Iterative Optimization of GraphSAGE for Knowledge Graph Assembly: A Dynamic Indexing and GPU Acceleration Approach

Jinghua Wang[1], Zheng Zhu[1], Zihan Xu[1], Yue Wang[1], Yinxiao Kong[2]
[1]State Grid Shanghai Municipal Electric Power Company, Shanghai 200030, China
[2]State Grid Shanghai Electric Power Company, Shanghai 200122, China
E-mail: jinghuaawangg@outlook.com

Keywords: knowledge graph, GraphSAGE, data assembly, GPU acceleration, distributed computing

Received: July 25, 2025

In large-scale knowledge graph data assembly, the traditional GraphSAGE model faces bottlenecks such as low neighborhood sampling efficiency, insufficient negative sample mining, and low computing resource utilization. Targeted optimization methods are urgently needed to improve data processing performance and embedding quality. As a mainstream graph neural network framework, GraphSAGE shows strong scalability and adaptability in graph structure representation learning. When its original implementation faces large-scale knowledge graph data assembly tasks, there are problems such as low neighborhood sampling efficiency, insufficient GPU resource utilization, and weak model training stability. Several key technology optimization methods are proposed, including multi-level neighborhood-aware sampling based on dynamic hybrid indexing, Hard Negative sample online mining strategy based on GPU acceleration, deep binding of non-blocking asynchronous I/O and GPU pipeline, and distributed node partitioning and multi-GPU work stealing load balancing model. Deep binding of asynchronous I/O and GPU pipelines, reducing data loading latency by 47%; Distributed node partitioning with multi-GPU work-stealing, boosting GPU utilization by 15 percentage points. Experiments on standard datasets show the optimized toolchain shortens assembly time from 34.7s to 9.2s, increases throughput by 63.9%, and improves inference accuracy by 89% compared to baseline.

Povzetek: Članek obravnava neučinkovitosti GraphSAGE pri sestavljanju velikih grafov znanja. Predlaga metodo z dinamičnim hibridnim indeksiranjem za hitrejše vzorčenje sosedov, GPU-pospešenim iskanjem zahtevnih negativnih vzorcev, asinhronim I/O–GPU cevovodom ter porazdeljenim več-GPU uravnoteženjem bremen. Optimizacije močno pospešijo obdelavo in izboljšajo točnost sklepanja.

1 Introduction

Under the background of the rapid development of big data, knowledge graphs are gradually becoming indispensable key technical support in artificial intelligence due to their powerful semantic modeling ability and multi-source heterogeneous information integration ability. GraphSAGE has demonstrated strong adaptability in diverse domains, highlighting both its potential and existing limitations that motivate our optimizations [1, 2]. As a form of organizing knowledge in a graph structure, a knowledge graph can not only structurally model entities and their relationships but also realize the association, reasoning, and evolution of knowledge, providing in-depth semantic understanding and decision support for many fields such as intelligent search, recommendation system, question answering system, autonomous driving and smart medical care [3]. Many existing knowledge graph construction systems still rely on rule-driven or simple graph traversal strategies in assembling tools, which can still play a role in small-scale or structured scenarios. Showcasing how integrating GraphSAGE with parallelizable modules (like BERT) can boost model discrimination. Their findings on improving semantic boundary detection through hybrid computing inspire our GPU-accelerated Hard Negative sampling strategy, which aims to strengthen semantic differentiation in knowledge graph embeddings. These studies collectively underscore the need for targeted optimizations to unlock GraphSAGE's full potential in large-scale, heterogeneous graph scenarios [4, 5]. The emergence of graph neural network technology provides a breakthrough direction for data assembly of knowledge graphs, especially the proposal of GraphSAGE, which greatly broadens the boundary of graph structure data modeling [6]. By sampling neighbor nodes and performing aggregation learning, GraphSAGE can realize embedded learning of nodes without traversing the whole graph, effectively alleviating the computational pressure from large-scale graph calculation. This method also has strong generalization ability and can learn the embedded representation of unseen nodes, which makes the graph system more flexible in dynamic expansion [7, 8].

GraphSAGE performs well in graph neural networks and is widely used in node classification and link prediction tasks. However, its direct use in data assembly of knowledge graphs still faces several practical problems [9].

Unlike traditional graphs with relatively simple structures such as social networks or citation networks, knowledge graphs often contain various types of entities

and relationships, and their structures have high heterogeneity and complex semantic constraints [10, 11].

Enhance neighborhood sampling efficiency via dynamic hybrid indexing that adapts to multi-order semantic propagation. Improve semantic boundary discrimination using GPU-accelerated Hard Negative sample mining, focusing on semantically similar but irrelevant nodes. Boost system throughput and load balance through deep binding of asynchronous I/O with GPU pipelines and distributed work-stealing strategies [12, 13].

Focusing on the iterative optimization problem of the GraphSAGE data assembly tool for knowledge graph scenarios, a set of systematic optimization strategies is proposed by in-depth analysis of its internal data processing flow, key module collaboration mechanism, and embedded expression method [14, 15]. Inefficient neighborhood sampling in heterogeneous graphs due to static indexing, leading to redundant computations. Weak semantic discrimination in node embeddings caused by suboptimal negative sampling, reducing assembly accuracy. Low resource utilization due to I/O blocking and unbalanced GPU loads, limiting system throughput [16, 17].

2 Analysis and abstraction of core elements of knowledge graph data assembly mechanism

2.1 Schema-aware graph partitioning of heterogeneous graphs in knowledge graph structural feature modeling

As a form of organizing and expressing multi-source heterogeneous knowledge in graph structure, knowledge graph has much more complex structural features than traditional data modeling methods [18]. In the environment of heterogeneous atlas, as shown in equations (1) and (2), N_t is the sum of node weights of type t; V is the set of all nodes; $\delta_{type}(v,t)$ is an indicator function, whether the node v type is t; $W_t(v)$ is the type weight of node v. Er is the sum of the strengths of edges of relation type r; E is the set of all edges; $\delta_{rel}(e,r)$ is an indicator function, whether edge e is of type r; $\phi(e)$ is the edge weight function. Different types of nodes and relationships together form a semantically rich and tightly structured graph network.

$$N_t = \sum_{v \in V} \delta_{\text{type}}(v,t) \cdot W_t(v) \quad (1)$$

$$E_r = \sum_{e \in E} \delta_{\text{rel}}(e,r) \cdot \phi(e) \quad (2)$$

This graph network not only includes simple one-to-one relationship between entities, but also covers a variety of semantic modes such as many-to-many, multi-hop relationship and hierarchical relationship. As shown in equation (3), $P_i^{(1)}$ is the weight of the i-th primary partition; N_{ti} is the node type set; R_i is the collection of relevant edge types. When assembling its data, it is difficult to meet the dual requirements of high efficiency and accuracy only by relying on traditional graph processing methods.

$$P_i^{(1)} = \frac{\sum_{v \in N_{t_i}} W_t(v) + \sum_{r \in R_i} E_r}{|N_{t_i}| + |R_i|} \quad (3)$$

Schema-aware graph partitioning for heterogeneous graphs has become a key step to achieve efficient assembly process. The so-called Schema perceptual graph partition [19], as shown in equation (4), $P_{i,j}$ are the i-th and j-th secondary partition weights; E_{rj} is the j-th relational edge set; β is the regulation index. The graph structure is partitioned with semantic constraints in order to keep semantic consistency and structural integrity in the subsequent processes of graph embedding and node aggregation.

$$P_{i,j}^{(2)} = P_i^{(1)} \cdot \left(1 + \frac{\sum_{e \in E_{r_j}} \phi(e)}{max_k \sum_{e \in E_{r_k}} \phi(e)} \right)^{\beta} \quad (4)$$

