Poligras: Policy-based Graph Summarization

Jiyang Bai
Florida State University
Tallahassee, Florida, U.S.A.
bai@cs.fsu.edu

Peixiang Zhao
Florida State University
Tallahassee, Florida, U.S.A.
zhao@cs.fsu.edu

ABSTRACT

Large graphs are ubiquitous. Their sizes, rates of growth, and complexity, however, have significantly outpaced human capabilities to ingest and make sense of them. As a cost-effective graph simplification technique, *graph summarization* is aimed to reduce large graphs into concise, structure-preserving, and quality-enhanced summaries readily available for efficient graph storage, processing, and visualization. Concretely, given a graph G, graph summarization condenses G into a succinct representation comprising (1) a *supergraph* with supernodes representing disjoint sets of vertices of G and superedges depicting aggregate-level connections between supernodes, and (2) a set of *correction* edges that help reconstruct G losslessly from the supergraph. Existing graph summarization solutions offer non-optimal graph summaries and are time-demanding in real-world large graphs. In this paper, we propose a learning-enhanced graph summarization approach, Poligras (**Poli**cy-based **gra**ph **s**ummarization), to model the most critical computational component in graph summarization: supernode selection and merging. Specifically, we design a probabilistic policy learned and optimized by neural networks for efficient optimal supernode pair selection. As the first learning-enhanced, scalable graph summarization method, Poligras achieves significantly improved performance over state-of-the-art graph summarization solutions in real-world large graphs.

PVLDB Reference Format:
Jiyang Bai and Peixiang Zhao. Poligras: Policy-based Graph Summarization. PVLDB, 17(10): 2432 - 2444, 2024.
doi:10.14778/3675034.3675037
PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/jiyangbai/Poligras.

1 INTRODUCTION

Recent years have witnessed the ubiquity of graphs and networks that pervade the natural, technological, and societal worlds surrounding us. Their unprecedented sizes, rates of growth, and complexity, however, have significantly surpassed the storage, computation, and communication capacity at our disposal. Worse yet, human capabilities to ingest and make sense of such complex data have not scaled accordingly, rendering big graph management a daunting task [11, 35]. To address this challenge, it is desirable to *summarize* big graphs into concise, structure-preserving, and quality-enhanced *summaries* that are readily amenable for efficient, cost-effective, and scalable graph storage and computation [45]. As a remarkably useful methodology for graph data simplification and reduction, graph summarization has imposed fundamental impacts on widely varying real-world applications [19]:

(1) Graph summaries provide compact structure representations that incur substantially smaller overhead than original graphs. In many occasions, graph summaries can naturally fit in memory or cache, contributing to a significant reduction to I/O and communication costs in graph computation [18, 21, 37, 45];
(2) Utility- or query-driven graph summaries retain salient and query-relevant information of original graphs, which speeds up query processing in large graphs [8–10, 12, 20];
(3) By generating a series of graph summaries with varied sizes and resolutions, graph summarization facilitates exploratory studies and interactive visualization for big graphs [2, 14, 19, 42, 46];
(4) Real-world graphs are fraught with distorted, spurious information. Graph summarization helps filter such noise by preserving crucial and task-relevant information only from massive graphs [15, 43].

In this paper, we study a fundamental graph summarization scheme that has sparked lasting interest due in particular to its generality, flexibility, and wide applicability [18, 31, 37, 45]. Given a graph $G = (V, E)$, graph summarization is aimed to identify a compact representation of G consisting of (1) a summary graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, and (2) an edge correction set C. Specifically, $\mathcal{G}$ is a *supergraph* with a *supernode* set $\mathcal{V}$ (each supernode $\mu \in \mathcal{V}$ represents a disjoint subset of vertices of G), and a superedge set $\mathcal{E}$ (a superedge $(\mu, \nu) \in \mathcal{E}$ symbolizes *all* the possible connections, not necessarily actual edges of G, between the vertices within supernodes μ and ν, respectively). The intuition is to exploit the connectivity proximity in real-world graphs: if two vertices u and v connect to the same, or a very similar, set of other vertices in G, they can be coalesced to a common supernode. Furthermore, we integrate incident edges of u and v connecting to their common neighbors as a single superedge to represent a *group-level* connection between supernodes. In addition, the edge correction set C keeps track of actual edges of G that need to be inserted to, or removed from, the summary graph $\mathcal{G}$ towards a lossless reconstruction of the original graph G. Compliant with the minimum description length (MDL) principle, this summarization scheme creates succinct, coarse-grained graph summaries with desirable properties:

