Next Level Message-Passing with Hierarchical Support Graphs

Carlos Vonessen*, Florian Grötschla* (✉), and Roger Wattenhofer

ETH Zurich, Zurich, Switzerland {cvonessen,fgroetschla,wattenhofer}@ethz.ch

Abstract. Message-Passing Neural Networks (MPNNs) are extensively employed in graph learning tasks but suffer from limitations such as the restricted scope of information exchange, by being confined to neighboring nodes during each round of message passing. Various strategies have been proposed to address these limitations, including incorporating virtual nodes to facilitate global information exchange. In this study, we introduce the Hierarchical Support Graph (HSG), an extension of the virtual node concept created through recursive coarsening of the original graph. This approach provides a flexible framework for enhancing information flow in graphs, independent of the specific MPNN layers utilized. We present a theoretical analysis of HSGs, investigate their empirical performance, and demonstrate that HSGs can surpass other methods augmented with virtual nodes, achieving state-of-the-art results across multiple datasets.[1]

Keywords: Graph Neural Networks · Message Passing · Clustering.

1 Introduction

Graph Neural Networks (GNNs) have emerged as the leading method for learning on graph-structured data, through their ability to capture complex relationships between nodes. At the core of this field are Message-Passing Neural Networks (MPNNs), which iteratively exchange information between neighboring nodes through edges. Despite their success, MPNNs face significant limitations, primarily due to their restricted receptive field. This constraint hinders the ability of MPNNs to capture long-range dependencies effectively. To overcome the limited receptive field of MPNNs several methods have been proposed. A common strategy is to rewire the graph and enhance message-passing by creating additional edges. Another prominent approach is adding a virtual node that connects to all nodes in the graph, thus enabling global information exchange. This method has shown promise in extending the capabilities of MPNNs without significantly altering a graph's underlying structure.

In parallel, Graph Transformers (GTs), have gained traction by bypassing many of the inherent problems of MPNNs. GTs allow nodes to attend to all

[*] These authors contributed equally to this work

[1] Our implementation is available at https://github.com/carlosinator/support-graphs

other nodes simultaneously, thereby eliminating the locality constraint of message passing. As a result, GTs have demonstrated superior performance across various benchmarks. However, due to their asymptotic complexity, the scalability of GTs remains a challenge. GTs with full attention are computationally expensive and often impractical for large-scale graphs. Scalable variants of GTs, though more efficient, make other tradeoffs to stay performant.

We propose enhancing global information exchange in MPNNs and alleviating information bottlenecks by generalizing the concept of virtual nodes to general support structures we call Hierarchical Support Graphs (HSGs). In contrast to GTs, this approach only adds a small computational overhead and has the potential to be scaled to much larger graphs. The HSG is constructed through recursive coarsening of the original graph, providing a multi-level framework that enhances information exchange while maintaining computational efficiency. Our method integrates seamlessly with existing MPNN architectures, offering a scalable solution that bridges the gap between local and global information propagation. This study presents a comprehensive theoretical analysis of HSGs, evaluates their empirical performance, and demonstrates their superiority over existing virtual node-augmented methods. Our results show that HSGs achieve state-of-the-art performance on several benchmark datasets, highlighting their potential as a robust and scalable enhancement for graph learning tasks.

2 Related Work

Limitations of MPNNs. Multiple works have investigated the limitations of standard MPNNs. [1] and [31] show that MPNNs struggle to transmit information through bottlenecks and along long paths. Further work has shown that simple MPNN configurations are only as expressive as the 1-Weisfeiler-Leman graph isomorphism test [33]. [23] prove that on sufficiently dense graphs, simple GNNs with too many layers asymptotically lose their expressive power.

Augmented Message-Passing. Several techniques have been proposed that augment the graph used for message-passing to improve information flow. Virtual nodes (VNs), which connect to all nodes in a graph, are one of the most notable approaches [24]. There have also been efforts to prove that VNs can improve the expressiveness of GNNs [5,30]. Most recently, [27] show that VNs can reach state-of-the-art performance on some datasets. Instead of only adding edges to newly introduced nodes, one can add or remove edges. This is usually referred to as graph rewiring, and various edge selection methods exist, ranging from curvature-based analysis [31,2] to dynamic rewiring [14] and the combination of rewiring with VNs [25]. Hierarchical structures have also been used for domain-specific tasks [12]. In contrast to previous work, HSGs offer general support structures that do not change node updates and only require minimal pre- and post-processing adaptation. Furthermore, we do not need to adapt the synchronous message-passing rounds, simplifying the architecture and implementation.

Graph Transformers. Transformers on graphs have recently gained traction, as they outperform traditional MPNNs on many datasets. They further alleviate many shortcomings of MPNNs by allowing all nodes to attend to each other simultaneously. [26] present GraphGPS, a scalable architecture that employs sparse graph attention and message passing to combine the advantages of either architecture. [22] show that adding inductive biases to graph transformers eliminates the need for message-passing modules. Several architectures, such as GraphGPS [26] or Exphormer [28], only require sparse attention that scales linearly in the number of nodes and edges. Hierarchical approaches have also been introduced for Graph Transformers [35].

Graph Coarsenings. Several works introduce coarsening and hierarchical clustering for learning on graphs. [7] use graph clustering to identify well-connected subgraphs on large graphs. They show that it suffices to load these well-connected neighborhoods into memory rather than learning on the entire graph. [36,35] show that one can use hierarchical clustering to improve the scalability of GTs on large-scale graphs. Similarly, [16] create coarser representations of large-scale graphs and train a GNN only on the coarser representation, yielding message-passing with sublinear complexity. [34] recursively coarsen graphs using learned node embeddings to generate graph classification predictions. [13] use graph coarsening to compute representations for graph drawing efficiently. Further works also employ graph coarsening to improve learning on graphs [29,10,3].
In contrast to previous works, we propose a generalizable and flexible approach by integrating the HSG into the original graph. We further use hierarchical clustering to improve long-range information exchange in MPNNs rather than to decrease the asymptotic complexity of GTs as done in previous works [35,36,21].

