node for this part, as well as edges from this node to all existing nodes in the graph. For each of the newly inserted edges, we train a GNN, the *edge score predictor*, to predict a scalar score depending on the features of the two nodes, which allows us to determine the most promising edge and thus the extension node. In our setting, we assume that not only the individual parts but also their combination in the assembly, i.e., the overall context, are relevant for determining the extension point. Therefore, the overall context should be included in the edge scoring of the second model. The graph classification model for the first sub-task predicts the part type for a partial assembly. This GNN has therefore already captured the context of the assembly through message passing and encoded it inside its hidden state. Thus, we build the edge score predictor upon the hidden state of the first model to capture the assembly context. However, as the new part was not yet included in the graph in the first step, the GNN for the first sub-task could not learn a hidden representation for this particular part. To generate a compatible representation, we employ an initialization MLP which learns from the pretrained part type embedding. In summary, the second sub-model consists of the initialization model followed by the edge score predictor.

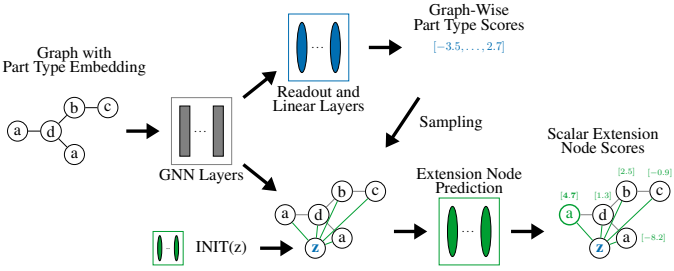

Fig. 1. High-level view of the model architecture for use case II: An assembly graph over part types $\mathcal{T} = \{a,b,c,d,z\}$ is initially enriched with part embeddings. A first model (blue) processes the graph context and predicts next parts to be added to the assembly. A second model (green) suggests the placement conditioned on the predicted part type. It operates on the hidden state of the first model for the preexisting parts and a suitable learned initialization as representation for the sampled part type.

Framing the Task: Similar to use case I, we reuse the Evergreen model and the upper bound from [2] and extend them to recommend parts together with their location. Instead of recommending the most frequent part types occurring in the training data regardless of the input graph, the simple baseline model examines all tuples of part and extension node in the training set: After filtering them for extension nodes which actually occur in the input graph (i.e., it contains a node with the same part type), it returns the most frequent remaining tuples. The upper bound model, however, still only needs to memorize the test set, as this now consists of input graphs and corresponding target parts together with their location. Consequently, it returns the most frequent target tuples for a graph from the test set.

V. Experimental Results

To test the applicability of our approaches, we evaluate them on the three assembly datasets of [2] – referred to as A, B and C – based on three disjoint parts type sets. Each dataset contains almost 12,000 assemblies, but they differ in

TABLE I
NUMBER OF RECOMMENDATION INSTANCES FOR ALL THREE DATASETS.

Dataset	# Part Types	Training	Validation	Test
A	1 930	133 271	42 531	43 420
B	3 099	1 531 257	474 233	491 492
C	1 924	1 050 993	378 712	399 763

the vocabulary size and structurally. Each dataset was split 60:20:20 into training, validation and test assemblies, which were afterwards transformed into the localization-aware triple-instances consisting of partial graph, extension point and expected part type, confer Table I.

Depending on the use case, the extension point serves as input or target. We adhere to the findings of [2] and use the pretrained 100-dimensional part type embedding, which greatly reduced the features and thus model parameters.

In terms of GNN architectures, we investigate GIN [16] and GraphSAGE [17] as well as the best performing architecture of [2], namely GAT [15]. Preliminary experiments revealed that neighborhood sampling in GraphSAGE is disadvantageous for our small assembly graphs, thus we skip that step for our experiments. Dropout is used as regularization technique.

The main evaluation metric remains the top-k rate, referring to the percentage of the target being in a model's top-k predictions, used for both target type and target tuples (part type and extension point). It evaluates both the correctness and ranking of the prediction. Assessing the performance of extension node prediction presents a greater level of complexity, as it depends on the graph size: Finding the right extension point for a large graph is much more difficult than for small graphs, as the model has to choose from more options. A better rank of the target on a larger graph should therefore lead to a bigger reward. Our goal is to evaluate the ranking relatively: the first position with three nodes should be rated the same as the first five position with 15 nodes. To achieve this, we include the graph size in the reciprocal rank metric [3] over an instance set D, which leads to the *mean weighted reciprocal rank (MWRR)* with different numerator:

$$\text{MWRR} = \frac{1}{|D|} \sum_{\langle \mathcal{A},\ n \in \mathcal{N}(\mathcal{A}),\ \tau \in \mathcal{T} \rangle \in D} \frac{|\mathcal{A}|}{\text{rank}(n)} \tag{1}$$

where the size of an assembly graph $|\mathcal{A}|$ is the number of its parts (i.e., nodes). The maximum value of this metric is no longer 1, but the number of nodes.