(1) **Generality.** Although initially proposed for simple, undirected graphs [31], this general-purpose graph summarization scheme can be extended to directed, attributed, or heterogeneous graphs. In addition, the resultant supergraph $\mathcal{G}$ and the edge correction set C can be further summarized or compressed by other data summarization techniques;

Proceedings of the VLDB Endowment, Vol. 17, No. 10 ISSN 2150-8097.
doi:10.14778/3675034.3675037

(2) **Flexibility.** Graph summarization admits a *lossless* compression of the input graph G, which can be reconstructed from $\mathcal{G}$ and C without information loss. Furthermore, by regulating an error rate ϵ to drop part of the information in $\mathcal{G}$ and C, we can immediately enable *lossy* graph summarization that trades off between the graph summary size and the tolerable amount of information loss [31];
(3) **Neighborhood-preservation.** Graph summaries retain the complete neighborhood information about vertices. Therefore, neighborhood queries and their variants, such as DFS and BFS based algorithms, can be addressed directly on graph summaries, as opposed to the original big graph G.

State-of-the-art solutions, such as SWeG [37] and LDME [45], typically follow a two-stage algorithmic paradigm for graph summarization. At the first, *supernode selection and merging* stage, a set of supernode pairs are identified and merged in succession to larger-size supernodes; At the second, *edge-encoding* stage, edges of G are either aggregated as superedges to account for group-level connections between supernodes, or explicitly maintained in the edge correction set C. The performance-critical operation here is *supernode selection*, which determines, at each step of graph summarization, the optimal pair of supernodes, the merge of which will lead to the maximum size reduction to the resultant graph summary. Existing solutions consider approximate metrics to quantify the *summarization effectiveness* for supernode selection and merging, which are inaccurate to characterize the sheer size reductions to graph summaries. As exact computation of such metrics is time-consuming, SWeG and LDME instead consider approximation and randomized algorithms for supernode selection and merging, which, however, lack theoretical performance guarantees. In sum, SWeG and LDME generate excessively large, non-optimal graph summaries, and are time-consuming when employed in real-world, large graphs.

Inspired by recent advances in online and reinforcement learning, we propose in this paper a learning-enhanced graph summarization method, Poligras (**Poli**cy-based **gra**ph **s**ummarization). In Poligras, the crucial step of supernode selection is modeled as a sequential, probabilistic decision-making process, in which optimal supernode pairs are learned and identified with an objective to minimizing the ultimate graph summary cost. We introduce in Poligras a novel notion of *summarization reward* to quantify the sheer size reduction between consecutive graph summaries subject to a supernode-pair merging, and prove that selecting the supernode pairs with maximum summarization rewards will lead to a graph summary with the minimum summarization cost, which is the prime goal for graph summarization. To facilitate optimal supernode selection in Poligras, we propose a probabilistic *supernode selection policy* that is modeled, learned, and optimized by a multi-layer neural network. As the first learning-enhanced graph summarization approach, Poligras proves to be a highly effective, efficient, and scalable method that achieves significantly improved performance for summarizing real-world large graphs when compared with the state-of-the-art solutions, SWeG and LDME. Specifically, the key contributions of Poligras are outlined as follows:

(1) We reformulate the problem of graph summarization as a policy learning and optimization problem, where summarizing a graph can be considered as making a sequence of supernode selection and merging decisions under the guidance of an optimized supernode selection policy. To this end, Poligras establishes a new, learning-enhanced solution for the classic graph summarization problem (Section 3);
(2) We introduce a new notion of summarization reward to quantify the merging effectiveness of a pre-selected supernode pair. We further prove that iteratively selecting the supernode pairs with maximum summarization rewards will lead to the graph summary with the minimum summarization cost, which is the prime goal for graph summarization (Section 3.1);
(3) We design a probabilistic supernode selection policy that is trained and optimized by multi-layer neural networks in order to identify the promising supernode pairs with high summarization rewards (Section 3.2);
(4) We carry out systematic experimental studies on ten real-world graph datasets, and the results validate the significant advantages of Poligras, in comparison with the state-of-the-art graph summarization solutions, in terms of graph summarization effectiveness, efficiency, and scalability (Section 4).

Table 1: A Notation Primer

Symbol	Definition
$G = (V, E)$	A graph with vertex set V and edge set E
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	A supergraph with supernode set $\mathcal{V}$ and superedge set $\mathcal{E}$
$\mathcal{S} = (\mathcal{G}, C)$	A graph summary with supergraph $\mathcal{G}$ and edge correction set C
$\mathcal{P}$	A set of partitioned supernode groups
$r(\mu, v)$	The summarization reward of merging supernodes μ and v
$\mathbf{H}_\mu$	The supernode embedding for supernode μ
$\mathbb{P}(\cdot)$	The policy function
$\mathbf{M}_P$	The selection probability matrix for group $P \in \mathcal{P}$
T	The number of iterations for graph summarization

The remainder of this paper is organized as follows: In Section 2, we introduce the key definitions, preliminary concepts, and a generic algorithm framework for graph summarization. In Section 3, we discuss in detail our learning-enhanced solution, Poligras. We report experimental results and key findings in Section 4. In Section 5, we brief related work for graph summarization, followed by concluding remarks in Section 6. For ease of reference, We provide a list of notations used in Poligras in Table 1.

2 AN ALGORITHMIC FRAMEWORK FOR GRAPH SUMMARIZATION

2.1 Problem Formulation

We consider as input a simple, undirected graph $G = (V, E)$ for graph summarization, where V is a set of vertices, and $E \subseteq V \times V$ is a set of edges. Starting from G, a *supergraph* $\mathcal{G}$ is defined as follows,

Definition 1. **[SUPERGRAPH]** *A supergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consists of a supernode set $\mathcal{V}$ and a superedge set $\mathcal{E}$: Each supernode $\mu \in \mathcal{V}$ represents a subset V_μ of vertices in G, $V_\mu \subseteq V$, such that $\forall \mu, \nu \in \mathcal{V}$, $V_\mu \cap V_\nu = \emptyset$, and $\bigcup_{\mu \in \mathcal{V}} V_\mu = V$; Each superedge $(\mu, \nu) \in \mathcal{E}$ between supernodes μ and ν represents all the pairwise connections, not necessarily actual edges, between the plain vertices $u \in \mu$ and $v \in \nu$, where $u, v \in V$.* □

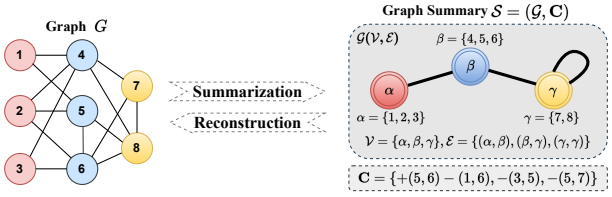

Figure 1: A graph G and its graph summary $\mathcal{S} = (\mathcal{G}, C)$.