3 Preliminaries

3.1 MPNNs

In this section, we introduce standard MPNNs and MPNNs augmented with a virtual node. We first state some elementary definitions.

Definition 1. *Given an undirected graph G with node-set $V(G)$ and edge-set $E(G)$, we define the following*

1. *Let $\mathcal{N}_G(v) := \{u \in V(G) \mid \{u, v\} \in E(G)\}$ be the neighborhood of a vertex $v \in V(G)$.*
2. *The vector $h_v^{(t)}$ denotes the hidden representation of a node after message-passing round t. $h_v^{(0)}$ is the initial node feature.*
3. *We denote* AGG *as an aggregation module that combines a multiset of vectors into one.*
4. *We denote* UPDATE *as a function that combines two vectors into one.*

When it is clear which graph is being referenced we shorten the notation to V and E.

Definition 2 (Message-Passing Layer). *Given a graph G we define a message-passing (MP) module as*

$$m_v^{(t+1)} = \text{AGG}(\{\{h_u^{(t)} \mid u \in \mathcal{N}_G(v)\}\}), \tag{1}$$

$$h_v^{(t+1)} = \text{UPDATE}(h_v^{(t)}, m_v^{(t+1)}), \tag{2}$$

where $\{\{\cdot\}\}$ denotes a multi-set.

An MPNN typically consists of an array of MP modules executed in series. Additional task-specific pre- and post-processing is often done with multi-layer perceptrons (MLP). A common MP module is the Graph Convolutional Network (GCN), introduced by [18]. One can extend the definition of MP modules to take edge features into account in the UPDATE step. A popular example of this is the GatedGCN architecture [4]. In node property prediction tasks, one can directly use a node's hidden representation after an array of MP modules as input to an MLP to generate a prediction. In graph property prediction tasks, a common approach is to pool all nodes with a global pooling function to produce one unified graph representation.

3.2 Graph Measures

We introduce several common graph measures that we will use to empirically assess the impact of HSG augmentation on graph topology.

The notion of effective resistance on graphs is derived from electric circuit analysis, where one considers all edges in a graph to be resistors. In this work, we consider all edges to have a resistance of 1. Intuitively two nodes will have low effective resistance if they are connected either through a few short paths or a larger amount of longer paths.
The following theorem defines the graph resistance for two nodes. While it is unknown who first proved it, a proof can be found in [19].

Theorem 1 (Effective Resistance). *Given a graph G, let $D \in \mathbb{R}^{|V|\times|V|}$ be the diagonal matrix containing the node degrees and $A \in \mathbb{R}^{|V|\times|V|}$ be the adjacency matrix. Then $L = D - A$ defines the graph Laplacian. Furthermore, L^+ denotes the Moore-Penrose pseudoinverse of L and $\mathbf{e}_x$ the all zero vector with a 1 at position x.*
Given a graph G one can compute the effective resistance R_{ab} between a pair of nodes $a, b \in V$ as

$$R_{ab} = (\mathbf{e}_a - \mathbf{e}_b)L^+(\mathbf{e}_a - \mathbf{e}_b)\ . \tag{3}$$

[19] also prove that the effective resistance is a distance measure and can thus be used to measure how well two nodes are connected. It is thus intuitive to use the effective resistance to model how well two nodes can communicate in an MPNN.

The commute time in a random walk on a graph is defined as the expected number of steps necessary to reach b from a and return.

Definition 3 (Hitting Time). *In a random walk $(X_t)_{t\in\mathbb{N}_0}$ on a graph G with $a,b \in V(G)$ and $X_0 = a$, the hitting time is the time required to reach b from a.*

$$H_{ab} = \inf\{t \in T \mid X_t = b\} \ . \tag{4}$$

From the definition of the hitting time, one can define the commute time as $C_{ab} = H_{ab} + H_{ba}$.

Theorem 2 (Resistance and Commute Time [6]). *In a graph G it holds that for any node pair $a,b \in V$*

$$\mathbb{E}[C_{ab}] = 2 \cdot |E| \cdot R_{ab} \tag{5}$$

It follows that the commute time scales proportionally to the effective resistance and the number of edges. We consider this measure in addition to the effective resistance, as we study the effect of the additional edges introduced by the HSG.

The connectivity of a node pair $u,v \in V$ is defined as the number of node disjoint paths between u and v. This is equivalent to the minimum number of nodes one must remove from the graph to disconnect u and v.

Definition 4 (Node Connectivity). *Let $\#cc$ denote the number of connected components of a graph, and $G[X]$ the subgraph induced by $X \subset V$. We define the graph node connectivity* (GNC) *as*

$$\text{GNC} := \min_{S\subset V}\{|S| \mid \#cc(G[V\backslash S]) > 1\} \ . \tag{6}$$

We further define the connectivity between two nodes $u,v \in V$ (with $u \neq v$) as

$$c_{uv} = \min\{|S| \mid \text{there exists no path between } u \text{ and } v \text{ in } G[V\backslash S]\} \ , \tag{7}$$

and the average node connectivity (ANC) *as the average over all c_{uv}.*

ANC is a useful measure to model the ability two nodes have to exchange information. In MPNNs, each node on a path only has a limited capacity to transmit information. We thus consider the number of node-disjoint paths between two nodes when assessing their ability to communicate.

