SC-GNN: A Communication-Efficient Semantic Compression for Distributed Training of GNNs

Jihe Wang
School of Computer Science
Northwestern Polytechnical University
Xi'an, China
wangjihe@nwpu.edu.cn

Ying Wu
Engineering Research Center of Embedded System Integration, Ministry of Education
Northwestern Polytechnical University
Xi'an, China
wuying39@mail.nwpu.edu.cn

Danghui Wang
National Engineering Laboratory for Integrated Aero-Space-Ground-Ocean Big Data Application Technology
Northwestern Polytechnical University
Xi'an, China
wangdh@nwpu.edu.cn

ABSTRACT

Training big graph neural networks (GNNs) in distributed systems is quite time-consuming mainly because of the ubiquitous aggregate operations that involve a large amount of cross-partition communication for collecting embeddings/gradients during the forward and backward propagation. To reduce the volume of the communication, some recent approaches focused on decaying each of connections via sampling, quantifying, or delaying until satisfactory trade-off are obtained between volume and accuracy. However, when applied to popular GNNs, those approaches are found to be bounded by a common volume/accuracy Pareto frontier which shows that the decaying for individual connection cannot further accelerate the aggregate of training. In this work, SC-GNN, a semantic compression of the cross-partition communication, is proposed to concentrate a group of connections as a high-level semantics and transmit to a target partition. Since carrying the overall intent of a group, the semantics can keep transferring the interactions, i.e., embeddings/gradients, between a pair of remote partitions until GNN models converge. In addition, a connection-pattern based differential optimization is proposed to further prune those weak connections, while guaranteeing the training accuracy. The results show that, for multi-field datasets, the compression rate of SC-GNN is 40.8 × higher than SOTA methods and the epoch time is reduced to 31.8% on average.

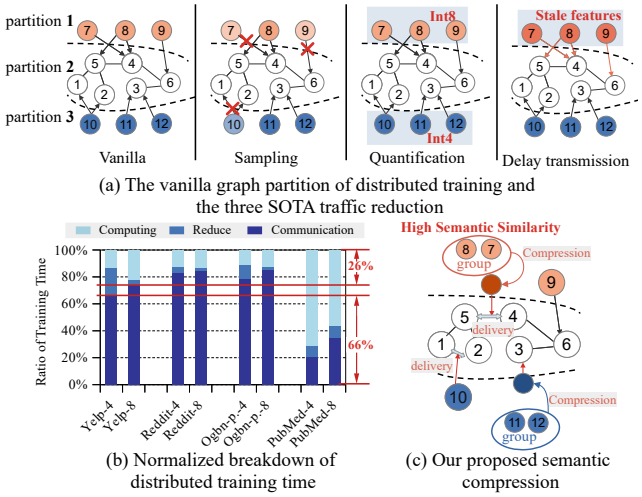
KEYWORDS

Graph, distributed training, graph neural network, communication compression.

ACM Reference Format:
Jihe Wang, Ying Wu, and Danghui Wang. 2024. SC-GNN: A Communication-Efficient Semantic Compression for Distributed Training of GNNs . In *61st ACM/IEEE Design Automation Conference (DAC '24), June 23–27, 2024, San Francisco, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3649329.3657383

1 INTRODUCTION

As the size of graph neural networks (GNNs) continues to grow up, it's increasingly difficult to accomplish a training on single computing device. Therefore, dividing a huge graph into many partitions to perform a distributed training has become a standard approach in both industry [20] and academia [7, 17, 1]. However, in vanilla GNNs, the ubiquitous and expensive **aggregate** operations [2] involve a large amount of cross-partition communication for collecting *embeddings/gradients* in the stage of forward and backward propagation, respectively. It has been confirmed that the slow aggregate sets a new bottleneck, called *aggregate-wall*, that occupies average 35 – 85% training time on the popular GNNs [2]. Resultantly, "*how to reduce the cross-partition communication while maintaining or even improving this training accuracy*" is the kernel challenge.



(a) The vanilla graph partition of distributed training and the three SOTA traffic reduction

(b) Normalized breakdown of distributed training time

(c) Our proposed semantic compression

Figure 1: Problem of SOTA distributed GNN training and our solution.

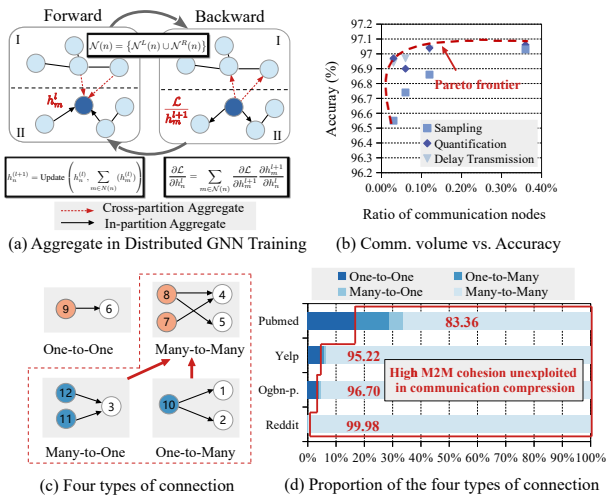
To reduce the communication, the recent studies were devoted to abridge the cross-partition edges, *a.k.a. structural connectivity*, via the three orthogonal dimensions in below (Fig. 1(a)). **1) *Sampling*** [16] randomly picks a fraction of edges for the aggregates according to a specified sample-rate; **2) *quantification*** [15] trades lower bit-width of *embeddings/gradients* for faster exchanging among partitions; and **3) *delayed transmission*** [12, 8] permitted some out-of-date *embeddings/gradients* to be held and involved into up-to-date aggregates, thereby the frequency of communication was reduced. Even though a "1-to-1" edge (O2O) can be successfully decayed with above efforts, it only occupies very small part of the overall (6.2%), leaving the vast majority of "many-to-many" edges (M2M) unexploited.