Given a supergraph $\mathcal{G}$, its supernode set $\mathcal{V}$ represents a mutually exclusive, collectively exhaustive partition for the vertex set V of the graph G. Consider two vertices $u, v \in V$. If they are similar in terms of neighborhood connectivity: they connect with the same, or a highly similar, set of adjacent vertices in G, u and v can thus be coalesced into the same supernode of $\mathcal{G}$. Likewise, all the edges incident to u, v, and their common neighbors can be consolidated into a superedge depicting an aggregate, group-level connection between two sets of vertices represented by the corresponding supernodes. Consequently, the supergraph $\mathcal{G}$ is a compact graph representation that encodes dominant connectivity patterns and coarse-grained structural insights from the original graph G.

To account for the actual edges of G that is encoded by a superedge (μ, ν) between supernodes μ and ν, we define by $\Pi_{\mu\nu}$ the set of all vertex pairs (u, v), where $u \in \mu$, $v \in \nu$, and $u, v \in V$; Equivalently, $\Pi_{\mu\nu}$ represents all the potential edges that may possibly be present across supernodes μ and ν, and it is immediate that $|\Pi_{\mu\nu}| = |V_\mu| \cdot |V_\nu|$[1]. We further denote by $E_{\mu\nu}$ the set of actual edges present in the graph G; that is, $E_{\mu\nu} = \Pi_{\mu\nu} \cap E$. Once a superedge (μ, ν) is present in $\mathcal{G}$, we may falsely introduce a set $\Pi_{\mu\nu} \setminus E_{\mu\nu}$ of *spurious edges*, each of which is denoted by "$-(u, v)$", where $u \in \mu, v \in \nu$, and the minus sign indicates (u, v) is a spurious edge not actually present in G. Likewise, if there exists no superedge (μ, ν) in $\mathcal{G}$, we may miss actual edges of $E_{\mu\nu}$ in the summary, thus rendering an information loss. Here we use "$+(u, v)$" to denote a *missing edge*, where the plus sign indicates the edge needs to be reclaimed to $\mathcal{G}$. To warrant correctness for graph summarization, we introduce an *edge correction* set, $C \subseteq V \times V$, that maintains all spurious and missing edges. To this end, the graph summary can be defined below:

Definition 2. **[GRAPH SUMMARY]** *Given a graph G, its graph summary, $\mathcal{S} = (\mathcal{G}, C)$, consists of a supergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and an edge correction set C.* □

Consider a graph summary $\mathcal{S} = (\mathcal{G}, C)$. The supergraph $\mathcal{G}$ highlights the essential grouping characteristics (via supernodes) and dense connectivity patterns (via superedges) in G, and the edge correction set C allows for *lossless reconstruction* of G from $\mathcal{G}$. In particular, we can reconstruct the original graph G (or a subgraph of G) from the graph summary $\mathcal{S}$ as follows: (1) For each supernode μ of $\mathcal{G}$, we unfold it to a subset V_μ of constituent vertices of G; (2) For each superedge (μ, ν) of $\mathcal{G}$, we expand it by adding pairwise edges (u, v) for any $u \in \mu$ and $v \in \nu$; (3) We scan the edge correction set C by either eliminating a spurious edge, $-(u, v)$, or inserting a missing edge, $+(u, v)$. To this end, the graph summary $\mathcal{S}$ can be regarded as a structure-preserving, information-lossless compression of the graph G. To further condense the graph summary $\mathcal{S}$, we can selectively remove a subset of edges from the edge correction set C, or even part of the supergraph $\mathcal{G}$, thus yielding a *lossy summary* $\mathcal{S}'$, regulated by an error parameter, ϵ [31]. However, as lossy summarization is typically a post-processing step once the graph summary $\mathcal{S}$ is available, in this paper, we focus primarily on lossless graph summarization ($\epsilon = 0$), which is also the target of the state-of-the-art solutions [37, 45]

[1]The self-superedge (μ, μ) is allowed in a supergraph, and in this case, $|\Pi_{\mu\mu}| = (|V_\mu| - 1) \cdot |V_\mu|/2$.