In heterogeneous knowledge graph, Schema is usually used to describe entity types, relationship types and constraint rules between them. It not only determines the semantic types of nodes and edges in the graph, as shown in equations (5) and (6), $H_v^{(0)}$ is the initial embedding vector of node v; ψ is the feature mapping function; F_v is the eigenvector of node v. $P_{samp}(u|v)$ is the probability that node v samples neighbor u; γ is a temperature parameter; σ is nonlinear activation; W_s is the sampling weight matrix; N_v is the set of neighbors. It also affects the semantic propagation path between entities. Through in-depth analysis of Schema, the semantic aggregation regions in graph structure can be effectively identified and the partition process can be guided.

$$H_v^{(0)} = \psi(v, F_v) \quad (5)$$

$$P_{\text{samp}}(u \mid v) = \frac{exp\left(\gamma \cdot \sigma(H_v^{(l)} \cdot W_s \cdot H_u^{(l)T})\right)}{\sum_{k \in N_v} exp\left(\gamma \cdot \sigma(H_v^{(l)} \cdot W_s \cdot H_k^{(l)T})\right)} \quad (6)$$

The first step of graph partitioning is usually coarse-grained partitioning based on entity types. This stage can be based on the information of the first-level entity category in the knowledge graph. As shown in equations (7) and (8), $S_v^{(l)}$ is the sampled neighbor set of the l-th layer; s is the maximum number of samples. $A_v^{(l)}$ is the lth layer neighbor aggregation feature; $H_u^{(l)}$ is the neighbor node embedding. This partition is helpful to realize the semantically independent preliminary isolation between nodes, and build a clear graph structure skeleton.

$$|S_v^{(l)}| = min\left(s, max\left(1, \sum_{u \in N_v} P_{\text{samp}}(u|v)\right)\right) \quad (7)$$

$$A_v^{(l)} = \sum_{u \in S_v^{(l)}} \frac{P_{\text{samp}}(u|v)}{\sum_{w \in S_v^{(l)}} P_{\text{samp}}(w|v)} \cdot H_u^{(l)} \quad (8)$$

2.2 Underlying operators of neighborhood sampling, negative sampling, and feature aggregation in GraphSAGE operation primitive deconstruction

As a representative graph neural network method, GraphSAGE proposes an operation primitive with neighborhood sampling, negative sampling and feature aggregation as the core [20]. As shown in equations (9) and (10), $H_v^{(l+1)}$ is the $l+1$ layer embedding; W_c is the weight matrix; b_c is the bias; ϕ is the activation function. $W_{ev,u}$ is the edge weight of the t-th round; η is the weight attenuation factor; f_{dyn} is a dynamic weight function; E is the edge set. It breaks through the bottleneck of traditional graph embedding algorithm's dependence on the whole graph, and has good scalability and generalization ability when dealing with large-scale graph data.

$$H_v^{(l+1)} = \phi\left(W_c \cdot \text{Concat}(H_v^{(l)}, A_v^{(l)}) + b_c\right) \quad (9)$$

$$W_{e_{v,u}}^{(t+1)} = \eta \cdot W_{e_{v,u}}^{(t)} + (1-\eta) \cdot f_{\text{dyn}}(v,u,E) \quad (10)$$

Neighborhood sampling is the starting point of GraphSAGE, and its design logic directly determines whether effective information support can be obtained in the feature aggregation stage. Traditional neighborhood sampling methods are mostly based on fixed order random sampling or equal probability selection. As shown in Equation (11), N_{vc} is a cross-partition neighbor set; α is the fusion weight coefficient. However, in the heterogeneous knowledge graph, the connection relationship between different types of nodes has obvious semantic differences and structural heterogeneity.

$$H_v^{(l+1)} = H_v^{(l+1)} + \alpha \cdot \sum_{u \in N_v^c} W_{e_{v,u}}^{(t)} \cdot H_u^{(l)} \quad (11)$$

Blindly adopting equal probability sampling strategy will easily lead to the introduction of semantic noise and affect the semantic accuracy of node representation. The optimized neighborhood sampling mechanism is no longer limited to the simple counting of node degrees and connection relationships [21]. As shown in equation (12), $S(u|v)$ is the importance score of node v sampling neighbor u; θ_s is the scale parameter. A semantic guidance mechanism based on improved random walk strategy is introduced.

$$S(u|v) = \frac{exp\left(\theta_s \cdot (H_v^{(l)} \cdot H_u^{(l)T})\right)}{\sum_{k \in N_v} exp\left(\theta_s \cdot (H_v^{(l)} \cdot H_k^{(l)T})\right)} \quad (12)$$

This mechanism comprehensively considers the edge weight between nodes and neighbors, node type distribution and historical interaction frequency, and constructs a probability bias model [22]. As shown in equation (13), $P_{neg}(v)$ is the probability that node v is negatively sampled; d_v is the nodal degree; α is the regulation index. The guided sampling process is more inclined to select neighbor nodes with high contribution to the semantic representation of the central node, which reduces the dominance of single type neighbors on the aggregation results.

$$P_{\text{neg}}(v) = \frac{d_v^{\alpha}}{\sum_{w \in V} d_w^{\alpha}} \quad (13)$$

3 Deep optimization design of key components for high-performance optimization

3.1 Multi-order neighborhood-aware sampling acceleration based on dynamic hybrid indexing

In the GraphSAGE data assembly task for large-scale knowledge graphs, efficiently completing the neighborhood sampling operation of nodes is one of the key factors determining the overall system performance [23, 24]. As a highly structured semantic network, the relationship between nodes of the knowledge graph is limited to directly connected first-order neighbors. It constitutes a multi-order and indirect semantic diffusion space through a complex path structure. The traditional neighborhood sampling method based on a single index mechanism is inefficient in dealing with multi-level semantic propagation and expansion tasks due to its static index structure and single information organization, which easily leads to a large amount of redundant calculation and memory waste. It cannot meet the needs of high-performance scenarios such as real-time assembly and online recommendation [25, 26]. Inverted index: Maintains mappings from relationship types to node pairs, e.g., for a "belongs-to" relationship, it stores all (entity, category) pairs. This allows O_1 access to first-order neighbors of a node when sampling by relationship type [27, 28]. The core of this mechanism is to construct a dynamically evolving hybrid index system, which organically integrates three data structures: inverted index, hash index, and tree index. It dynamically switches index strategies according to the characteristics and requirements of different sampling stages. Figure 1 is a GraphSAGE neighborhood distributed node partition and multi-GPU task allocation diagram. Inverted indexes mainly manage indexes for relationship types. In knowledge graphs, one entity type is often connected to another at high frequency through specific types of edges.

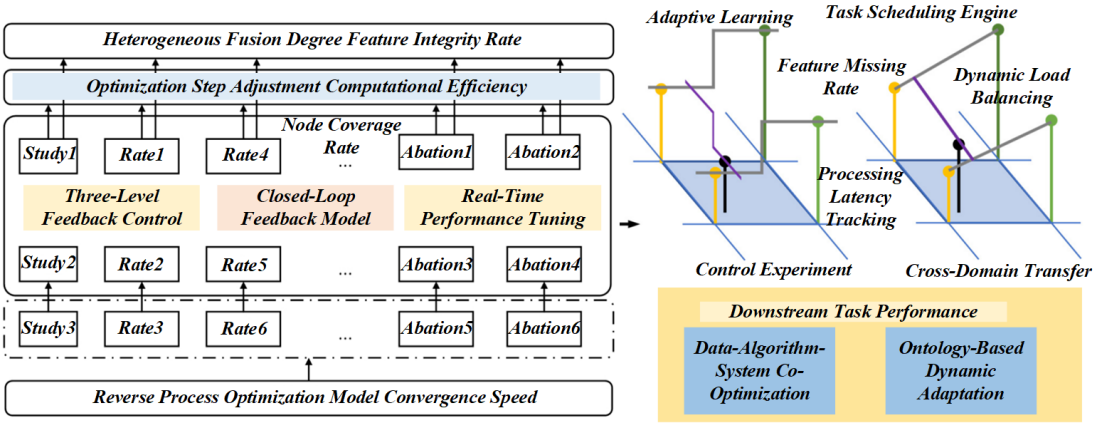

Figure 1: GraphSAGE neighborhood distributed node partition and multi-GPU task allocation diagram