For training, we employed cross-entropy as loss function and chose early stopping as stopping criterion. During monitoring the loss values and performance measures for the training and validation sets over time, the following phenomenon emerged: While the validation loss was already increasing (indicating overfitting), the validation (and training) performance continued to increase. The reason for this seems to be the not exact correlation of loss and performance measure: While cross-entropy evaluates the difference of the float-valued softmaxed score $\hat{y} \in [0, 1]$ to the binary target value $y \in \{0, 1\}$, the top-k hit rate measures between binarized prediction $\tilde{y} \in \{0, 1\}$ and the binary target value y. Due to the logarithmic nature of cross-entropy, false "probability values" for classes are also weighted more heavily. As a consequence, we applied early stopping based on the performance metrics top-k hit rate and MWRR instead of the loss function. As evaluating performance metrics is significantly more computationally intensive, this greatly increased training time. For the experiments of the first use case, this was still acceptable; due to the subsequent models in the second use case, however, the metrics were only evaluated on 20% of the instances during training with a patience of 3 epochs.

TABLE II
RESULTS FOR PART RECOMMENDATION FOR GIVEN ASSEMBLY GRAPH AND EXTENSION POINT (USE CASE I): EVALUATION OF THE TOP-k RATE ON THE TEST SET FOR $k = 1, \ldots, 10$ RECOMMENDATIONS.

Dataset	Model \ k	1	3	5	10
A	Upper Bound	89.5%	98.8%	99.6%	99.9%
	GraphSAGE	**70.1%**	**86.3%**	**90.2%**	**93.8%**
	GIN	68.2%	83.6%	87.3%	90.7%
	GAT	58.0%	80.2%	85.7%	90.9%
	Baseline	36.2%	58.7%	69.3%	83.2%
B	Upper Bound	77.8%	98.3%	99.4%	99.9%
	GraphSAGE	**53.2%**	**75.6%**	**81.6%**	**87.5%**
	GIN	52.2%	73.8%	79.4%	85.6%
	GAT	50.7%	72.9%	79.4%	86.2%
	Baseline	33.1%	56.9%	66.0%	77.1%
C	Upper Bound	64.0%	94.2%	99.3%	99.9%
	GraphSAGE	50.0%	82.4%	**92.0%**	**97.5%**
	GIN	**54.5%**	**83.3%**	90.9%	96.2%
	GAT	42.0%	75.5%	86.6%	94.8%
	Baseline	36.7%	69.4%	81.7%	90.9%

A. Use Case I: Given Extension Point

Compared to the graph classification problem of [2], we generated half as many instances with the same input and therefore fewer correct target part types. This means that better hit rates can be expected for fewer recommendations.

Table II presents an overview of the performance of the models as well as the upper bound and baseline for the top-k rate. As our proposed models outperform the baseline model for all numbers of recommendations, they demonstrate that they are capable of incorporating the whole graph context and that this is beneficial for the task, however the margin fluctuates, showing both stronger (datasets A and B) and weaker (dataset C) outcomes across the datasets.

GraphSAGE proved to be the best performing architecture for all datasets, whereby GIN is a little ahead for dataset C for up to 3 recommendations. For datasets A and C, their is a notable performance difference between GAT, GIN and GraphSAGE for the top-k rate. On dataset B, however, all architectures are performing very similar with difference up to 2.5 percentage points.

B. Use Case II: Additional Prediction of the Extension Point

In preliminary experiments for this use case, GATs turned out to be inferior to models based on GIN and GraphSAGE

TABLE III
EVALUATION RESULTS FOR RECOMMENDING NEW PARTS ALONG WITH THEIR PLACEMENT FOR GIVEN ASSEMBLY GRAPH (USE CASE II): TOP-k RATE FOR $k = 1, \ldots, 10$ RECOMMENDATIONS FOR THE OVERALL MODEL.

Dataset	Model \ k	1	3	5	10
A	Upper Bound	59.4%	96.1%	98.3%	99.3%
	GraphSAGE	**44.9%**	**78.8%**	**84.9%**	**90.0%**
	GIN	44.5%	77.2%	82.6%	87.1%
	Baseline	12.7%	27.0%	37.3%	52.9%
B	Upper Bound	42.6%	86.8%	97.7%	99.2%
	GraphSAGE	23.5%	53.9%	**67.2%**	**76.9%**
	GIN	**25.2%**	**54.6%**	66.6%	75.8%
	Baseline	12.7%	29.4%	40.1%	54.9%
C	Upper Bound	35.6%	78.0%	95.6%	99.5%
	GraphSAGE	**27.3%**	**62.3%**	**80.1%**	**92.0%**
	GIN	27.1%	62.1%	79.9%	91.8%
	Baseline	7.6%	20.5%	36.4%	70.0%

– which is in line with the results of the first use case – and are therefore omitted for the sake of brevity. For simplicity, we use the same GNN architecture for both sub-models.