Algorithm 1 The Graph Summarization Framework

Input: A graph $G = (V, E)$, the number of iterations, T, for summarization
Output: The graph summary $\mathcal{S} = (\mathcal{G}, C)$

```
V ← {{v} | v ∈ V}
for k = 1, 2, ..., T do
    Partition V into a set P of disjoint groups          ▷ Group Partitioning
    for each supernode group P_i ∈ P do                  ▷ Supernode Merging
        Merge a selected pair of supernodes μ, ν into μ ∪ ν ▷ Algorithm 2
        V ← (V \ {μ, ν}) ∪ {μ ∪ ν}
Encode edges of G into E and C                           ▷ Edge Encoding
return S = (G, C)
```

Example 1. *Consider a graph G with 8 vertices and 14 edges as shown in Figure 1. We summarize G to a graph summary $\mathcal{S} = (\mathcal{G}, C)$, where $\mathcal{G}$ is a supergraph with 3 supernodes and 3 superedges, including a self-superedge (γ, γ); C is the edge correction set containing 1 missing edge and 3 spurious edges. Note that the graph summary $\mathcal{S}$ is significantly smaller than G. Furthermore, we can reconstruct the original graph G from $\mathcal{S}$ without information loss.* □

The graph summarization scheme in Definition 2 embraces the principle of minimum description length (MDL) as its theoretical underpinning [31]. According to MDL [34], the potentially best theory to infer from a dataset is the one that minimizes the total sizes of (1) the theory per se, and (2) the data encoded by the theory. In the case of graph summarization, the data to be summarized is the input graph G, and the theory is the supergraph $\mathcal{G}$. The edge correction set C is the encoding of G in terms of $\mathcal{G}$. To this end, we define the *cost* of a graph summary $\mathcal{S} = (\mathcal{G}, C)$ as[2]

$$cost(\mathcal{S}) = |\mathcal{E}| + |C|. \tag{1}$$

The first term $|\mathcal{E}|$ corresponds to the size of the theory (supergraph $\mathcal{G}$), while the second term $|C|$ is the encoding size of G *w.r.t.* $\mathcal{G}$. The objective of graph summarization is therefore to identify the min-cost graph summary, $\mathcal{S}^*$, from G.

Definition 3. **[GRAPH SUMMARIZATION]** *Given an input graph G, the problem of graph summarization is to find as output the min-cost graph summary $\mathcal{S}^*$, where $\mathcal{S}^* = \arg\min_{\mathcal{S}}(cost(\mathcal{S})) = \arg\min_{\mathcal{S}}\{|\mathcal{E}| + |C|\}$.* □

2.2 The Algorithmic Framework

We present in Algorithm 1 a generic algorithm framework for graph summarization, in which the state-of-the-art solutions, such

[2]We omit $|\mathcal{V}|$ and the cost of supernode-to-vertex mappings, $V_\mu = \{v_\mu^1, v_\mu^2, \ldots\}$ where $v_\mu^i \in V$ for all $\mu \in \mathcal{V}$, as they are significantly smaller than $|\mathcal{E}|$.

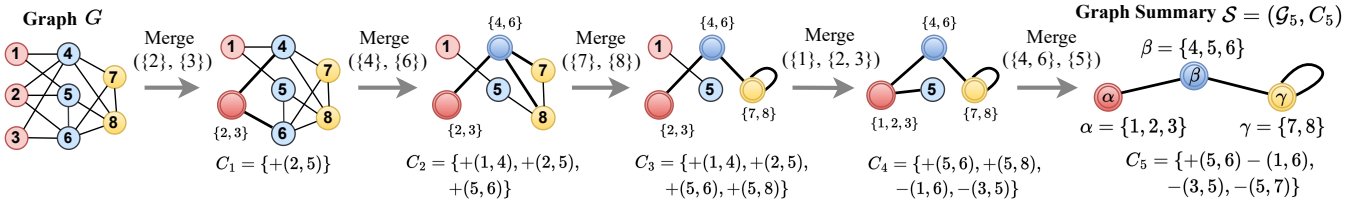

Figure 2: The iterative process of graph summarization from G to its ultimate graph summary $\mathcal{S} = (\mathcal{G}_5, C_5)$