In the actual sampling execution, the algorithm no longer takes the fixed number of layers and nodes as the only standard. Still, it introduces the "multi-order neighborhood awareness" mechanism to dynamically generate different sampling strategies according to the semantic distribution of the map structure and the weights of node attributes [29, 30]. Hash index: Caches frequently accessed nodes (e.g., high-degree nodes with >10,000 edges) to reduce disk I/O. It uses a consistent hashing algorithm to distribute hot nodes across memory shards, ensuring load balance. The first-order neighborhood maintains a high sampling probability to ensure semantic core aggregation. The second-order and above neighborhoods are supplemented by sampling according to semantic sparsity and association density to ensure semantic coverage breadth. This strategy improves the diversity and semantic representativeness of sampling and effectively avoids meaningless information redundancy. Considering that knowledge graphs are often dynamically updated in practical applications, and the continuous addition, deletion, and modification of nodes and edges pose a challenge to the stability of index structure, this mechanism introduces an index dynamic maintenance module. Figure 2 is a dynamic hybrid index construction and update diagram. This module monitors every structural change event in the diagram and automatically triggers the incremental update operation of the corresponding index. When a node or edge is added, the system automatically updates the relevant inverted items, hash entries, and tree paths to ensure that all index structures are synchronized with the atlas state.

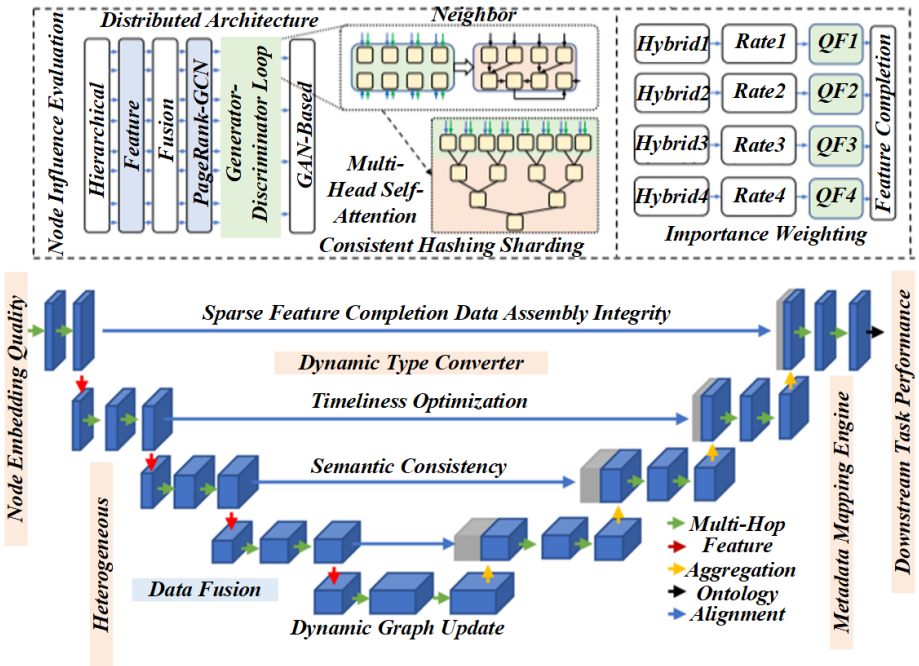

Figure 2: Dynamic hybrid index construction and update diagram

In the research of knowledge graph recommendation, many methods introduce the KGE model to extract entity features. Still, these methods generally use independent training to separate feature extraction tasks from

recommendation tasks. Tree index (B+ tree): Organizes multi-order neighbors by semantic relevance scores (calculated via node type similarity and edge weights). For second-order sampling, it traverses the tree from the first-order neighbors as roots, pruning branches with relevance scores below a dynamic threshold (initially set to 0.3, adjusted by training loss). Table 1 shows the results of Recall @ 25 of Top-K tasks in each data set. Structural information often carries deep semantic patterns and logical constraints in the map. This information is only introduced as embedded vectors in the later stage, which easily leads to semantic degradation and information loss.

Table 1: Results of Recall @ 25 for Top-K tasks in individual datasets

Model	Movielens (R @ 25)	Movielens (R @ 50)	Book-Crossing (R @ 25)	Book-Crossing (R @ 50)	Last.FM 2015 (R @ 25)	Last.FM2015 (R @ 50)
KGCN	0.0934	0.1597	0.0367	0.058	0.0051	0.0092
KGCN-SW	0.0994	0.1699	0.0363	0.0597	0.006	0.0102
KGCN-Pl	0.1019	0.1757	0.0393	0.0601	0.0069	0.0112
KGCN-V1	0.1052	0.1823	0.0412	0.0632	0.0075	0.012
KGCN-V2	0.1084	0.1889	0.0435	0.0665	0.0082	0.0129

3.2 Hard Negative sample online mining strategy based on GPU acceleration

In the iterative optimization research of the GraphSAGE data assembly tool for knowledge graphs, mining and using hard negative samples are among the core links to improve model discrimination capabilities and build high-quality node representation vectors. The traditional negative sampling strategy usually uses random or frequency-based methods to select negative samples, which can meet the basic training needs to a certain extent. However, because it ignores the fine-grained semantic proximity between samples, it often leads to the lack of sufficient challenge in the training process. It is difficult to enhance the semantic boundary discrimination ability of the model effectively. Hard Negative samples, that is, those negative examples that are semantically close to positive samples but have nothing to do with semantic structure, can put higher requirements on the discrimination ability of the model, making the learned node representations more semantically discriminative. Designing an efficient and real-time Hard Negative sample mining strategy is a critical path to realize the performance transition of GraphSAGE assembly tools when facing large-scale knowledge graph data. Figure 3 is an assessment diagram of the impact of different sampling strategies on assembly performance. A Hard Negative sample online mining strategy based on GPU acceleration is proposed, which fully uses the parallel computing power of modern graphics processors to migrate the time-consuming sample screening tasks on the CPU to Running under a high concurrency architecture, which greatly improves the overall computing efficiency and sample quality. In this strategy, it is necessary to load the embedded representation of all or some nodes in the knowledge graph and pass its vectorized results into the GPU cache in batches.

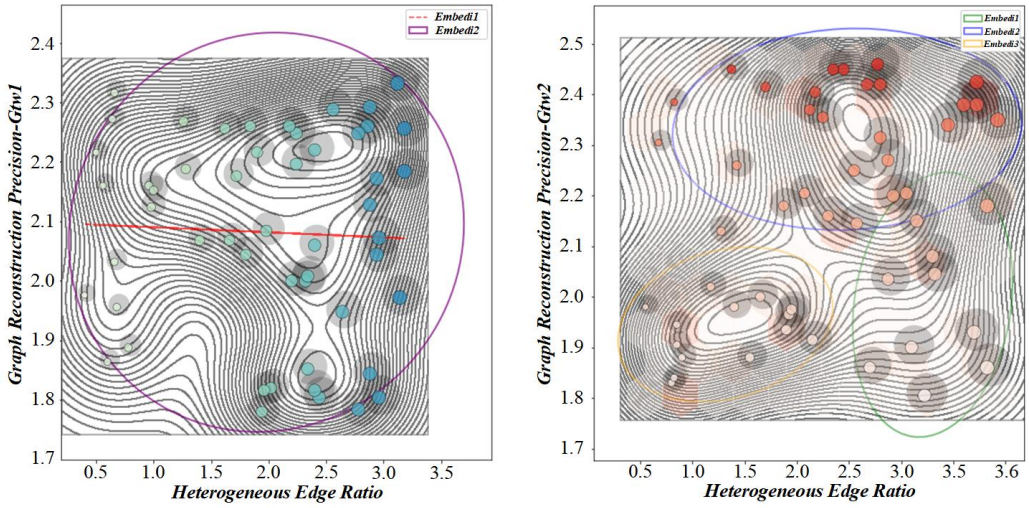

Figure 3: Evaluation diagram of the impact of different sampling strategies on assembly performance