The methodology: This work aims to compress the multiple M2M-edges to one high-level "*semantical connectivity*" that only uses a few of communication to carry a large amount of *structural connectivity*. The holistic speedup by *semantical connectivity* mainly sources from the workload trade-off between compression and communication. As shown in Fig. 1(b), the current GNN training [16, 15, 12] spends 66% time in the cross-partition communication, however, the computation only consumes 26% time on average. The great temporal gap hints that it's profitable to involve extra lightweight expression to generate *semantical connectivity* as long as communication can be reduced substantially.

In this work, the above methodology motivates us to propose SC-GNN, a communication-efficient semantic compressing method that expands a novel dimension to optimize the aggregate in distributed training of GNNs (Fig. 1(c)). **First**, the compressing potentiality of *semantic connectivity* is revealed, based on which, "*semantic similarity*" is proposed to statically measure the cohesion of nodes: *only two nodes that are sufficiently high cohesive to each other can be divided into a semantic group*. **Second**, within each

DAC '24, June 23–27, 2024, San Francisco, CA, USA

ACM ISBN 979-8-4007-0601-1/24/06
https://doi.org/10.1145/3649329.3657383



(a) Aggregate in Distributed GNN Training (b) Comm. volume vs. Accuracy

(c) Four types of connection (d) Proportion of the four types of connection

Figure 2: A motivational analysis of the slow aggregate in distributed GNN training.

semantic group, semantics is generated via an ultra lightweight compression and, then, sent to its destination partition, where, the semantics is disassembled proportionally and aggregated into the associated nodes. **Third**, a full-batch training framework is built to further amplify the compressing benefit via a differential optimization, by which, any weak *structural connectivity* is selectively excluded, meanwhile the training accuracy can still be guaranteed.

The results show that, for multi-field datasets, the compression rate of SC-GNN is 40.8 × higher than SOTA methods and the epoch time is reduced to 31.8% on average. Our main contributions are listed in below:

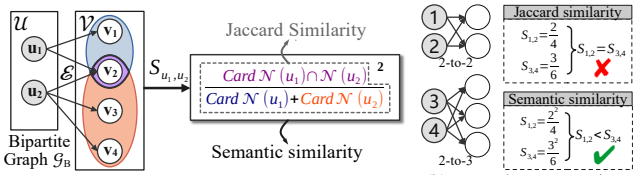
1. A semantic similarity is proposed to expand a distance space for efficient node-grouping.
2. The groups are approximated with local full-mapping to achieve semantic fusion and compression during exchanging embeddings/gradients.
3. A dedicated training framework is proposed to guarantee systematic speedup and training accuracy simultaneously.

2 BACKGROUND AND MOTIVATION

2.1 Aggregate-Wall of Distributed GNN Training

Aggregate-wall. A distributed training of GNN invokes the **aggregate** intensively to exchange embeddings (h_m^l) and gradients ($\frac{\partial \mathcal{L}}{\partial h_m^{l+1}}$) among partitions until its parameters converge (Fig. 2(a)). For large-scale GNNs [2, 19, 3], the cross-partition exchange forms a heavy burden of communication, a.k.a *aggregate-wall*, when those partitions are mapped to distributed devices, e.g., GPUs [17] or even clusters [20, 1]. For example, to train the Reddit dataset by a GraphSAGE [2] model, the communicating volume even achieves to 30MB and consumes 85% bursting bandwidth. To mitigate the speed degradation by aggregate-wall, many recent studies focused on decaying the *structural connectivity*, i.e., sampling [16], quantification [15], and delayed transmission [12], while ensuring that there is no significant drop in model accuracy.

After all, those decaying methods destroy training accuracy on individual dimensions so, inevitably, they seek to trade-off between accuracy and reduction, which restricts their earnings. As shown in Fig. 2(b), our study on the typical GNNs [16, 15, 12] reveals that all of the methods are restrained beneath a common Pareto frontier, where, both of *quantification* and *delay-transmission* methods can touch to the frontier but *sampling* still suffers a large gap. In addition, the compatibility problem among those decaying methods seems more difficult to resolve. For example, the *sampling* always trends to concentrate a mass of *embeddings/gradients* in "the lucky few"[16], however, the *quantification* cannot provide adequate bitwidth for their



(a) Visual explanations to semantic similarity (b) Jaccard *vs.* Semantic: an example

Figure 3: The explanations of semantic similarity $S(u_1, u_2)$ that extends from Jaccard similarity.

representations. Therefore, a new approach is needed to further break through the *aggregate-wall* and provide an elegant compatibility among the existing methods.

2.2 An Analysis of Semantical Cohesion

Logically, the connections between two partitions can be classified into two types: 1) one-to-one (O2O) and 2) many-to-many (M2M) that also covers one-to-many (O2M) and many-to-one (M2O) cases (Fig. 2(c)). With existing decaying methods, i.e., *sampling*, *quantification*, and *decayed transmission*, all edges of M2M connections are treated as individual O2O ones via redetermine their existence, bitwidth, timing respectively, resulting a common Pareto frontier. However, in real graph datasets, pure O2O connections are extremely rare cases and M2M covers even up to 99.98% cross-partition ones, which illustrates a high *cohesion* of both source and sink nodes, i.e. low inertia [13] (see Fig. 2(d)). It means that, by utilizing the high *cohesion*, it's possible to compress a group of *structural connectivity* as one *semantics* to further reduce communication sharply.

In order to create **node-groups** that can carry compressed semantics between partitions, there are two main issues to be addressed before a GNN training. **First**, a *semantic similarity* (*S*) between two same-side nodes should be defined and measured in source partitions (Sec. 3.1). **Then**, the similarity is treated as node distance to classify all nodes into semantic groups with high cohesion (Sec. 3.2 & 3.3).