as SWeG [37] and LDME [45], naturally fit. Given an input graph G, we start by initializing each vertex of G as a trivial supernode (Line 1), and iteratively select supernode pairs to merge in a bottom-up, greedy fashion (Lines 2-6). Specifically, in each iteration, the supernodes are first partitioned into disjoint groups based on their connectivity proximity (Line 3). Within each group, a pair of supernodes μ and v, which, if merged, will lead to a significant size reduction to graph summaries, are identified and merged into a new supernode, denoted $\mu \cup v$ (Lines 4-6). When the supernodes set $\mathcal{V}$ is finalized after T iterations, the superedges, together with the edge correction set C, are further generated to accomplish the ultimate graph summary $\mathcal{S}$ (Line 7).

Example 2. *Given the graph G in Figure 1, we present the iterative process of graph summarization that simplifies G to its graph summary $\mathcal{S}$ in Figure 2. First of all, all the vertices of G are partitioned into three groups colored in red, blue, and yellow. At each step of graph summarization, a pair of supernodes are selected from within the same group, and merged into a new supernode. This supernode selection and merging process continues for a certain number of iterations $T = 5$. In addition, the edges of G can be either aggregated as superedges, or maintained in the edge correction set C. We thus have a series of intermediate graph summaries, and the ultimate one is $\mathcal{S} = (\mathcal{G}_5, C_5)$.* □

In what follows, we look into three performance-critical steps for graph summarization as sketched in Algorithm 1.

1. **Group Partitioning (Line 3).** In this step, the supernode set $\mathcal{V}$ is partitioned to a set $\mathcal{P}$ of disjoint groups, each of which contains supernodes with similar neighborhood proximity. The main objective here is to confine the subsequent, time-demanding supernode selection and merging within each group $P_i \in \mathcal{P}$, as opposed to upon the whole supernode set $\mathcal{V}$. Additionally, this enables parallel execution of supernode merging across separate partitions, leading to further speedup for graph summarization. In particular, SWeG chooses *Jaccard coefficient* to quantify the neighborhood proximity of supernodes, and uses *shingling* and *minhashing* techniques [7] for supernode partitioning. LDME considers the *SuperJaccard* similarity, a min-max variant of Jaccard coefficient, and uses weighted *locality sensitive hashing* (LSH) to partition $\mathcal{V}$ [45]. It is reported that the weighted LSH method typically leads to a greater number of smaller-size groups than minhashing [45]. As a result, LDME is more precise than SWeG in supernode partitioning: supernodes with high SuperJaccard similarity are guaranteed, *w.h.p.*, to be partitioned to the same group.

2. **Supernode Selection and Merging (Lines 4-6).** In each supernode partition $P_i \in \mathcal{P}$, we identify a pair of supernodes, μ and v, and merge them to a new supernode, $\mu \cup v$. The goal here is to choose among $O(|P_i|^2)$ supernode pairs an *optimal* one, the merge of which leads to a maximum reduction to edges/superedges in the graph summary $\mathcal{S}$. In each iteration, however, there exist $O(\sum_{i=1}^{|\mathcal{P}|} |P_i|^2)$ supernode pairs under investigation, which poses a significant performance barrier for graph summarization [31, 37, 45]. Furthermore, if not carefully designed, the supernode pair selection may result in notable inaccuracy for the graph summary cost (Equation 1). Worse yet, such inaccuracy can be iteratively amplified, thereby yielding excessively large, non-optimal graph summaries.