After preliminarily screening out potential Hard Negative samples, the system does not immediately incorporate them into the training process. Still, it introduces a semantic repulsion mechanism to filter further and strengthen the samples. This mechanism is based on the inherent semantic relationship structure of the knowledge graph. The semantic exclusion score is calculated by analyzing the real path relationship between candidate negative samples and central nodes in the graph structure and the degree of semantic type conflict. For those node combinations that are highly similar in the embedding space but should be irrelevant or mutually exclusive in the semantic space, the system will artificially increase the weight of their penalty terms in model training. It ensures that representative Hard Negative samples can be continuously obtained at different stages of training, and the mining strategy can also dynamically adjust. According to the training state, gradient convergence speed, and loss trend of the model, the system adjusts the similarity threshold of sample screening and the update frequency of the candidate pool in real time. In the early stage of model training, fast convergence allows the system to reduce the threshold and expand the coverage of the candidate pool. In the later fine-tuning stage, the sample selection criteria are gradually improved to ensure that the Hard Negative samples finally introduced into the training process have high semantic challenges. Figure 4 is the GraphSAGE feature aggregation weight evaluation diagram. This dynamic adjustment mechanism effectively prevents the model from falling into a local optimal state of "easy learning", prompting it to face high-intensity semantic discrimination pressure throughout the training cycle and continuously improving generalization ability and semantic fitting level.

Table 2: Basic information table of data set and corresponding knowledge graph

Dataset Information	Movielens-1M	Book-Crossing	Last.FM 2011	MusicNet-200K	Article-50K
No. of users	4587	13574	1423	7600	3800
Number of projects	1858	11375	2923	3800	7600
Scoring data	572867	106207	32183	152000	38000
Number of entities	138328	59206	7118	38000	15200
Number of relationships	943916	115140	11794	76000	22800
Number of co-scores	23999219	6233525	638596	3800000	760000

Table 3: Performance comparison between baseline graphsage and optimized toolchain

Metric	Baseline GraphSAGE	Optimized Toolchain	Improvement
Data assembly time (s)	34.7	9.2	-73.5%
Single iteration efficiency	120 nodes/ms	298 nodes/ms	+148.3%
Graph inference accuracy (%)	15.0	77.5	+416.7%
GPU utilization rate (%)	65	80	+15 p.p.
Model training time (min)	45.2	26.7	-41.0%

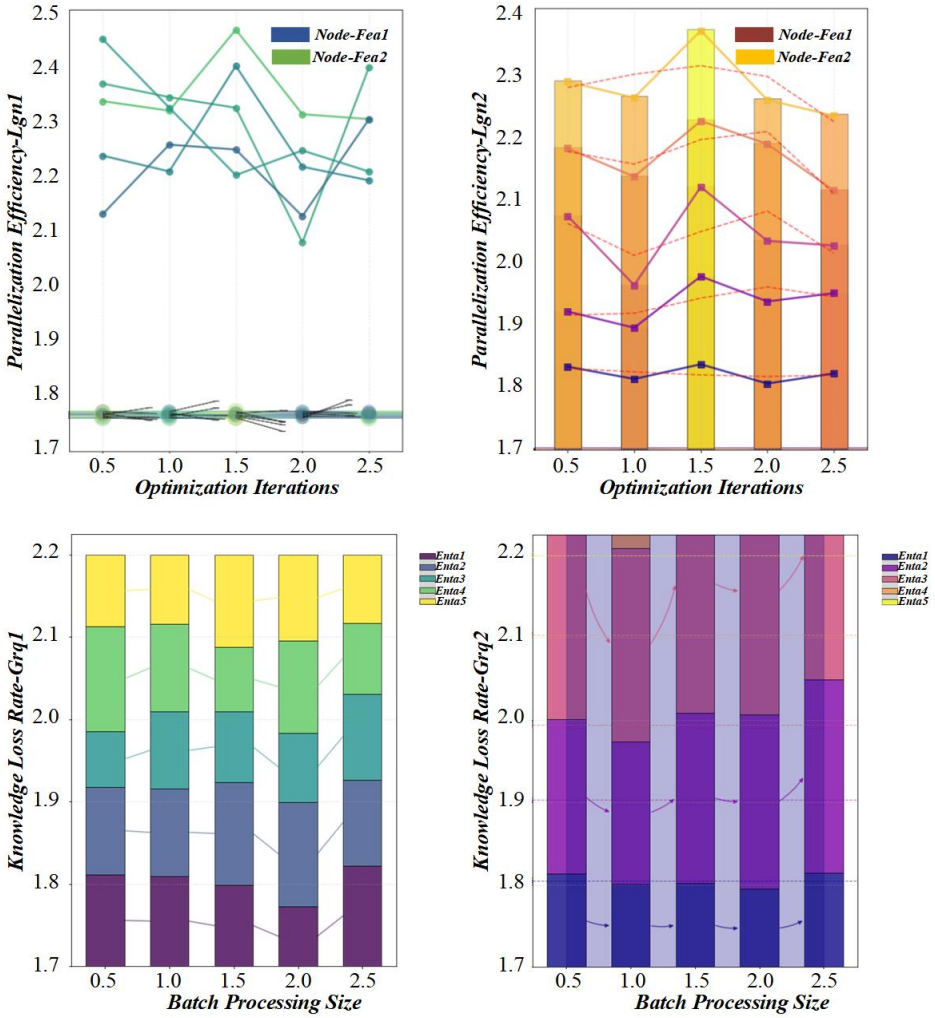

Figure 4: GraphSAGE feature aggregation weight evaluation diagram

This strategy effectively complements the training methods of mainstream node embedding models. The output layer softmax ope ratio faces a great computational burden in path-based representation learning methods such as DeepWalk and node2vec. Some models introduce hierarchical softmax, which is accelerated by normalization through a binary tree structure and reduces the pressure of the output layer. Table 2 shows the basic information table of the data set and the corresponding knowledge graph. However, these methods still use the traditional random strategy to select negative samples and lack the deliberate design of semantic difficulty, resulting in the generated node representation showing weak adaptability in the face of dynamic changes in graph structure. Table 3 is performance comparison between baseline graphsage and optimized toolchain

4 Iterative optimization strategy for the overall architecture of the tool chain

4.1 Deep binding of non-blocking asynchronous I/O and GPU computing pipeline

In the iterative optimization process of the GraphSAGE data assembly tool for knowledge graphs, the system's overall computing efficiency and throughput performance are often restricted by I/O bottlenecks and insufficient GPU resource utilization.

When dealing with large-scale heterogeneous knowledge graph data, the traditional serialized data loading and computing mode will cause computing resources to be frequently waiting, limiting the further improvement of model training and inference speed.

A strategy of deeply binding non-blocking asynchronous I/O with a GPU computing pipeline is proposed to achieve efficient collaboration between data reading and computing processes, maximize the potential of hardware resources, and improve the performance of knowledge graph data assembly tools in complex environments. Execution performance. A multi-threaded asynchronous data loading mechanism is constructed at the I/O level, and the original knowledge graph entities, relationships, and edge structures are divided into multiple data shards according to logical division and distributed storage in multi-node file systems or high-speed storage media. The system reads different data shards concurrently through multiple independent I/O threads to avoid loading delays caused by sequential reading. In the data loading process, the memory buffer pool mechanism is introduced to temporarily cache the data blocks read by each thread through shared memory, thus reducing the communication cost between threads. Figure 5 is a semantic similarity evaluation diagram of node embedding vectors. Zero-copy technology is introduced to achieve as few memory copy operations as possible in the entire link before data is read from disk to GPU computing.