3 SEMANTIC COMPRESSION

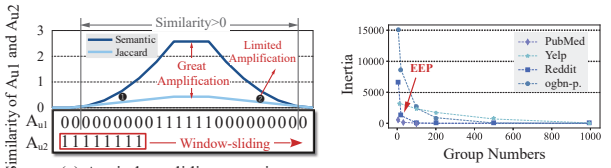
3.1 Semantic Similarity: An Effective Measurement between Nodes

The object of measurement. As shown in Fig. 3(a), a *directed bipartite graph* (DBG) $\mathcal{G}_B = (\mathcal{U}, \mathcal{V}, \mathcal{E}_{\mathcal{U}\to\mathcal{V}})$ can be constructed by only extracting the "boundary nodes" and their "cross-partition edges" of two partitions. Therefore, the semantic similarity of any two source-nodes $u_1, u_2 \in \mathcal{U}$ is governed by the following rule: *two source nodes (u_1, u_2) should have higher semantic similarity $S(u_1, u_2)$ if they had more common neighbors $Card(\mathcal{N}(u_1) \cap \mathcal{N}(u_2))$ in the sink part $\mathcal{V}$*, where $\mathcal{N}(u_i)$ is the neighbor set of u_i. A traditional measurement of above similarity, a.k.a. Jaccard similarity [10], divides the common neighbors by the sum of two individual mappings (Fig. 3(a)) so that "more common neighbors" and "less individual mappings" together lead to a higher similarity.

Defining semantic similarity. Although Jaccard similarity works well for general cases, unfortunately, it always fails in discerning similarity for those fully connected DBGs. As shown in Fig. 3(b), Jaccard method measures the same similarity for the "2-to-2" and "2-to-3" full-connected DBGs ($S_{1,2} = S_{3,4}$), which is both indistinguishable and counter-intuitive: clearly, the "2-to-3" case should have stronger cohesion and semantic associations via its richer cross-partition edges. To fix the problem, in this work, the numerator item $Card(\mathcal{N}(u_1) \cap \mathcal{N}(u_2))$ is replaced by its square, in which, the repeated item effectively distinguishes different full-connected DBGs (Fig. 3(b)). Therefore, our proposed semantic similarity is defined as:

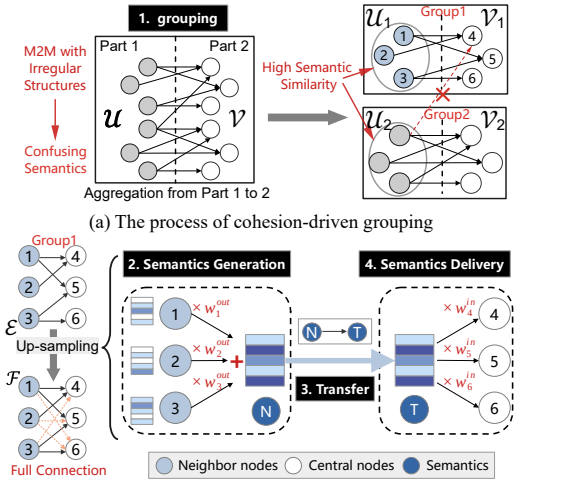
$$S(u_1, u_2) = \frac{Card(\mathcal{N}(u_1) \cap \mathcal{N}(u_2))^2}{Card(\mathcal{N}(u_1)) + Card(\mathcal{N}(u_2))} \tag{1}$$

Selective highlight of cohesion. A more significant benefit from the proposed semantic similarity (Eq. (1)) is that the quadratic item can selectively highlight the cohesion of two nodes: **1)** strong cohesion gains



(a) A window-sliding experiment for semantic and Jaccard similarity (b) The group number vs. cohesion

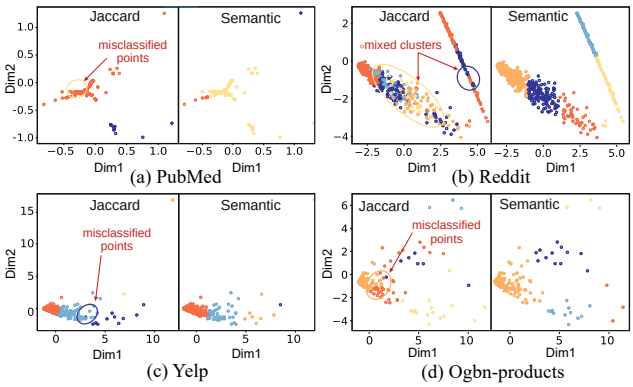
Figure 4: Performance of semantic similarity and group settings



(a) The process of cohesion-driven grouping

(b) The process of in-group up-sampling compression

Figure 5: The steps of grouped semantic compression.



(a) PubMed (b) Reddit (c) Yelp (d) Ogbn-products

Figure 6: Drop-dimensional distribution of nodes under four datasets.

super-linear amplification, **2)** weak cohesion sustains their influence, and **3)** non-cohesion is still excluded as the Jaccard method. Fig. 4(a) uses a window-sliding experiment to compare the cohesion by Jaccard and semantic similarities, respectively. Assuming A_{u1} and A_{u2} are two rows in the adjacent matrix of a DBG ($A^{\|\mathcal{U}\|\times\|\mathcal{V}\|}$) and the two types of their similarities are calculated during the A_{u1} valid-bits sliding from left to right. It shows that our semantic similarity dramatically amplifies the middle parts because of their high overlap of valid-bits, i.e. more common neighbors ($A_{u1} \cap A_{u2}$), in contrast, the amplification is very limited at the two ends due to the low overlap. It will be shown that the *cohesion-highlight* by our semantic similarity is quite friendly to improve the classifying accuracy in generating semantic groups (Sec. 3.2).