4 Hierarchical Support Graphs

HSGs are created through recursive graph clustering. One takes the original graph and creates k node clusters by, for example, minimizing the number of intercluster edges. These clusters are then contracted to k super-nodes and multiple edges between clusters are contracted to single super-edges. We call this graph a support graph, as it mimics the structure of the original graph at a coarser level.

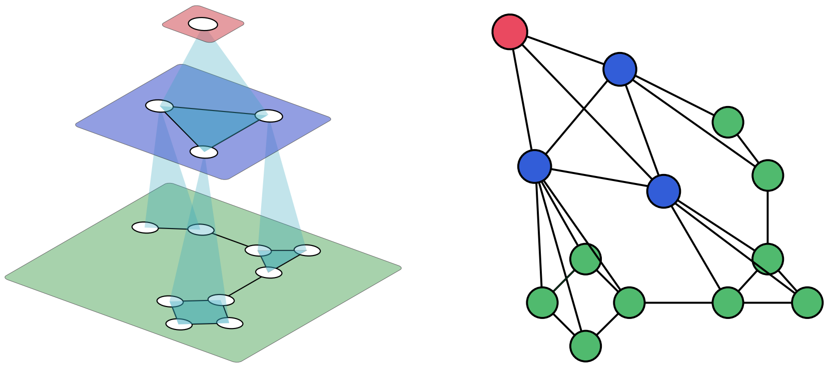

Fig. 1. Visualization of the graph coarsening procedure. First, the input graph G is recursively coarsened by clustering nodes in the same layer. Consequently, each node corresponds to a super-node representing the cluster. Finally, the super-nodes and edges are integrated as regular nodes into the graph.

We repeat this process recursively, creating a hierarchy of increasingly coarser graphs in each layer. We denote edges within one support graph layer as horizontal edges. As a final step, we connect all support hierarchies and the original graph by letting each node connect to its direct super-node. We denote these edges as vertical edges. We visualize the procedure in Figure 1.

Definition 5 (Graph Coarsening). *Let G be an input graph. A coarsening algorithm with strength $r \in [0,1]$ computes a partitioning $\{K_1, \dots, K_q\}$ of the node set $V(G)$ with $q = r \cdot n$. From this, one can construct a new graph H where each node $v_i \in H$ represents a node partition K_i in the original graph G. Furthermore, one adds an edge $\{v_i, v_j\} \in E(H)$ if there exists at least one edge between the nodes in K_i and K_j in G. We refer to this process as $H = f_r(G)$.*

Definition 6 (HSG Augmentation). *We apply a coarsening algorithm f_r to the input G, yielding the first support graph layer: $H^{(1)} = f_r(G)$. We repeat this recursively $Z-1$ more times, with $H^{(i+1)} = f_r(H^{(i)})$. The new graph, which is a combination of the original graph G and all support graph layers $(H^{(i)})_{i \in [Z]}$ is denoted as G^H.*
For convenience, we denote the set of all nodes as $\hat{V} := \bigcup_{i \in [Z]} V(H^{(i)}) \cup V(G)$.

1. *We define $\varphi : \hat{V} \to \hat{V}$ as the map of any node $v \in V$ to their direct super-node. The map is not defined for nodes in the highest layer $v \in H^{(Z)}$.*

2. *We define the node-set and edge-set of the HSG-augmented graph G^H as*

$$V(G^H) := \hat{V}, \tag{8}$$

$$E(G^H) = \bigcup_{i \in [Z]} E(H^{(i)}) \cup \{\{v, \varphi(v)\} \mid \forall\, v \in \hat{V}\} \cup E(G)\ . \tag{9}$$

This definition allows us to state some basic properties of HSG-augmented graphs.

Theorem 3 (Appendix A.1). *Given a graph G with n nodes and m edges. For any recursive coarsening method with a constant coarsening ratio $r \in [0,1]$, the total number of nodes in G^H is $\frac{n}{1-r}$. The diameter of G^H is bounded by $2\frac{\log n}{-\log r}$. Similarly, the number of edges is at most $\mathcal{O}(\frac{m \log n}{-\log r})$.*

Since the HSG is integrated into normal message-passing, each $H^{(i)}$ should preserve some properties of the original graph. For example, the node degree should not explode compared to the original nodes. In the following, we briefly examine this property and give bounds that coarsening methods should satisfy to fulfill this constraint.

Definition 7 (Cumulative Coarsening). *Let G be a graph of n vertices and m edges. In the i-th coarsening step, we define $r(i)$ as the node coarsening strength. We further study $c(i)$, which is the edge reduction factor.*
We denote $R(i) := \prod_{j \leq i} r(j)$ as the cumulative node reduction and $C(i) := \prod_{j \leq i} c(j)$ as the cumulative edge reduction compared to the original graph. Intuitively, $R(i) \cdot n$ is the number of nodes in the i-th hierarchical layer, and $C(i) \cdot m$ is the number of edges.

For many coarsening algorithms one can control $r(i)$, however $c(i)$, the factor by which the number of edges is reduced, commonly cannot be chosen.

Theorem 4 (Appendix A.2). *In a connected graph G with n vertices and m edges, any coarsening method that preserves the average node degree of horizontal and vertical edges $\frac{m+n}{n}$ must satisfy $\forall i \leq Z$*

$$r(i) = \Omega\left(\frac{n}{m}\right), \tag{10}$$

$$C(i) = \Theta(R(i)), \tag{11}$$

For $r(i) = \Theta(\frac{n}{m})$ the condition relaxes to $C(i) = \mathcal{O}(R(i))$.