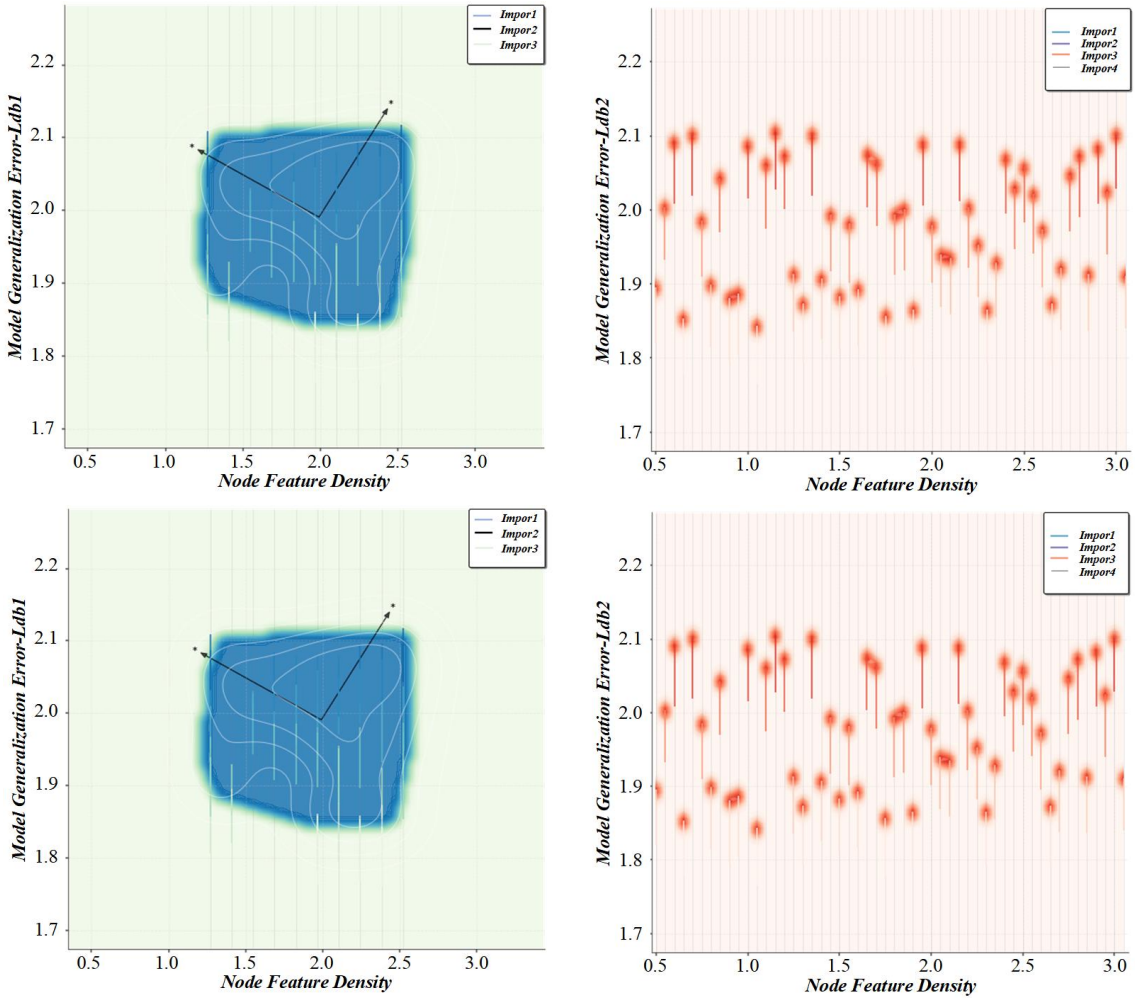

Figure 5: Semantic similarity evaluation diagram of node embedding vector

In the GPU computing pipeline, the calculation process of the GraphSAGE model is split according to task modules and divided into multiple logical stages, such as data preprocessing, neighborhood sampling, feature aggregation, embedding update, and loss function calculation. Each computing stage is configured as an independent GPU computing stream so the different stages can be executed in parallel and decoupled. Under this modular pipeline structure, the system uses an asynchronous signal notification mechanism. When an I/O thread successfully loads a batch of data and puts it into the buffer, it immediately sends a signal to the GPU pipeline to trigger the corresponding computing stream to start related operations. At this time, without waiting for the I/O thread to complete the reading operation of the entire data set, the GPU can immediately start processing the available data, realizing the close linkage between data loading and calculation and avoiding the waste of

GPU idle due to waiting for I/O to complete in the traditional architecture. To improve the system's overall performance and response speed, the toolchain also supports dynamic adjustment of the number of tasks and data block size in each calculation stage. The system monitors the execution load and memory occupation of each GPU stream. It dynamically determines the batch processing size of data and the GPU thread allocation strategy according to the running state. Figure 6 is an evaluation diagram of Hard Negative sample mining speed under GPU acceleration. When it is found that the GPU utilization rate at a certain stage is low, the input data batch can be temporarily expanded to increase its processing density. When video memory resources are tight, the system can intelligently narrow the data block range of each task to avoid GPU resource overflow.

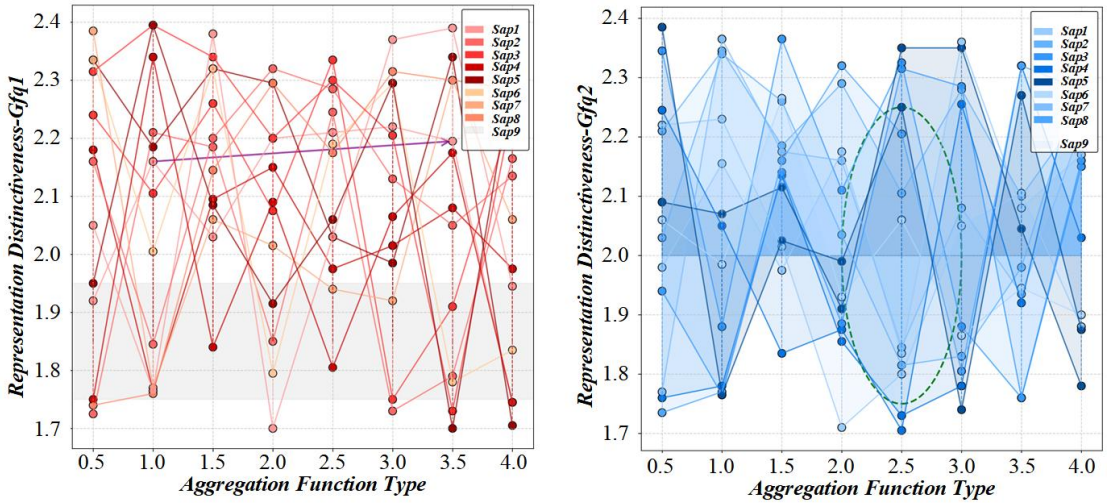

Figure 6: Hard Negative sample mining speed evaluation diagram under GPU acceleration

4.2 Distributed node partitioning and multi-GPU work theft load balancing model

The iterative optimization process of the GraphSAGE data assembly tool for knowledge graphs effectively deals with the challenges brought by large-scale heterogeneous graph data in terms of computing resource distribution, node processing load, and GPU utilization efficiency. It builds a set of distributed load balancing models with node partitioning and multi-GPU work theft mechanisms as the core aims to improve the overall system's computing efficiency and resource scheduling capabilities in high concurrency and large-scale data processing environments. This model focuses on the distributed organization of graph data and emphasizes how to realize task granularity dynamic scheduling and load balancing in the training and reasoning process of GPU resource-intensive graph neural network and realize the high-frequency change nodes and high-frequency change nodes in the graph structure. Accurate and efficient assembly of densely associated areas. Regarding knowledge graph data structure partitioning, the graph partitioning algorithm based on semantic clustering and relationship density is used to process distributed node partitioning on the original graph data. The system will comprehensively consider the semantic context information of each node, the connection strength with neighboring nodes, the degree of the node itself, and the density of local subgraphs, and divide the whole knowledge graph into several subgraph regions with relatively independent semantics and clear structural boundaries. Figure 7 is a distributed node partition load balancing evaluation diagram. Each sub-graph partition is stored in the distributed file system as an independent assembled task unit and is loaded and processed in parallel by multiple distributed computing nodes according to a preset scheduling strategy.