Vectorized acceleration of semantic similarity. The computation of semantic similarity can be accelerated easily by vectorizing its operations to enjoy the standard SIMD architectures, e.g., cuBLAS/VitisBLAS on GPGPUs. In Eq. (1), the AND-op of the molecular term is equivalent to the *inner products* of A_{u1} and A_{u2} and, in addition, the neighbor collection in the denominator can be obtained from a *shared* collection vector $\mathbf{C}_A$ that only accumulates the valid-bits for each line of $\mathbf{A}$. Therefore, the vectorized transformation of Eq. (1) is:

$$S(u1,u2) = \frac{(\mathbf{A}_{u1}\cdot\mathbf{A}_{u2}^T)^2}{\mathbf{C}_A^{u1}+\mathbf{C}_A^{u2}},\ \mathbf{C}_A = \mathbf{A}\cdot\mathbf{1}^{\|\mathcal{U}\|\times 1} \tag{2}$$

where, $\mathbf{C}_A^{u1}$ and $\mathbf{C}_A^{u2}$ fetch the corresponding items ($u1$ and $u2$) from the vector $\mathbf{C}_A$. Based on the SIMD acceleration, the slow set-operations in Eq. (1) are replaced by standard vector computations successfully, which is compatible well with typical parallel architectures in processing huge graphs.

The semantic similarity provides a measurement of node distance, by which, those nodes with strong similarity, i.e., high cohesion, could be grouped and compressed in communication to mitigate the aggregate-wall of the distributed training. To achieve this, this work proposes **1)** a cohesion-driven grouping method to cluster those high-cohesion nodes together (Sec. 3.2) and **2)** an approximated compression that uses a full-connected network to up-sample and fuse the edges within groups (Sec. 3.3). As a result, the large number of node-to-node transfers are regularized and reduced to a few group-to-group ones during the cross-partition exchange of *embeddings/gradients*.

3.2 Cohesion-driven Node Grouping

A round of aggregate includes many individual node-to-node transfers between a pair of partitions, a.k.a. $\mathcal{G}_B$, that consumes most of the training time. Therefore, as shown in Fig. 5(a), the source node set $\mathcal{U}$ is *statically* divided into many disjoint subsets, i.e., $\mathcal{U}_i$, in each of which, its nodes tend to have a high cohesion. After a $\mathcal{U}_i$ have been identified, the map from $\mathcal{U}_i$ to $\mathcal{V}_i$ can be extracted from $\mathcal{G}_B$ to form a high-cohesion group $g_i = (\mathcal{U}_i, \mathcal{V}_i, \mathcal{E}_{\mathcal{U}_i\to\mathcal{V}_i})$ as the basic compression unit in Sec. 3.3. As mentioned earlier, the semantic similarity $S(u_1,u_2)$ is designed to measure the semantic relatedness of a pair of nodes in $\mathcal{U}$, which expands a distance space to divide $\mathcal{U}$ to many $\mathcal{U}_i$ via a classical k-means grouping algorithm. However, "*group number*" is the only hyper-parameter unknown to the algorithm.

Picking group numbers. Indeed, the selection of group number is always a trade-off between cohesion and compression ratio. The traversal tests of group number (Fig. 4(b)) illustrate the trade-off, i.e., **first**, when the group number is small, the nodes within a larger group are less cohesive (higher k-means inertia), in which, *miss-classification* is likely to challenge the high-cohesion premise of grouping; **second**, too large number of groups, although enhancing the semantic cohesion of individual small groups (lower inertia), could lead to an excessive number of compression units, which fails to relieve the transmission pressure between partitions. To achieve the trade-off points of group number, "elbow equilibrium points" (EEPs) are adopted as the optimal balances [13], which is the point with the greatest curvatures. For example, the Reddit in Fig. 4(b) picks 20 as its group size since its EEP, i.e., the most distorted point, is found here.

By PCA dimensional reduction, the grouping results are visualized in Fig. 6 to illustrate the performances of semantic similarity in classification. The traditional Jaccard approaches (left-side) ubiquitously create the confusions, e.g., misclassified points and mixed clusters, on all datasets, which suggests that it is not sensitive enough to the semantic correlations between source nodes. In contrast, the proposed method (right-side) is able to distinguish the divergent inter-node semantics via the *highlighted cohesion*, creating the explicit groups. In next, the communicating compression is applied to each group.

3.3 In-group Up-sampling Compression

How to use one semantic connection to represent all the edges in a group? Topologically, connections within a group are non-regular, which

```
1 for u ∈ Neighbor nodes:
2   for e ∈ E(u,v):
3     Transmit h_u;
4 for v ∈ Central nodes:
5   for e ∈ E(u,v):
6     Receive h_u;
7   h_v = Aggregate(h_u);
```

(a) The traditional aggregation

```
1   W_u = D_u / ∑D_out;
2   h_g = ∑ (W_u · h_u);
3   Transmit h_g;
4   Receive h_g;
5   W_v = D_v / ∑D_in;
6   h_u = h_g · W_v;
7   h_v = Aggregate(h_u);
```

(b) The semantic aggregation

Figure 7: Semantic aggregate algorithm overcoming the vast transmission of traditional one.

means that, on the one hand, all of the nodes in a group have strong semantic relevance, and on the other hand, not all connections $(u, v) \in \mathcal{E}_{\mathcal{U}_i \to \mathcal{V}_i}$, where, $u \in \mathcal{U}_i$ and $v \in \mathcal{V}_i$. For example, in Fig. 5, the group 1 is constructed by the nodes $1 \sim 6$ with high cohesion but not all connections are intrinsic ones in the group, e.g., $(1, 6)$, $(2, 5)$, $(2, 6)$, and $(3, 4)$. These vacant edges hint that they exchange zero information during distributed training. Therefore, in this work, an up-sampling method is used to compress all edges of a group to one with following two steps (Fig. 5(b)). **First**, non-full-map $\mathcal{E}_{\mathcal{U}_i \to \mathcal{V}_i}$ is up-sampled to a full-map $\mathcal{F}_{\mathcal{U}_i \to \mathcal{V}_i}$, where $(u, v) \in \mathcal{F}_{\mathcal{U}_i \to \mathcal{V}_i}$ for all (u, v). **Second**, the nodes in $\mathcal{F}_{\mathcal{U}_i \to \mathcal{V}_i}$ are weighted via their connecting strength so that the weighted sum of the full-map connections is used as an effective compressed edge to be transferred in a group. Indeed, this method is to use a full-map $\mathcal{F}_{\mathcal{U}_i \to \mathcal{V}_i}$ to approximate and replace the original map $\mathcal{E}_{\mathcal{U}_i \to \mathcal{V}_i}$, which provide a chance to regularly compress the connections of a group to one semantic edge.