This bound on $C(i)$ restricts the types of coarsening that one can apply to approximately preserve the original average node degree. We analyze this restriction using a random coarsening on the Erdős–Rényi Random Graph model and prove that many Erdős–Rényi Graphs will, in expectation, preserve the original node degree up to constant factors.

Theorem 5. *Given an Erdős–Rényi Graph $G(n,p)$, and a random coarsening with equal cluster sizes (± 1) and coarsening ratio $r = \Theta(\frac{n}{m+n})$. The node degree for horizontal edges remains unchanged up to constant factors compared to the previous layer's degree for*

$$p(n) = \Theta(n^{-\beta}), \tag{12}$$

where $\beta \in \{\frac{1}{2}\} \cup [1, \infty)$. For $\beta \in [0, \frac{1}{2})$ the horizontal node degree is asymptotically smaller than the previous node degree. The horizontal node degree diverges for all other values of β.

From this, one can directly restrict the permissible region of p.

Theorem 6. *An Erdős–Rényi Graph $G(n,p)$ which, when coarsened with $r = \Theta(\frac{n}{m+n})$, preserves the node degree of the previous layer up to constant factors can be restricted to*

$$p(n) = \Theta(n^{-\beta}), \tag{13}$$

with $\beta \in [0, \frac{1}{2}] \cup [1, \infty)$.

For $p \geq \frac{(1+\epsilon)\ln n}{n}$ an Erdős–Rényi Graph will be connected with high probability [9]. In practice, Theorem 6 covers the edge density of many relevant graphs.

4.1 Integrating HSGs into GNNs

We generate HSGs using the METIS partitioning algorithm [17]. In each hierarchical support layer, the algorithm attempts to minimize the number of inter-cluster edges while creating similar sized node clusters. After generating the HSG layers, we introduce the created nodes and edges as regular edges into the message-passing framework, thereby eliminating the need for any changes to MP modules. As a result, compared to several previous works that compute a separate update step for the virtual node [27,30,5], our approach requires no custom layers or updates specific to the HSG. This also allows us to minimize the changes one has to make to an existing MPNN pipeline. In a pre-processing step we compute the coarsening and augment the input graph as a pre-transformation. Using imputation, one can compute features for the virtual nodes and edges a priori (see Section 5.3). Finally, using simple node and edge indicators, one can completely recover the original topology and add specific embeddings to nodes and edges.

Adapted Graph Pooling. We further make use of the highest HSG layer to adapt the kind of graph pooling we apply on graph-level tasks. We reduce the global pooling introduced in Section 3.1 to only operate over the highest layer of HSG nodes $H^{(Z)}$, which then serves as input to the MLP prediction head.

5 Experiments

We evaluate our approach on PascalVOC-SP, COCO-SP, Peptides-func, and Peptides-struct from the Long Range Graph Benchmark (LRGB) [8] and on ogbg-molpcba from the Open Graph Benchmark (OGB) [15]. PascalVOC-SP and COCO-SP are vision datasets where each node represents a contracted area, or superpixel, from the original image. Peptides-func, -struct, and ogbg-molpcba are molecule datasets. Details on the model configurations are given in Appendix C.

5.1 How HSGs Change Graph Topology

In this Section, we investigate the effect of HSG augmentation on real-world graphs. Due to its high computational cost, we restrict our analysis to 100 random graphs from Peptides-func. We compare several graph measures for unmodified graphs, graphs augmented with a virtual node, and graphs augmented with an HSG. We give average node properties such as pair-wise distances and ANC for the original node pairs and ignore virtual nodes or HSG nodes.

Table 1. Graph statistics for a random subset of 100 graphs from Peptides-func. • represents the contraction into a single virtual node. For example, the configuration $(0.25, \bullet)$ denotes a coarsening method where one contracts to 25% of the original graph and then contracts the new layer to one supernode.

Augmentation	Coarsening	Avg nodes	Avg edges	Diameter	Avg shortest path	$\mathbb{E}[R_{ab}]$	$\mathbb{E}[C_{ab}]$	GNC	ANC
original	-	151.12	153.93	57.13	20.90	20.24	7740.86	1.00	1.02
virtual node	$(\bullet)$	152.12	305.05	2.00	1.98	0.90	550.03	2.00	2.02
METIS	$(0.25, \bullet)$	189.81	380.22	4.00	3.81	1.44	1111.70	2.00	2.28
	$(0.5, \bullet)$	207.99	417.92	4.00	3.84	1.40	1181.68	2.00	2.37
Random	$(0.25, \bullet)$	189.31	474.88	4.00	3.50	0.96	914.67	2.00	2.60
	$(0.5, \bullet)$	217.59	516.39	4.00	3.70	1.01	1050.26	2.00	2.60

Table 1 shows both the effective resistance and the expected commute time between all original input nodes. We observe a drastic reduction for both measures in the VN and HSG-augmented graphs compared to the original graphs. We further observe that the additional hierarchical layer slightly increases both measures compared to the VN-graphs. This gap is slightly larger for commute times, as HSG-augmented graphs introduce more edges than a single VN. Furthermore, there is a significant increase in ANC compared to the virtual node, both for METIS and random coarsenings. It should be noted that ANC takes into account that nodes have a limited capacity for transmitting information, which might be of interest for certain tasks. Finally, one can observe that random coarsenings exhibit lower commute time, effective resistance, and higher ANC compared to METIS coarsenings, likely because Peptides-func contains sparse graphs, and introducing a small number of random edges and nodes can aid connectivity.