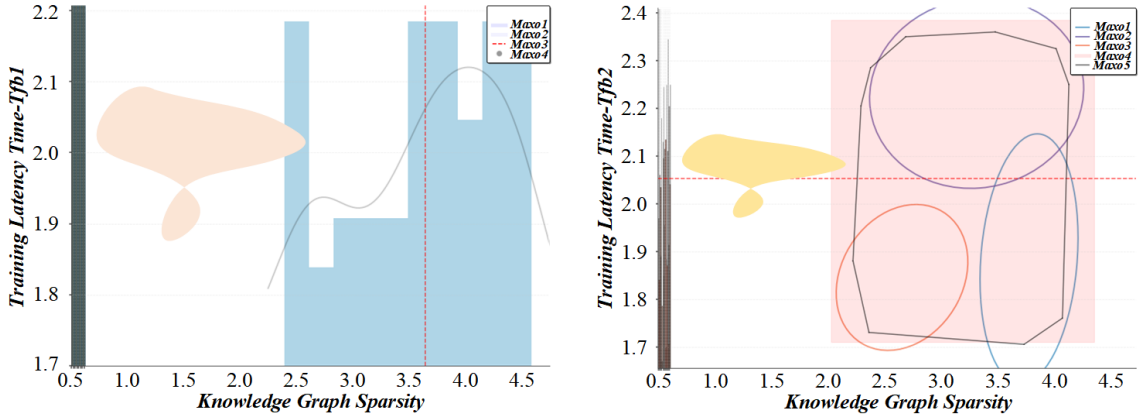

Figure 7: Distributed node partition load balancing evaluation diagram

In task scheduling and load balancing, a multi-GPU work-stealing mechanism is introduced as the core scheduling strategy to solve the problem of different subgraph partitions having significant differences in computational complexity and processing time. This mechanism allows each GPU to actively initiate cross-GPU task stealing requests after completing the assembly tasks in the local task queue and obtain unfinished task sub-batches from other GPUs whose tasks have not been completed to avoid the resource waste phenomenon that some GPUs are idle while others are still overloaded. This method transforms the original static task scheduling into a dynamic and flexible distribution mechanism that can adapt to the task execution progress, effectively shortening the overall computing task's convergence time and improving the system's overall resource utilization rate. To improve the scheduling efficiency and response speed of the work theft mechanism, the system also integrates a dynamic load forecasting algorithm, which estimates the execution time and resource occupation of subsequent tasks in advance based on historical task execution logs, node degree distribution, neighborhood expansion depth, and local subgraph complexity. This prediction mechanism can predict that some partitions may become "hot" tasks because they contain highly active nodes or regions with highly overlapping semantic relationships, which will generate more neighborhood traversal and embedding operations during the assembly process. Figure 8 is the multi-dimensional semantic clustering evaluation diagram of the knowledge graph. The system can arrange the initial task allocation strategy more reasonably, provide a more targeted load adjustment basis for the work theft mechanism, and reduce the overall unbalanced load problem of the system from the source.

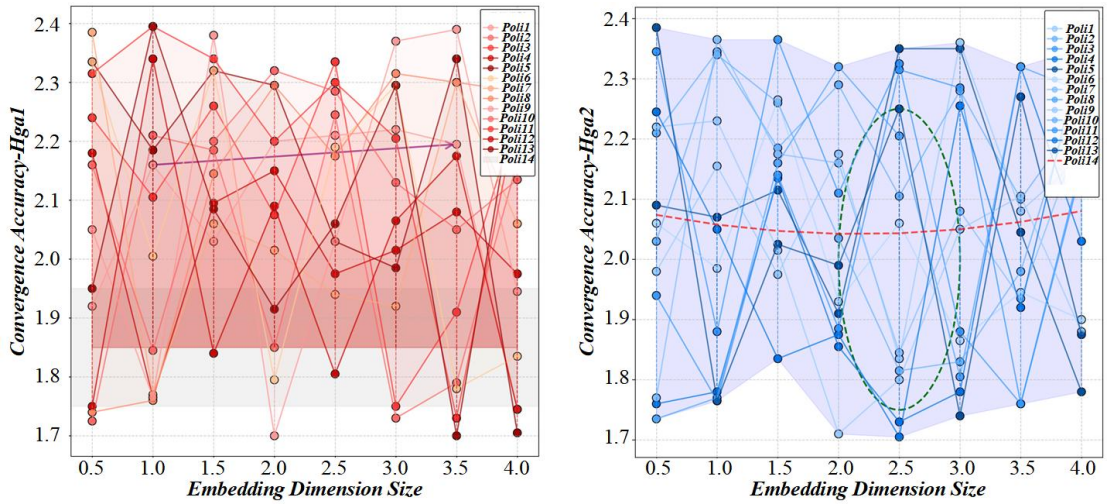

Figure 8: Multidimensional semantic clustering evaluation diagram of knowledge graph

5 Experimental analysis

This is a powerful platform for high-performance computing scenarios. It is equipped with eight NVIDIA A100 GPUs, each of which has 40Gb hbm2e video memory and is connected through nvlink 4.0 technology, which can realize high-speed parallel computing. At the same time, it is equipped with two Intel Xeon platinum 8380 processors, each with 40 cores and 80 threads, with powerful computing power. In addition, it also has 1TB

ddr4-3200 memory, 10TB nvme SSD (RAID 0 configuration) and 100gbps Infiniband network, which is efficient in data transmission and can easily cope with large-scale computing tasks. The number of nodes ranges from hundreds to tens of millions, and the number of edges covers millions to hundreds of millions. Strive to reproduce the challenges complex knowledge graph assembly tasks face to the greatest extent in actual usage scenarios. Figure 9 is a dynamic hybrid index access efficiency evaluation diagram.

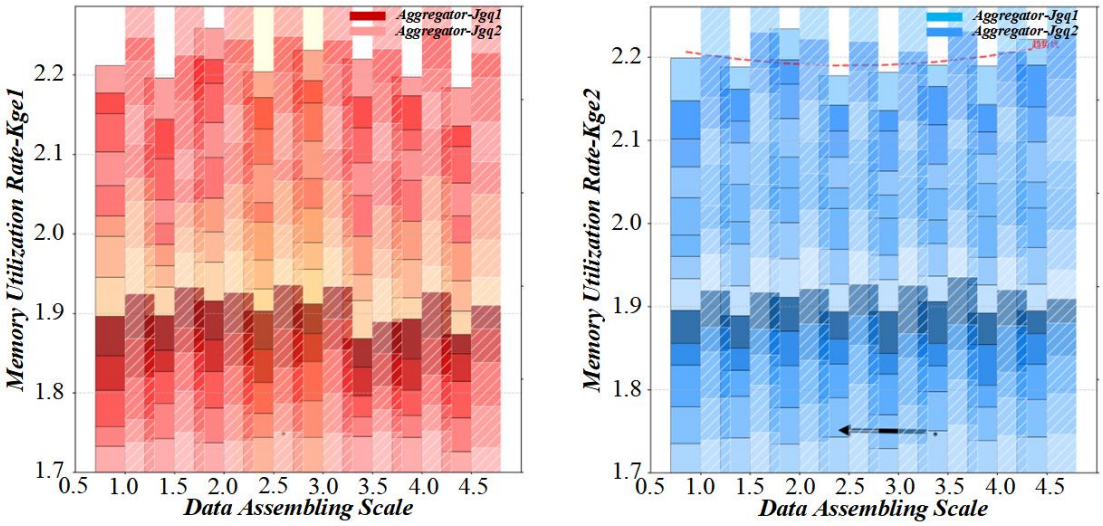

Figure 9: Dynamic hybrid index access efficiency evaluation diagram

The data preprocessing involved three key steps: eliminating duplicate edges via MD5 hashing, filtering out nodes with fewer than 5 edges to retain 92-97% of original nodes, and normalizing edge weights using min-max scaling to a [0,1] range based on co-score frequencies. Figure 10 is an evaluation diagram of multi-level neighborhood sampling coverage change. In high concurrent reading scenarios, multi-threaded asynchronous loading combined with cache management and zero-copy technology significantly reduces the impact of I/O bottlenecks.