As shown in Fig. 7(a), the vanilla solutions set up message transmission for each connection in $\mathcal{E}_{\mathcal{U}_i \to \mathcal{V}_i}$, leading to an overwhelming transmission pressure. However, the proposed compressing method in Fig. 7(b) propagates a compressed message with following three steps:

- Line 1-2: Fusion and compression of all individual messages into a semantic message (h_g) at source partitions according to their output-weights (w_u).
- Line 3-4: The message (h_g) is transmitted out of the source partition and received at target partitions.
- Line 5-7: the semantic message is disassembled according to their input-weights (w_v) and updated into h_v.

The advantage of the new approach is that all messages (h_u) within a group are compressed into a single semantic message (h_g) for transmission, from which, the compression rates are proportional to the number of connections $\| \mathcal{E}_{\mathcal{U}_i \to \mathcal{V}_i} \|$, dramatically saving the communication.

How to decide the weights of the nodes in $\mathcal{F}_{\mathcal{U}_i \to \mathcal{V}_i}$? The weight of a node in DBGs has been well-studied in the field of social networking services. SALSA algorithm [6], as a popular weight-determining method, uses the degree of nodes to label their connecting strength. However, its calculation involves whole graph degree counting, which is ill-suited for the distributed training. Therefore, in this work, a local SALSA variant, called L-SALSA, is proposed to assign the node weights only within a group. For a source node $u \in \mathcal{U}_i$, $\mathcal{D}(u)$ is its degree in its group. The output-weight is the percentage of connections in the group, statically generated by $w(u) = \frac{\mathcal{D}(u)}{\|\mathcal{E}_{\mathcal{U}_i \to \mathcal{V}_i}\|}$. Similarly, for a target node $v \in \mathcal{V}_i$, its input-weight $w(v) = \frac{\mathcal{D}(v)}{\|\mathcal{E}_{\mathcal{U}_i \to \mathcal{V}_i}\|}$. Finally, the full-mapping approximation ($\mathcal{F}_{\mathcal{U}_i \to \mathcal{V}_i}$) uses the weights to compress the messages, i.e., the compression in Fig. 7(b) refers the L-SALSA weights to assemble (Line 2) and deliver (Line 6) the messages (h_g).

4 FRAMEWORK OF SC-GNN TRAINING

Our SC-GNN extends the standard GNNs training [2, 14, 5] by adding the semantic-grouping steps between graph-partition and node-update steps, see Fig. 8.

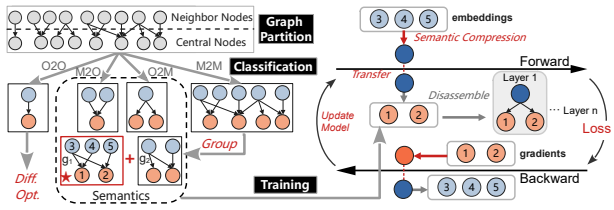


Figure 8: The framework of SC-GNN distributed training.

First, to deploy a big graph on distributed devices, a partition algorithm [4] cuts the graph into many partitions so that they can be out-sourced to devices. There are three popular partition algorithms adopted in recent GNN training, i.e., *edge-cut* minimisation [20, 8], *node-cut* minimisation [16], and *random-cut* [7]. Our results in Sec. 5.5 show that the node-cut provides better compatibility with our semantic compression since it always ignores the large number of edges linked to the same node, which is algorithmically isomorphic to our approximating compression. Therefore, in most of the experiments, *node-cut* has been used as the default partition algorithm.

Second, as Fig. 2(c), all of the connections between two partitions are classified into the four types (M2M, O2M, M2O, and O2O) and processed separately.

- **M2M**: as shown in Fig. 2(d), most of the connections are categorized as this type and they are necessary to be further grouped with the proposed semantic similarity in Sec. 3.1. Without the grouping, the semantic compression could blur a large set of connections, i.e., bad cohesion, and slow down the convergence of models.
- **O2M&M2O**: these types naturally form their groups since they are themselves *full-mapping* DBGs and can be processed directly by the code in Fig. 7(b). Therefore, they are trivially treated as semantic groups without any process.
- **O2O**: since most of the connections in Fig. 2(d) are dramatically compressed as semantic message, the major traffic comes from O2O now. A differential optimizing study in Sec. 5.3 illustrates that, removing all O2O connections from graph can further reduce the traffic loads with the ignorable loss of accuracy.

Third, the groups from M2M, O2M, and M2O are used for the semantic compression between two partitions, which replaces the individual exchange of embeddings/gradients (right-side of Fig. 8).

5 EXPERIMENTAL RESULTS

As the baselines, sampling [16], quantification [15], and delay [12] are compared to our semantic compression. In addition, four typical datasets expand a comprehensive test-scene, in which, 1) Reddit [2] and Yelp [3] feature with high/low-density graphs respectively, and 2) Ogbn-products [19] and PubMed [11] with strong/weak generalization. The settings of GNNs inherit that of BNS-GCN [16] and the algorithms are developed with DGL-v0.9.0 and PyTorch distributed packages [18], in which, gloo-backend [9] burdens the aggregating communication. The distributed platform consists of two AMD EPYC CPUs and four NVIDIA GeForce RTX 4090 GPUs.