5.2 Evaluation on Benchmark Datasets

Table 2 shows the empirical results on all tested datasets. On LRGB, we give the values of SAN and GPS as reported in [27]. The values of SAN on ogbg-molpcba are reported from [20]. We report the performance of DRew as given in [14] and of GRIT as reported in [22]. The performance of GatedGCN-VN and GCN-VN is given as reported in [27]. Finally, we use the baselines of GCN and GatedGCN as reported in [27]. We note that DRew and SAN are evaluated without the improved feature normalization introduced by [32]. We give the mean and standard deviation over 7 runs on ogbg-molpcba and over 4 runs over the LRGB datasets. Further details can be found in Appendix C.

Table 2. Test performance on four benchmarks from LRGB [8] and ogbg-molpcba [15]. The runs mark the **first**, **second**, and **third** highest performing models.

Model	PascalVOC-SP F1 Score ↑	COCO-SP F1 Score ↑	Peptides-func AP ↑	Peptides-struct MAE ↓	ogbg-molpcba AP ↑
GCN [27]	20.78 ± 0.31	13.38 ± 0.07	68.60 ± 0.50	**24.60 ± 0.07**	24.83 ± 0.37
GatedGCN [27]	38.80 ± 0.40	29.22 ± 0.18	67.65 ± 0.47	24.77 ± 0.09	**30.66 ± 0.13**
SAN [20,27]	32.30 ± 0.39	25.92 ± 1.58	64.39 ± 0.75	25.45 ± 0.12	27.65 ± 0.42
GPS [26,27]	**44.40 ± 0.65**	**38.84 ± 0.55**	65.35 ± 0.41	25.09 ± 0.14	29.07 ± 0.28
GRIT [22]	-	-	**69.88 ± 0.82**	**24.60 ± 0.12**	-
DRew [14]	33.14 ± 0.24	-	**71.50 ± 0.44**	25.36 ± 0.15	-
S^2GCN [11]	-	-	**73.11 ± 0.66**	**24.47 ± 0.32**	-
GatedGCN-VN [27]	**44.7 ± 1.37**	**32.44 ± 0.25**	68.23 ± 0.69	24.75 ± 0.18	**31.41 ± 0.19**
GCN-VN [27]	29.5 ± 0.58	20.72 ± 0.43	67.32 ± 0.66	25.05 ± 0.22	-
GatedGCN-HSG	**46.04 ± 0.59**	**35.35 ± 0.32**	68.66 ± 0.38	**24.21 ± 0.07**	**31.29 ± 0.20**
GCN-HSG	27.36 ± 0.49	18.89 ± 0.42	68.91 ± 0.29	24.79 ± 0.07	26.89 ± 0.25

Table 2 shows state-of-the-art performance on PascalVOC-SP and Peptides-struct. Furthermore, using GatedGCN, we outperform the virtual node implementations of [27]. We further highlight that our approach outperforms or matches GT approaches on PascalVOC-SP, Peptides-func, Peptides-struct, and ogbg-molpcba. On COCO-SP, GatedGCN-HSG shows significantly higher predictive performance than previous MPNN-based approaches and reduces the gap to GraphGPS. The graph-level tasks Peptides-struct and ogbg-molpcba are trained using the adapted graph pooling introduced in Section 4.1. Furthermore, these two datasets are only augmented with the smallest possible HSG, a single virtual node. This configuration probably works best because the average graph size in ogbg-molpcba is 26 [15]. We further note that the two vision datasets PascalVOC-SP and COCO-SP outperform previous virtual node implementations using GatedGCN, but fail to match the corresponding implementation using the GCN module. As we show in Table 4, masking HSG-edges with dummy features significantly improves predictive performance. As GatedGCN uses edge gates to distinguish neighbors, the dummy features could help mark the supernodes and -edges as such. In contrast, the lower performance of GCN compared to the virtual node implementations by [27] could be explained by GCN having no mechanism to gate edges in contrast to GatedGCN. [27] update the virtual node in a separate step, thus circumventing this issue.

5.3 Ablation studies

In this Section, we investigate multiple hyperparameters of HSGs to evaluate their effect on predictive performance. We first observe how different coarsening configurations impact predictive capabilities. We further evaluate the impact of feature imputation on model performance. Finally, we investigate the benefits of HSG-adapted graph pooling.

Coarsening Configurations. We investigate the effect of different coarsening configurations on a GCN-HSG model trained on Peptides-func. We show results in Table 3. We observe only minor differences, as many configurations lie within one standard deviation of each other. Here, the virtual node is outperformed by METIS but performs marginally better than random coarsening. However, as the model configurations in Appendix C indicate, in several cases, a virtual node integrated into message passing outperforms a larger METIS-coarsened HSG. We further observe a significant performance difference between METIS and random coarsening. Section 3.2 shows that random coarsenings have the highest ANC and lowest effective resistance among HSG graphs, yet one can observe a clear performance gap in Table 3. This may indicate that the graph structure, that is preserved through METIS coarsening is relevant to predictive performance.

Table 3. Average Precision of different coarsening configurations. The best-performing configuration is marked in **bold**. (•) corresponds to a virtual node integrated into normal message passing.

Coarsen config	METIS	Random
$(0.25, \bullet)$	68.71 ± 0.60	67.39 ± 0.79
$(0.5, \bullet)$	$\mathbf{68.91 \pm 0.29}$	67.92 ± 0.46
$(\bullet)$	68.19 ± 0.62	

Feature Imputation. We investigate the effect of different feature imputation methods on the created nodes and edges of an HSG. We test different approaches on PascalVOC-SP with GatedGCN-HSG and on Peptides-func with GCN-HSG. We show results in Table 4. **dummy-** denotes that nodes or edges only receive a dummy embedding denoting their hierarchical position in the graph. **impute-** denotes that the features of nodes or edges are imputed from their direct children. Based on what the features represent, we impute them using the mean in the vision dataset PascalVOC-SP and the mode in the molecule dataset Peptides-func.