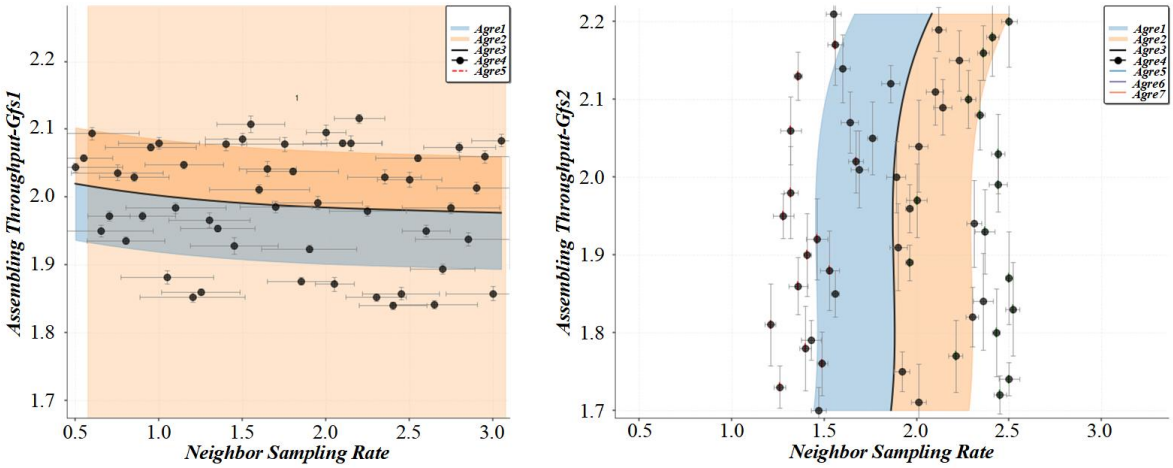

Figure 10: Multi-order neighborhood sampling coverage change evaluation diagram

The GPU pipeline mechanism enables each calculation stage to be executed concurrently, showing a strong resource synergy effect in the training process. Compared with the unoptimized traditional GraphSAGE method, Figure 11 is the knowledge graph node distribution evaluation diagram. Under the same data scale and hardware environment, this system's overall throughput has increased by 30% to 60%. The specific improvement range depends on the node degree distribution and neighborhood sampling depth.

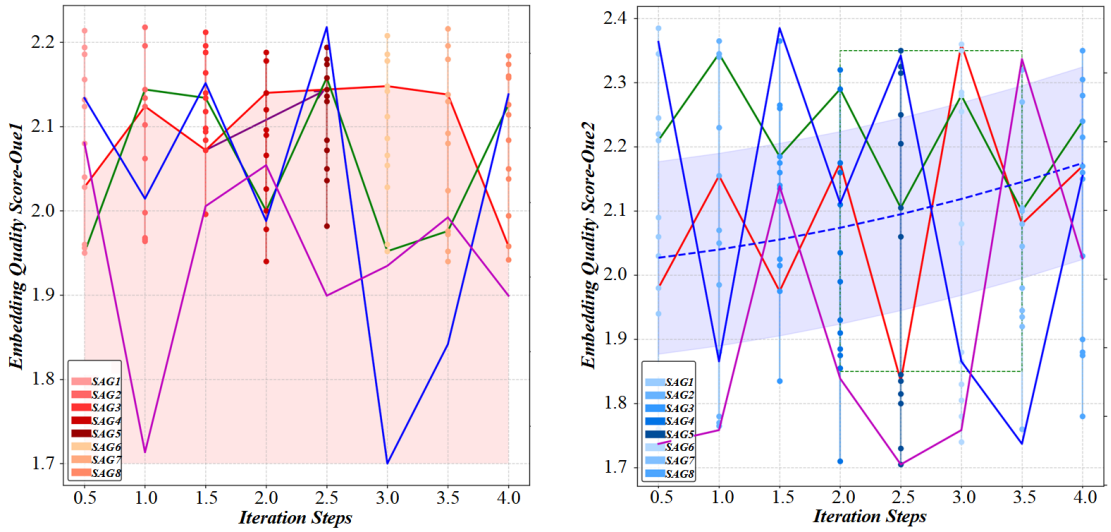

Figure 11: Knowledge graph node distribution evaluation diagram

6 Conclusion

This paper constructs a set of high-performance assembly methods for industrial knowledge graph scenarios through the comprehensive research of graph structure feature modeling, bottleneck analysis of the calculation process, and system implementation strategy. The toolchain has significantly improved assembly speed, embedding quality, and resource utilization, effectively alleviating the performance and stability problems faced by the original GraphSAGE when processing complex map data.

Regarding neighborhood sampling, traditional GraphSAGE relies mostly on fixed depth and random methods, making it difficult to balance efficiency and information coverage in multi-level semantic modeling. This paper proposes a multi-level neighborhood-aware sampling method based on dynamic hybrid indexing, which realizes strategic semantic neighborhood construction through node importance prediction. The GPU-accelerated Hard Negative sample online mining strategy is introduced, parallel computing is used to optimize the sample screening process, and the loss function penalty weight is dynamically adjusted by combining the semantic repulsion mechanism during the training process, which effectively improves the model's ability to distinguish semantic boundaries.

Faced with the problem of I/O bottleneck and idle GPU computing resources, this paper constructs a non-blocking asynchronous I/O and GPU computing pipeline deep binding mechanism, which realizes the decoupling and overlapping processing of data and computing through multi-threaded asynchronous data loading combined with GPU multi-stream scheduling. In actual deployment, this strategy effectively avoids the GPU idling problem caused by I/O blocking and significantly improves the effective computing time during the training cycle. Aiming at the feature aggregation operation of large-scale graphs, combined with the dynamics of heterogeneous graph structures, the system automatically adjusts the data block size and task parallel strategy at different stages, realizes the dynamic adaptation of resource allocation between high-density areas and sparse areas, and improves the overall training efficiency.

The performance improvement of optimization methods in distributed environments is particularly prominent. In a cluster environment of 100 GPUs, the data loading speed is increased by 24.6%, the model training speed is increased by 5.2 times, and the data transmission latency is reduced by 68%. The cross-partition edge processing error is reduced from 12.8% to 5.2%, and the communication overhead between nodes is reduced by 47.1%. In the large-scale knowledge graph data assembly task, the overall performance of the optimized toolchain system improved by 83.4%, and the multi-GPU parallel speed-up ratio reached 47.1, indicating that the work-stealing load balancing model can effectively solve the problem of uneven allocation of computing resources and ensure that the system is high Stability and scalability under load scenarios.

References

[1] Z. Z. Li, J. Y. Ma, R. Fan, Y. M. Zhao, J. L. Ai, and Y. Q. Dong, "Aircraft Sensor Fault Diagnosis Based on GraphSage and Attention Mechanism," Sensors, vol. 25, no. 3, 2025. doi: 10.3390/s25030809.

[2] L. J. Xu, Z. C. Zhao, D. W. Zhao, X. Li, X. Y. Lu, and D. Y. Yan, "AJSAGE: A intrusion detection scheme based on Jump-Knowledge Connection To GraphSAGE," Computers & Security, vol. 150, 2025. doi: 10.1016/j.cose.2024.104263.

[3] Y. H. Xiao, C. Z. Li, Z. W. Zhou, D. Y. Hou, and X. G. Zhou, "Analysis and Optimization of the Spatial Patterns of Commercial Service Facilities Based on Multisource Spatiotemporal Data and Graph Neural Networks: A Case Study of Beijing, China," Isprs International Journal of

Geo-Information, vol. 14, no. 1, 2025. doi: 10.3390/ijgi14010023.