5.1 Compression Ratio of Aggregation

Fig. 9 illustrates the normalized traffic volumes of the four methods, in which, our semantic method shows the varying degrees of compression ratio (40.8× on average). For Reddit, our traffic volumes are compressed to 0.72% of the other three ones and the ratio is further reduced to 0.38% if compared to the sampling methods. The great advantages source from our mechanism that approximately fuses all connections as one traffic, which is particularly effective for high-density graph datasets, e.g. the average degree of the Reddit dataset is 25.1×, 18.9×, and 108.7× higher than that of three other datasets, respectively. For Yelp and Ogbn-products, the lifts of compression ratio are 45.5% and 32.4% respectively, which sources from their medium graph densities. For PubMed, a typical low-density dataset,

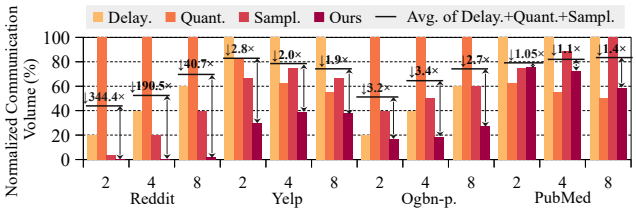


Figure 9: The improvement of the proposed semantic compression over sampling, quantification, and delay methods.

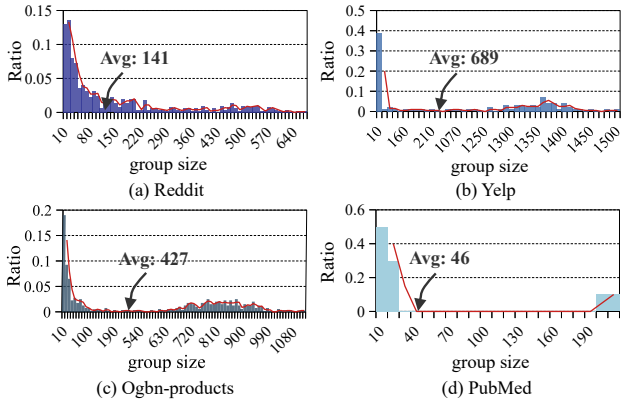


Figure 10: The distributions of group size and their means.

the quantification also demonstrates the strong competitiveness because it is handy to achieve 4 times of reduction by quantifying bit-width from 32-bit to 8-bit but the ultra-low density doesn't leave much room for our semantic compression in its sparse groups.

Fig. 10 reveals the underlying reasons why our compression ratios are significant. In theory, the ratio mainly comes from the M2M-connections that construct large groups, in each of which, many connections are reduced to one. By computing the average of group sizes, huge M2M-connections are found in Reddit (141 : 1), Yelp (689 : 1), Ogbn-products (427 : 1), and PubMed (46 : 1). Those great group sizes guarantee the reliable compression ratio as long as those M2M-connections go mainstream, which is a fact in the distributed GNN training (see Fig. 2(d)).

5.2 Speedup of Training Epoch

To demonstrate the systematic latency of the traffic-reducing methods, in this section, the communication of the three baselines is scaled to that of our semantic compression so that all methods are applying the same traffic pressure to interconnection. For instance, the sampling, quantification, and delay methods scale rates, bit-width, and frequency, respectively, to adjust traffic loads. By excluding above traffic factors, the advantages of our semantic method become evident in Table 1. All of the training epochs achieve their minimal latency when using our semantics (31.77% on average) and, in addition, our training accuracy is also the best under the vast majority of the cases, which hints that semantics have the highest processing efficiency by its up-sampling approximation.

It's worth noting that the baselines always process their messages with the very expensive ways in below that are absent in ours. **1)** *Sampling* has to recreate a new adjacency matrix for each round of random sampling, which involves a large amount of messy accesses to the memory. **2)** *Quantification* uses the library function of `torch.quantize_per_tensor()` to *offset* and *scale* all of the values in senders and convert them back in receivers, causing unbearable computing overheads. **3)** *Delay* is a memory-intensive method

Table 1: Communication volume, epoch time, and test accuracy of four compression on datasets. Partition numbers: 2, 4, and 8.

Dataset	Method	Comm. (MB)			Epoch Time (ms)			Test Accuracy (%)		
		2	4	8	2	4	8	2	4	8
Reddit	Vanilla.	303	790	1328	295.4	352.6	420.0	97.09	97.07	97.08
	Delay.	0.1	0.5	5.5	227.5	244.3	274.3	96.94	96.85	96.77
	Quant.				97.1	99.5	748.9	96.97	96.92	96.80
	Samp.				85.3	81.4	105.1	96.55	96.70	96.83
	Ours				**73.3**↓	**79.6**↓	**97.9**↓	**97.07**↑	**96.97**↑	**96.90**↑
Yelp	Vanilla.	885	1752	3374	509.3	352.5	387.4	65.27	65.27	65.24
	Delay.	77.9	275.1	570.2	1374.3	1597.7	2033.4	65.27	**65.34**↑	**65.33**↑
	Quant.				323.9	474.6	724.6	65.28	65.29	65.30
	Samp.				216.6	171.7	194.8	65.24	65.27	65.30
	Ours				**154.8**↓	**161.7**↓	**176.1**↓	**65.28**↑	65.24	65.12
Ogbn-p.	Vanilla.	413	1061	1538	904.9	1041.3	406.2	79.43	79.46	78.63
	Delay.	16.5	48.8	104.6	2497.1	2422.9	2938.9	78.85	78.83	78.57
	Quant.				323.9	474.6	1485.5	79.12	79.00	78.90
	Samp.				265.2	238.3	228.4	79.14	79.19	**79.35**↑
	Ours				**219.1**↓	**209.1**↓	**185.5**↓	**79.26**↑	**79.28**↑	79.30
PubMed	Vanilla	1.67	3.36	7.14	20.9	25.9	44.3	76.5	76.3	76.7
	Delay.	0.5	1.1	2.1	43.4	69.4	74.8	76.1	76.7	76.0
	Quant.				38.9	151.5	165.1	76.3	76.9	76.1
	Samp.				21.5	30.8	49.8	75.9	76.3	76.6
	Ours				**18.5**↓	**24.2**↓	**42.6**↓	**77.3**↑	**77.6**↑	**76.9**↑