Table III presents the evaluation results for the second use case. The models' performance is again framed by upper bound and simple baseline. All GNN-based models clearly stand out from the baseline, which shows that a graph learning-based approach is a good fit for this problem, too. GraphSAGE turned out to perform slightly better than GINs on datasets A and C, which is consistent to our findings of the first considered use case. On dataset B, the performance of the GIN model is ahead of GraphSAGE merely for up to $k = 3$ recommendations, however, both models achieved a significantly lower top-k rate than for the other datasets. The reason behind this is both the higher number of part types and the fact that the assemblies from dataset B each contain more nodes than those from the other datasets (confer [2]), which in turn leads to more candidates for recommendations.

VI. CONCLUSION AND FUTURE WORK

We have addressed two use cases for localized part recommendation within an existing design for CAD assembly modeling: predicting new parts for a given design and user-selected extension point, as well as recommending both a new part and its extension point within the existing design. We proposed and compared sophisticated single- and multi-task modeling variants based on different GNN architectures. The experiments on three assembly datasets from [2] proved that our approach could effectively support CAD design engineers when integrated into an interactive recommendation system: The top-10 rate on three different datasets was consistently between 87.5% and 97.5% for the first use case, and between 76.9% and 92.0% for the second.

In the future, we want to integrate our approach into a real CAD system and evaluate if it is beneficial for the design of assemblies in terms of search effort and designer satisfaction. As our models are designed to provide a fixed number of recommendations, we expect some improper suggestions, especially in very obvious cases. As a consequence, a designer using the recommendation system could lose confidence in it due to such suggestions. On the other hand, they may then have to search for the required part in the catalogs themselves. Therefore, in future work, we want to investigate ways to predict a context-based number of recommendations.

REFERENCES

[1] M. Sarcar, K. M. Rao, and K. L. Narayan, *Computer aided design and manufacturing*. PHI Learning Pvt. Ltd., 2008.

[2] C. Gajek, A. Schiendorfer, and W. Reif, "A recommendation system for cad assembly modeling based on graph neural networks," in *Machine Learning and Knowledge Discovery in Databases*. Springer International Publishing, 2023, pp. 457–473.

[3] T. Funkhouser, M. Kazhdan, P. Shilane, P. Min, W. Kiefer, A. Tal, S. Rusinkiewicz, and D. Dobkin, "Modeling by example," *ACM transactions on graphics (TOG)*, vol. 23, no. 3, pp. 652–663, 2004.

[4] X. Chen, S. Gao, S. Guo, and J. Bai, "A flexible assembly retrieval approach for model reuse," *Computer-Aided Design*, vol. 44, no. 6, pp. 554–574, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010448512000243

[5] K. Lupinetti, J.-P. Pernot, M. Monti, and F. Giannini, "Content-based CAD assembly model retrieval: Survey and future challenges," *Computer-Aided Design*, vol. 113, pp. 62–81, 2019.

[6] R. Wu, C. Xiao, and C. Zheng, "DeepCAD: A Deep Generative Network for Computer-Aided Design Models," in *Proceedings of the International Conference on Computer Vision*, 2021, pp. 6772–6782.

[7] R. Liao, Y. Li, Y. Song, S. Wang, W. L. Hamilton, D. Duvenaud, R. Urtasun, and R. Zemel, *Efficient Graph Generation with Graph Recurrent Attention Networks*. Red Hook, NY, USA: Curran Associates Inc., 2019.

[8] J. You, R. Ying, X. Ren, W. Hamilton, and J. Leskovec, "GraphRNN: Generating realistic graphs with deep auto-regressive models," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 5708–5717.

[9] Y. Li, O. Vinyals, C. Dyer, R. Pascanu, and P. Battaglia, "Learning deep generative models of graphs," 2018. [Online]. Available: https://arxiv.org/abs/1803.03324

[10] S. Koch, A. Matveev, Z. Jiang, F. Williams, A. Artemov, E. Burnaev, M. Alexa, D. Zorin, and D. Panozzo, "Abc: A big cad model dataset for geometric deep learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 9601–9611.

[11] K. D. Willis, P. K. Jayaraman, H. Chu, Y. Tian, Y. Li, D. Grandi, A. Sanghi, L. Tran, J. G. Lambourne, A. Solar-Lezama *et al.*, "Joinable: Learning bottom-up assembly of parametric cad joints," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 15 849–15 860.

[12] B. Jones, D. Hildreth, D. Chen, I. Baran, V. G. Kim, and A. Schulz, "Automate: A dataset and learning approach for automatic mating of cad assemblies," *ACM Transactions on Graphics (TOG)*, vol. 40, no. 6, pp. 1–18, 2021.

[13] C. Lenzen, A. Schiendorfer, and W. Reif, "Graph Machine Learning for Assembly Modeling," in *LoG 2022: The First Learning on Graphs Conference, 9-12 December 2022, virtual event*, 2022.

[14] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," 2013. [Online]. Available: https://arxiv.org/abs/1301.3781

[15] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," 2018. [Online]. Available: https://arxiv.org/abs/1710.10903

[16] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" 2019. [Online]. Available: https://arxiv.org/abs/1810.00826

[17] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in Neural Information Processing Systems*, vol. 30, 2017.