GatedGCN-HSG trained on PascalVOC-SP profits from imputed nodes and dummy edges. The dummy edges especially may help the edge-gating mechanism in GatedGCN to distinguish real and hierarchical nodes. GCN-HSG trained on Peptides-func exhibits no performance difference for edge imputation as the GCN does not take edge features into account. This missing information could be compensated by clearly marking hierarchical nodes with dummy embeddings.

Table 4. Impact of super-node and -edge feature imputation for the PascalVOC-SP and Peptides-func datasets. The best-performing configuration is marked in **bold**.

Dataset	PascalVOC-SP		Peptides-func	
	impute-node	dummy-node	impute-node	dummy-node
impute-edge	45.06 ± 1.06	44.19 ± 0.37	66.31 ± 0.23	68.88 ± 0.86
dummy-edge	$\mathbf{46.04 \pm 0.59}$	43.81 ± 0.95	66.04 ± 1.04	$\mathbf{68.91 \pm 0.29}$

Adapting the Prediction Head. The HSG allows us to experiment with additional graph pooling methods on graph-level tasks. In Table 5, we compare the model performance when we apply global pooling and when we aggregate only the node features of the highest HSG level $H^{(Z)}$.
The results show that for ogbg-molpcba and Peptides-struct, only aggregating the highest layer of the HSG significantly improves performance. Compared to Peptides-func, both ogbg-molpcba and Peptides-struct only use a single virtual node as HSG augmentation, which means that global information must travel through the virtual node. Peptides-func is augmented with a two-layer HSG, thus information can also travel through lower layers.

Table 5. Performance on three graph-level datasets comparing predictive performance using global node pooling and top layer node pooling. The best configuration in each column is marked in **bold**.

Pred. Head	Peptides-func (AP $\uparrow$)	Peptides-struct (MAE $\downarrow$)	ogbg-molpcba (AP $\uparrow$)
global pooling	$\mathbf{68.91 \pm 0.29}$	24.75 ± 0.12	30.14 ± 0.24
top layer pooling	67.30 ± 0.17	$\mathbf{24.21 \pm 0.07}$	$\mathbf{31.29 \pm 0.20}$

6 Conclusion

In this paper, we introduced Hierarchical Support Graphs (HSGs), an extension of the virtual node concept designed to enhance message-passing in graph neural networks by alleviating information bottlenecks. Through a combination of empirical and theoretical analyses, we demonstrate the impact of HSGs on graph topology and information exchange. Our experiments on common benchmarking datasets further reveal that HSGs can achieve state-of-the-art performance, surpassing methods that rely solely on single virtual nodes, and achieving a new state-of-the-art on two datasets. In conclusion, HSGs offer a generalizable framework for improving message-passing neural networks that effectively extends the reach of information propagation without introducing substantial complexity.

Acknowledgments. The authors acknowledge Leo Widmer for his assistance in resolving the proofs in Appendix B.

References

1. Alon, U., Yahav, E.: On the bottleneck of graph neural networks and its practical implications (2021)
2. Arnaiz-Rodríguez, A., Begga, A., Escolano, F., Oliver, N.: Diffwire: Inductive graph rewiring via the lov\'asz bound. arXiv preprint arXiv:2206.07369 (2022)
3. Bergmeister, A., Martinkus, K., Perraudin, N., Wattenhofer, R.: Efficient and scalable graph generation through iterative local expansion (2024)
4. Bresson, X., Laurent, T.: Residual gated graph convnets (2018)
5. Cai, C., Hy, T.S., Yu, R., Wang, Y.: On the connection between mpnn and graph transformer (2023)
6. Chandra, A.K., Raghavan, P., Ruzzo, W.L., Smolensky, R.: The electrical resistance of a graph captures its commute and cover times. In: Proceedings of the twenty-first annual ACM symposium on Theory of computing. pp. 574–586 (1989)
7. Chiang, W.L., Liu, X., Si, S., Li, Y., Bengio, S., Hsieh, C.J.: Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '19, ACM (Jul 2019), http://dx.doi.org/10.1145/3292500.3330925
8. Dwivedi, V.P., Rampášek, L., Galkin, M., Parviz, A., Wolf, G., Luu, A.T., Beaini, D.: Long range graph benchmark (2023)
9. Erdős, P., Rényi, A.: On the Evolution of Random Graphs by (1960)
10. Fang, Y., Sun, S., Gan, Z., Pillai, R., Wang, S., Liu, J.: Hierarchical graph network for multi-hop question answering (2020)
11. Geisler, S., Kosmala, A., Herbst, D., Günnemann, S.: Spatio-spectral graph neural networks (2024)
12. Grötschla, F., Mathys, J.: Hierarchical graph structures for congestion and eta prediction. arXiv preprint arXiv:2211.11762 (2022)
13. Grötschla, F., Mathys, J., Veres, R., Wattenhofer, R.: Core-gd: A hierarchical framework for scalable graph visualization with gnns (2024)
14. Gutteridge, B., Dong, X., Bronstein, M., Giovanni, F.D.: Drew: Dynamically rewired message passing with delay (2023)
15. Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., Leskovec, J.: Open graph benchmark: Datasets for machine learning on graphs (2021)
16. Huang, Z., Zhang, S., Xi, C., Liu, T., Zhou, M.: Scaling up graph neural networks via graph coarsening (2021)
17. Karypis, G., Kumar, V.: A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM Journal on Scientific Computing **20**(1), 359–392 (1998), https://doi.org/10.1137/S1064827595287997
18. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks (2017)
19. Klein, D.J., Randić, M.: Resistance distance. Journal of Mathematical Chemistry **12**(1), 81–95 (Dec 1993). https://doi.org/10.1007/BF01164627
20. Kreuzer, D., Beaini, D., Hamilton, W.L., Létourneau, V., Tossou, P.: Rethinking graph transformers with spectral attention (2021)
21. Kuang, W., WANG, Z., Li, Y., Wei, Z., Ding, B.: Coarformer: Transformer for large graph via graph coarsening (2022), https://openreview.net/forum?id=fkjO_FKVzw
22. Ma, L., Lin, C., Lim, D., Romero-Soriano, A., Dokania, P.K., Coates, M., Torr, P., Lim, S.N.: Graph inductive biases in transformers without message passing (2023)