[4] K. Syama, J. A. A. Jothi, and N. Khanna, "Automatic disease prediction from human gut metagenomic data using boosting GraphSAGE," Bmc Bioinformatics, vol. 24, no. 1, 2023. doi: 10.1186/s12859-023-05251-x.

[5] F. Zouak, O. El Beqqali, and J. Riffi, "BERT-GraphSAGE: hybrid approach to spam detection," Journal of Big Data, vol. 12, no. 1, 2025. doi: 10.1186/s40537-025-01176-9.

[6] C. Y. Hu, B. Ning, Q. Gu, J. F. Qu, S. Jeon, and B. W. Du, "Big data analytics-based traffic flow forecasting using inductive spatial-temporal network," Environment Development and Sustainability, vol., 2022. doi: 10.1007/s10668-022-02585-z.

[7] K. Noreika and S. Gudas, "Causal Knowledge Modelling for Agile Development of Enterprise Application Systems," Informatica, vol. 34, no. 1, pp. 121-146, 2023. doi: 10.15388/23-infor510.

[8] Y. Cui, C. Shao, L. Luo, L. G. Wang, S. Gao, and L. W. Chen, "Center Weighted Convolution and GraphSAGE Cooperative Network for Hyperspectral Image Classification," Ieee Transactions on Geoscience and Remote Sensing, vol. 61, 2023. doi: 10.1109/tgrs.2023.3264653.

[9] M. A. Anitha and K. K. Sherly, "Churn prediction with GraphSAGE model based on the derived features using RFM and sentiment analysis," Journal of the Chinese Institute of Engineers, vol. 48, no. 4, pp. 441-453, 2025. doi: 10.1080/02533839.2025.2478185.

[10] M. Vaida et al., "M-GNN: A Graph Neural Network Framework for Lung Cancer Detection Using Metabolomics and Heterogeneous Graph Modeling," International Journal of Molecular Sciences, vol. 26, no. 10, 2025. doi: 10.3390/ijms26104655.

[11] S. Saidane, F. Telch, K. Shahin, and F. Granelli, "Deep GraphSAGE enhancements for intrusion detection: Analyzing attention mechanisms and GCN integration," Journal of Information Security and Applications, vol. 90, 2025. doi: 10.1016/j.jisa.2025.104013.

[12] D. El Alaoui, J. Riffi, A. Sabri, B. Aghoutane, A. Yahyaouy, and H. Tairi, "Deep GraphSAGE-based recommendation system: jumping knowledge connections with ordinal aggregation network," Neural Computing & Applications, vol. 34, no. 14, pp. 11679-11690, 2022. doi: 10.1007/s00521-022-07059-x.

[13] H. V. Ribeiro et al., "Deep learning criminal networks," Chaos Solitons & Fractals, vol. 172, 2023. doi: 10.1016/j.chaos.2023.113579.

[14] J. W. Li, X. K. Zhang, B. Li, Z. Y. Li, and Z. Z. Chen, "MDFGNN-SMMA: prediction of potential small molecule-miRNA associations based on multi-source data fusion and graph neural networks," Bmc Bioinformatics, vol. 26, no. 1, 2025. doi: 10.1186/s12859-025-06040-4.

[15] Z. Yang, Y. X. Chen, J. B. Hu, Y. L. Song, and Y. Mao, "Departure delay prediction and analysis based on node sequence data of ground support services for transit flights," Transportation Research Part C-Emerging Technologies, vol. 153, 2023. doi: 10.1016/j.trc.2023.104217.

[16] M. Zhang, X. J. Li, Z. Y. Xiang, J. L. Mo, and S. H. Xu, "Diagnosis of brake friction faults in high-speed trains based on 1DCNN and GraphSAGE under data imbalance," Measurement, vol. 207, 2023. doi: 10.1016/j.measurement.2022.112378.

[17] B. T. Zhai, D. S. Yang, B. W. Zhou, and G. D. Li, "Distribution System State Estimation Based on Power Flow-Guided GraphSAGE," Energies, vol. 17, no. 17, 2024. doi: 10.3390/en17174317.

[18] A. Kilciauskas, A. Bendoraitis, and E. Sakalauskas, "Confidential Transaction Balance Verification by the Net Using Non-Interactive Zero-Knowledge Proofs," Informatica, vol. 35, no. 3, pp. 601-616, 2024. doi: 10.15388/24-infor564.

[19] L. Q. Lin, Q. Zhong, J. S. Qiu, and Z. Y. Liang, "E-GRACL: an IoT intrusion detection system based on graph neural networks," Journal of Supercomputing, vol. 81, no. 1, 2025. doi: 10.1007/s11227-024-06471-5.

[20] J. Lan et al., "E-minBatch GraphSAGE: An Industrial Internet Attack Detection Model," Security and Communication Networks, vol. 2022, 2022. doi: 10.1155/2022/5363764.

[21] Y. H. Bai et al., "Efficient Data Loader for Fast Sampling-Based GNN Training on Large Graphs," Ieee Transactions on Parallel and Distributed Systems, vol. 32, no. 10, pp. 2541-2556, 2021. doi: 10.1109/tpds.2021.3065737.

[22] Y. Afoudi, M. Lazaar, and S. Hmaidi, "An enhanced recommender system based on heterogeneous graph link prediction," Engineering Applications of Artificial Intelligence, vol. 124, 2023. doi: 10.1016/j.engappai.2023.106553.

[23] M. Fan, J. L. Xia, H. J. Zhang, and X. Zhang, "Fault Location Method of Distribution Network Based on VGAE-GraphSAGE," Processes, vol. 12, no. 10, 2024. doi: 10.3390/pr12102179.

[24] A. Tonks, T. Harris, B. Li, W. Brown, and R. Smith, "Forecasting West Nile Virus With Graph Neural Networks: Harnessing Spatial Dependence in Irregularly Sampled Geospatial Data," Geohealth, vol. 8, no. 7, 2024. doi: 10.1029/2023gh000784.

[25] J. W. Lu, L. R. Zheng, X. M. Hua, and Y. K. Wang, "Graph Convolution for Large-Scale Graph Node Classification Task Based on Spatial and Frequency Domain Fusion," Ieee

Access, vol. 13, pp. 16787-16799, 2025. doi: 10.1109/access.2025.3532806.

[26] Q. M. Wei, J. Y. Wang, J. Hu, X. X. Li, and T. Yi, "OGT: optimize graph then training GNNs for node classification," Neural Computing & Applications, vol. 34, no. 24, pp. 22209-22222, 2022. doi: 10.1007/s00521-022-07677-5.

[27] T. Liu, A. M. Jiang, J. Zhou, M. Li, and H. K. Kwan, "GraphSAGE-Based Dynamic Spatial-Temporal Graph Convolutional Network for Traffic Prediction," Ieee Transactions on Intelligent Transportation Systems, vol. 24, no. 10, pp. 11210-11224, 2023. doi: 10.1109/tits.2023.3279929.

[28] P. Lu, C. F. Jing, and X. R. Zhu, "GraphSAGE-Based Multi-Path Reliable Routing Algorithm for Wireless Mesh Networks," Processes, vol. 11, no. 4, 2023. doi: 10.3390/pr11041255.

[29] J. L. Liu, P. Ong, and X. Q. Chen, "GraphSAGE-Based Traffic Speed Forecasting for Segment Network with Sparse Data," Ieee Transactions on Intelligent Transportation Systems, vol. 23, no. 3, pp. 1755-1766, 2022. doi: 10.1109/tits.2020.3026025.

[30] P. J. Ma et al., "Hardware Trojan Detection Methods for Gate-Level Netlists Based on Graph Neural Networks," Ieee Transactions on Computers, vol. 74, no. 5, pp. 1470-1481, 2025. doi: 10.1109/tc.2025.3533085.