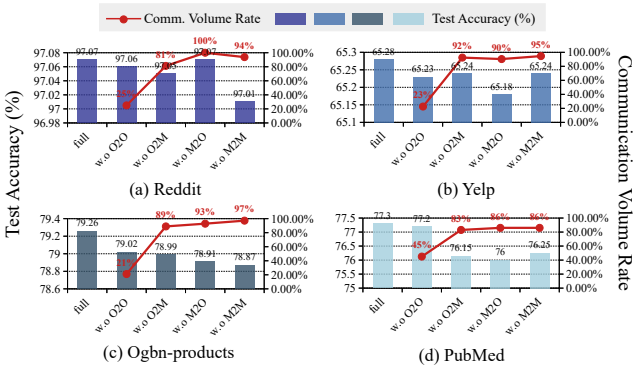


Figure 11: The results of differential optimization. The full cases do not involve any optimization.

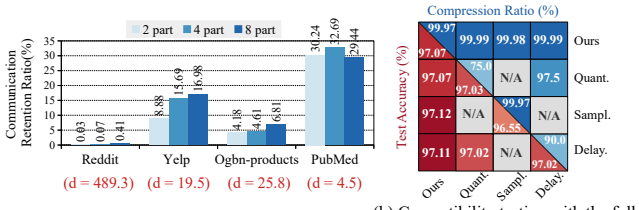
that repeatedly refers the out-of-date data in memory and periodically changes the data, incurring a memory-wall problem.

5.3 Accuracy vs. Volume: A Differential Optimization

The SC-GNN framework applies a deep compression to O2M, M2O, and M2M types of connections, but there is no processing to the remaining O2O traffic volume. A drawback is that the untamed O2O could become a new bottleneck in communications. Therefore, a differential optimization is proposed to test the model accuracy if a part of connections are removed from the graph. As shown in Fig. 11, an unexpected discovery is that totally removing any one type of connections has a very limited reduction in their training accuracy, however, the "without-O2O" is the only scenario that can further reduce the traffics down to 24% – 45%. As a result, the differential optimization is quite suitable for bandwidth-constrained distributed training.

5.4 Impact of Graphical Connectivity

Indeed, the in-group semantic compression is to use local full-mapping DBGs ($\mathcal{F}_{\mathcal{U}_i \to \mathcal{V}_i}$) to approximate high-cohesion ones, expressed as ($\mathcal{E}_{\mathcal{U}_i \to \mathcal{V}_i}$), and the full-mapping one is further compressed to single semantic message on traffic. This mechanism means that the compression ratio is quite sensitive to graphical connectivity, i.e., a graph with higher edge-density is more



(a) Average compression ratios over vanilla: degree vs. ratio

(b) Compatibility testing with the full combination of the compressions. N/A is non-convergence.

Figure 12: The impact of average degrees and the compatibility tests.

likely to have a better compression ratio. This phenomenon is evidenced in Fig. 12(a), where, Reddit traffic is successfully compressed to less than 0.5% since each node in the graph has an average degree $d = 489.3$, but the other three graphs cannot achieve that ratio since their average degrees are only 19.5, 25.8, and 4.5 respectively. In summary, our semantic compression is more friendly to those *intensive*-graph trainings.

In the proposed semantic compression, the compression ratio of the whole graph is also partially determined by number of groups. High cohesiveness leverages both *high* training accuracy and *low* compression ratio in that high cohesiveness mainly stems from small enough groups but a large number of small groups need to be compressed individually and costly. Our statistics averaged over the four datasets show that the compression rates decrease from 86.8% to 81.6% as the number of groups increases from 2 to 20, i.e., EEP, while the accuracy improved slightly by 0.13%. Beyond the EEP, the compression rate falls below 75%. The results show that it is not a sustainable optimization to infinitely increase the group numbers since the compression ratio suffers accelerated declines. Therefore, our method is able to achieve the trade-off by searching EEPs (Sec. 3.2) to avoid the ratio dropping.

5.5 Compatibility

As analyzed in Sec. 2.1, the compatibility of the three baselines is problematic with each other. A cross-compatibility testing in Fig. 12(b) traverses the all combining cases and gives the following three conclusions. **1)** Among the four methods, ours has the best compatibility with all others and both the accuracy and the compression are optimized. **2)** Quantification and delay are also usable but scarifies some compressing performances. **3)** Sampling is the most exclusive method, combining with which, quantification and delay methods even fail to converge.

Another compatibility issue is about how to choose a graph partition algorithm (Sec. 4). The results in Table 2 exhibit that the node-cut wins on the accuracy of the most of datasets (except Ogbn-products), furthermore, the absolute traffic volumes can be reduced up to 3.8 times, verifying its algorithmic isomorphism with the approximating compression used in our method. In opposite, the edge-cut and random partitions have the poor compatibility in volumes, eliminating their fairly good performance at accuracy.

6 CONCLUSIONS

In this work, we introduce SC-GNN, a comprehensive GNNs distributed training system to reduce cross-partition communication during the aggregation of GNNs. SC-GNN semantically clusters node embeddings/gradients into groups and compresses them to minimize communication overhead. The differential optimization is proposed to further prune weak connections. The experimental results show that our method successfully breakthroughs the Pareto frontier of SOTA methods while the training accuracy even has a slight improvement. At the same time, our method illustrates a better compatibility to be jointly used under resource-constrained training scenarios.

Table 2: Performance of node-cut, edge-cut, and random-cut algorithms on communication volume (CV) and accuracy.