23. Oono, K., Suzuki, T.: Graph neural networks exponentially lose expressive power for node classification (2021)
24. Pham, T., Tran, T., Dam, H., Venkatesh, S.: Graph classification via deep learning with virtual nodes (2017)
25. Qian, C., Manolache, A., Morris, C., Niepert, M.: Probabilistic graph rewiring via virtual nodes. arXiv preprint arXiv:2405.17311 (2024)
26. Rampášek, L., Galkin, M., Dwivedi, V.P., Luu, A.T., Wolf, G., Beaini, D.: Recipe for a general, powerful, scalable graph transformer (2023)
27. Rosenbluth, E., Tönshoff, J., Ritzert, M., Kisin, B., Grohe, M.: Distinguished in uniform: Self attention vs. virtual nodes (2024)
28. Shirzad, H., Velingker, A., Venkatachalam, B., Sutherland, D.J., Sinop, A.K.: Exphormer: Sparse transformers for graphs (2023)
29. Sobolevsky, S.: Hierarchical graph neural networks (2021)
30. Southern, J., Giovanni, F.D., Bronstein, M., Lutzeyer, J.F.: Understanding virtual nodes: Oversmoothing, oversquashing, and node heterogeneity (2024)
31. Topping, J., Giovanni, F.D., Chamberlain, B.P., Dong, X., Bronstein, M.M.: Understanding over-squashing and bottlenecks on graphs via curvature (2022)
32. Tönshoff, J., Ritzert, M., Rosenbluth, E., Grohe, M.: Where did the gap go? reassessing the long-range graph benchmark (2023)
33. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? (2019)
34. Ying, R., You, J., Morris, C., Ren, X., Hamilton, W.L., Leskovec, J.: Hierarchical graph representation learning with differentiable pooling (2019)
35. Zhang, Z., Liu, Q., Hu, Q., Lee, C.K.: Hierarchical graph transformer with adaptive node sampling (2022)
36. Zhu, W., Wen, T., Song, G., Ma, X., Wang, L.: Hierarchical transformer for scalable graph learning (2023)

A Properties of Hierarchical Support Graphs

A.1 Proof of Theorem 3

Proof. We denote the original graph as G with n nodes and m edges. We further call G^H the HSG-augmented graph. Clearly, for a constant coarsening ratio $r \in [0, 1)$, the number of nodes is the limit of a geometric series, yielding at most $\frac{n}{1-r}$ nodes in G^H. For a constant coarsening ratio, it is easy to prove that for appropriate n the graph contains at most $\frac{\log n}{-\log r}$ layers, yielding a graph diameter of $2\frac{\log n}{-\log r}$. Based on the number of layers, the number of edges in each layer is at most m, yielding an upper bound $\mathcal{O}(\frac{m \log n}{-\log r})$.

A.2 Proof of Theorem 4

Proof. We define $r(i)$ to be the node reduction factor and $c(i)$ to be the edge reduction factor from layer $i-1$ to layer i. We write $R(i)$ and $C(i)$ as short form for the cumulative contraction until layer i The average node degree in one support layer $H^{(i)}$ can thus be expressed as

$$\frac{C(i)m + R(i-1)n + R(i)n}{R(i)n} = \frac{C(i)m}{R(i)n} + \frac{1}{r(i)} + 1 \tag{14}$$

For this expression to have order $\Theta(\frac{m}{n})$ one gets

$$r(i) = \Omega\left(\frac{n}{m}\right) \tag{15}$$

$$C(i) \leq \Theta(R(i)) \tag{16}$$

If $m = o(n)$, meaning for sufficiently large n the graph is disconnected, the bound on r becomes $r = \Omega(1)$.

B Random Coarsening of Erdős–Rényi Graphs

As shown in Appendix A.2 the conditions for a useful coarsening are restrictive in the coarsening ratio r and in the edge reduction factor c. In this section, we investigate to what extent these properties are satisfied for a random coarsening on an Erdős-Rényi Graph.