Dataset	Method	Vanilla CV (MB)	SC-GNN CV (MB)	Acc.(%)
Reddit	node-cut	789.9	**0.5** ↓	**96.97** ↑
	edge-cut	**745.9** (0.94×) ↓	1.03 (2.05×)	96.91 (-0.06)
	random	1309.2 (1.66×)	1.08 (2.16×)	94.20 (-2.77)
Yelp	node-cut	**1752.2** ↓	**275.1** ↓	**65.24** ↑
	edge-cut	1979.8 (1.13×)	464.9 (1.69×)	65.17 (-0.07)
	random	5894.3 (3.36×)	495.2 (1.80×)	64.98 (-0.26)
Ogbn-p.	node-cut	**1060.9** ↓	**48.8** ↓	79.28
	edge-cut	1093.7 (1.03×)	70.8 (1.45×)	78.98 (-0.30)
	random	1100.6 (1.04×)	73.2 (1.50×)	**79.51** (+0.23) ↑
PubMed	node-cut	**3.4** ↓	**1.1** ↓	**77.3** ↑
	edge-cut	4.6 (1.41×)	1.89 (1.71×)	75.6 (-1.7)
	random	28.4 (8.60×)	4.18 (3.80×)	77.1 (-0.2)

ACKNOWLEDGMENTS

This work was supported by National NSF (No. 62272393) of China.

REFERENCES

[1] Zhenkun Cai, Xiao Yan, Yidi Wu, Kaihao Ma, James Cheng, and Fan Yu. 2021. DGCL: An efficient communication library for distributed GNN training. In *European Conference on Computer Systems (EuroSys)*, 130–144. ISBN: 9781450383349. DOI: 10.1145/3447786.3456233.

[2] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Conference on in Neural Information Processing Systems (NeurIPS)*, 1025–1035. arXiv: 1706.02216.

[3] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open graph benchmark: Datasets for machine learning on graphs. *Conference on Neural Information Processing Systems (NeurIPS)*, 1–16. arXiv: 2005.00687.

[4] George Karypis and Vipin Kumar. 1995. Metis – unstructured graph partitioning and sparse matrix ordering system, version 2.0. *Applied Physics Letters*, 97, 12, 1–3.

[5] Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations (ICLR)*, 1–14. arXiv: 1609.02907.

[6] R. Lempel and S. Moran. 2001. Salsa: the stochastic approach for link-structure analysis. *ACM Trans. Inf. Syst.*, 19, 2, (Apr. 2001), 131–160. DOI: 10.1145/382979.383041.

[7] Lingxiao Ma, Zhi Yang, Youshan Miao, Jilong Xue, Ming Wu, Lidong Zhou, and Yafei Dai. 2019. Neugraph: Parallel deep neural network computation on large graphs. *Proceedings of the 2019 USENIX Annual Technical Conference (USENIX ATC)*, 443–457. ISBN: 9781939133038.

[8] Vasimuddin Md, Sanchit Misra, Guixiang Ma, Ramanarayan Mohanty, Evangelos Georganas, Alexander Heinecke, Dhiraj Kalamkar, Nesreen K. Ahmed, and Sasikanth Avancha. 2021. Distgnn: scalable distributed training for large-scale graph neural networks. In *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)* Article 76. St. Louis, Missouri, 1–14. ISBN: 9781450384421. DOI: 10.1145/3458817.3480856.

[9] Adam Paszke, Sam Gross, and Massa. 2019. PyTorch: An imperative style, high-performance deep learning library. *Conference on Neural Information Processing Systems (NeurIPS)*. arXiv: 1912.01703.

[10] Stefania Salvatore et al. 2020. Beware the Jaccard: The choice of similarity measure is important and non-trivial in genomic colocalisation analysis. *Briefings in Bioinformatics (BiB)*, 21, 5, 1523–1530. DOI: 10.1093/bib/bbz083.

[11] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI Magazine*, 29, 3, 93–106. DOI: 10.1609/aimag.v29i3.2157.

[12] John Thorpe et al. 2021. Dorylus: Affordable, scalable, and accurate GNN training with distributed CPU servers and serverless threads. *Proceedings of the 15th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 495–514. ISBN: 9781939133229. arXiv: 2105.11118.

[13] Edy Umargono, Jatmiko Endro Suseno, and S.K Vincensius Gunawan. 2020. K-means clustering optimization using the elbow method and early centroid determination based on mean and median formula. *Proceedings of the 2nd International Seminar on Science and Technology (ISSTEC)*, 234–240. https://api.semanticscholar.org/CorpusID:226504265.

[14] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference on Learning Representations*. https://openreview.net/forum?id=rJXMpikCZ.

[15] Borui Wan, Juntao Zhao, and Chuan Wu. 2023. Adaptive message quantization and parallelization for distributed full-graph gnn training. *Proceedings of Machine Learning and Systems (MLSys)*, 1–14. eprint: 2306.01381 (cs.LG).

[16] Cheng Wan, Youjie Li, Ang Li, Nam Sung Kim, and Yingyan Lin. 2022. BNS-GCN: efficient full-graph training of graph convolutional networks with partition-parallelism and random boundary node sampling. *Proceedings of Machine Learning and Systems (MLSys)*, 1–11.

[17] Lei Wang, Qiang Yin, Chao Tian, Jianbang Yang, Rong Chen, Wenyuan Yu, Zihang Yao, and Jingren Zhou. 2021. Flexgraph: a flexible and efficient distributed framework for gnn training. In *Proceedings of the 16th European Conference on Computer Systems (EuroSys)*, 67–82. ISBN: 9781450383349. DOI: 10.1145/3447786.3456229.

[18] Minjie Wang et al. 2019. Deep graph library: a graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315*.

[19] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor K. Prasanna. 2021. Graphsaint: graph sampling based inductive learning method. *International Conference on Learning Representations (ICLR)*, abs/1907.04931, 1–19. https://api.semanticscholar.org/CorpusID:195886159.

[20] Rong Zhu, Kun Zhao, Hongxia Yang, Wei Lin, Chang Zhou, Baole Ai, Yong Li, and Jingren Zhou. 2018. AliGraph: A comprehensive graph neural network platform. *VLDB*, 12, 12, 2094–2105. arXiv: 1902.08730. DOI: 10.14778/3352063.3352127.