B.1 Proof of Theorem 5

Proof. We employ the common notation that $\sim$ denotes equality up to a multiplicative constant.
For clusters $K_1, \ldots, K_{r \cdot n}$ we calculate the expected number of intercluster edges between super-nodes for $i \neq j$ as

$$P[e(K_i, K_j)] = 1 - (1-p)^{\frac{1}{r^2}} \ . \tag{17}$$

Clearly, in the next layer one has created a new Erdős–Rényi Graph with $G(rn, 1-(1-p)^{\frac{1}{r^2}})$. We investigate the ratio of expected horizontal node degrees between the new layer $\mathbb{E}[d^{(1)}]$ and the original layer $\mathbb{E}[d^{(0)}]$

$$\frac{\mathbb{E}[d^{(1)}]}{\mathbb{E}[d^{(0)}]} \sim \frac{r(1-(1-p)^{\frac{1}{r^2}})}{p} \tag{18}$$

Given $p = n^{-1-\alpha}$ for $\alpha \geq 0$, $\mathbb{E}[m] = \binom{n}{2}p \sim n^{1-\alpha}$. Thus one finds the necessary coarsening ratio

$$r = \Theta\left(\frac{n}{n+m}\right) = \Theta\left(\frac{n}{n^{1-\alpha}+n}\right) = \Theta(1)\ . \tag{19}$$

As a result Equation 18 becomes

$$\frac{r(1-(1-p)^{\frac{1}{r^2}})}{p} \sim n^{1+\alpha}(1-(1-n^{-1-\alpha})^{\frac{1}{r^2}}) \sim 1 \tag{20}$$

Thus, sufficiently sparse graphs can be coarsened with a constant coarsening ratio.

For $p = n^{-1+\alpha}$ and $\alpha > 0$ one gets $r \sim n^{-\alpha}$ and finds

$$\frac{r(1-(1-p)^{\frac{1}{r^2}})}{p} \sim n^{1-2\alpha}\left(1-(1-n^{-1+\alpha})^{(n^{2\alpha})}\right) \tag{21}$$

For $\alpha = \frac{1}{2}$ the outer term becomes one, and the inner term converges to a constant. We give the following lower bound for the other cases:

$$n^{1-2\alpha}\left(1-(1-n^{-1+\alpha})^{(n^{2\alpha})}\right) \geq n^{1-2\alpha}\left(1-\exp\left(-n^{-1+3\alpha}\right)\right) \tag{22}$$

We make a case distinction on α and first consider the case $\alpha \in [\frac{1}{3}, \frac{1}{2})$. Since $\alpha \geq \frac{1}{3}$ the exponential term goes to zero, meaning the lower bound diverges.
We next consider $\alpha \in (0, \frac{1}{3})$. By applying L'Hôpital's rule one can show that the lower bound also diverges in this interval.
Finally, for $\frac{1}{2} < \alpha \leq 1$ the outer $n^{1-2\alpha}$ in Equation 21 converges to zero, while the inner probability term can be upper-bounded by 1.

Thus, the average horizontal node degree in the next layer remains unchanged up to constant factors only for $p = \Theta(n^{-\beta})$ and $\beta \in \{\frac{1}{2}\} \cup [1, \infty)$. For $\beta \in [0, \frac{1}{2})$ the horizontal node degree goes to zero.

B.2 Proof of Theorem 6

Proof. It follows directly from Theorem 5 that values of β for which the horizontal node degree diverges to infinity are not candidates for a node degree preserving coarsening. By assumption, the average node degree with respect to vertical edges must lie within $\Theta(\frac{n}{m+n})$. As a result, all values of β for which the node degree ratio as defined in Equation 18 does not diverge to infinity are permissible. This yields the original statement.

C Model Configurations

We base our implementation on the GraphGPS [26] codebase, and integrate the coarsening approach from [36], which itself is based on the ClusterGCN implementation [7]. We reuse several useful code snippets from the codebases of [32,27]. The models trained on the LRGB benchmark have less than 500K parameters, while models trained on ogbg-molpcba are not constrained to a parameter limit. In Tables 6 and 7 the row *HSG node feats* denotes whether the node features of the created super-nodes are imputed or set to dummy edges. The same holds for *HSG edge feats*. The *pooling* row only refers to graph-level tasks and denotes whether standard global pooling or top layer pooling is used as introduced in Section 4.1. *PE/SE* denotes which positional or structural encodings are used. LapPE denotes Laplacian Positional Encodings, and RWSE random walk structural encodings. All models are trained using a learning rate of 0.001.

Table 6. Hyperparameter configurations for GCN-HSG on all tested datasets

hyperparam.	PascalVOC-SP	COCO-SP	Peptides-func	Peptides-struct	ogbg-molpcba
dropout	0.2	0.1	0.1	0.1	0.4
num. layers	10	14	6	12	4
embed dim.	140	200	235	105	1024
pooling	-	-	global	top layer	top layer
head depth	2	1	3	2	1
coarsening	$(\bullet)$	$(0.05, \bullet)$	$(0.5, \bullet)$	$(\bullet)$	$(\bullet)$
HSG node feats	dummy	mean	dummy	dummy	dummy
HSG edge feats	dummy	mean	dummy	dummy	dummy
PE/SE	none	none	none	LapPE	RWSE
batch size	50	256	200	200	512
num. epochs	200	200	250	250	75
num. params	223K	475K	492K	164K	4.5M

Table 7. Hyperparameter configurations for GatedGCN-HSG on all tested datasets

hyperparam.	PascalVOC-SP	COCO-SP	Peptides-func	Peptides-struct	ogbg-molpcba
dropout	0.2	0.1	0.1	0.1	0.4
num. layers	10	12	10	9	6
embed dim.	95	85	80	100	1024
pooling	-	-	global	top layer	top layer
head depth	2	1	2	2	1
coarsening	$(0.05, \bullet)$	$(0.05, \bullet)$	$(\bullet)$	$(\bullet)$	$(\bullet)$
HSG nodes	mean	mean	dummy	dummy	dummy
HSG edges	dummy	mean	dummy	dummy	dummy
PE/SE	none	none	LapPE	LapPE	RWSE
batch size	50	256	200	200	512
num. epochs	200	200	250	250	75
num. params	473K	452K	348K	486K	31.8M