Existing solutions, such as SWeG and LDME, propose the notion of saving(μ, v) to quantify the potential *benefit* of merging supernodes μ and v in graph summarization. Formally, saving(μ, v) is the *ratio* of cost reduction as a result of merging toward a new supernode $\mu \cup v$ over the combined cost of μ and v before the merge. As an exact computation of saving(μ, v) is costly, SWeG adopts the *SuperJaccard* similarity of μ and v as an approximation. To further lower the computation cost, SWeG considers a randomized algorithm that first picks a supernode μ uniformly at random, and then selects μ's best possible merging candidate v to approximate SuperJaccard. In contrast, LDME develops a two-level hash table to compute saving(μ, v), which incurs significant space overhead.

In sum, existing solutions consider the approximate metric, saving(μ, v), as opposed to the actual graph summary cost (Equation 1), for supernode selection. Their approximation algorithms also lack theoretical performance guarantees. Consequently, the estimated metric, saving, oftentimes deviates from the goal of graph summarization (Definition 3), rendering suboptimal supernodes selected at each step, and non-optimal graph summary in the end.

3. **Edge Encoding (Line 7).** Once the supernode set $\mathcal{V}$ is determined, we further encode the edges of G into the superedge set, $\mathcal{E}$, or the edge correction set, C. Consider any pair of supernodes $\mu, v \in \mathcal{V}$. There exist two options to encode the actual edges of $E_{\mu v}$ between supernodes μ and v:

(1) We create a superedge (μ, v) in $\mathcal{G}$. As a consequence, all the spurious edges, $-(u, v)$, need to be added to the edge correction set C for $u \in \mu$ and $v \in v$. The cost incurred is $(1 + |\Pi_{\mu v}| - |E_{\mu v}|)$;
(2) We add all the missing edges, $+(u, v)$, to the edge correction set C without creating a superedge between μ and v, where $(u, v) \in E_{\mu v}$. This way, the cost incurred is $|E_{\mu v}|$;

We then choose the one with the smaller *superedge cost*:

$$cost(\mu, v) = \min\{1 + |\Pi_{\mu v}| - |E_{\mu v}|, |E_{\mu v}|\} \quad (2)$$

to encode actual edges of $E_{\mu v}$: When $|E_{\mu v}| > (1 + |\Pi_{\mu v}|)/2$, the superedge (μ, v) is present in $\mathcal{G}$, and all spurious edge in $\Pi_{\mu v} \setminus E_{\mu v}$ are inserted to the edge correction set C; Otherwise, the missing

edges of $E_{\mu\nu}$ are directly inserted to C. This edge encoding principle minimizes the number of superedges/edges in the graph summary $\mathcal{S}$, while still warranting the correctness and completeness of edge connectivity in the original graph. Therefore, when $\mathcal{V}$ is determined, it is straightforward to compute $\mathcal{E}$, C, and also the ultimate graph summary $\mathcal{S}$ by investigating every supernode pair for edge encoding as discussed above. As a result, to identify the min-cost graph summary $\mathcal{S}^*$ (Definition 3), it is essential to find the optimal supernode set, $\mathcal{V}^*$, which is the prime goal of our work.

3 POLIGRAS

To address the weaknesses of existing graph summarization solutions, and more importantly, to explore a systematic approach for optimal supernode pair selection in graph summarization, we propose in this paper a learning-enhanced approach, Poligras (**Poli**cy-based **gra**ph **s**ummarization). In Poligras, the critical step for supernode selection and merging is modeled as a sequential decision-making process. We introduce the notion of *summarization reward* to quantify the merging effectiveness for a pair of supernodes, and demonstrate that selecting supernode pairs with maximum summarization rewards will result in the min-cost graph summary, which is the objective of graph summarization (Section 3.1). To facilitate the computation for supernode selection, we propose a probabilistic supernode selection policy that is modeled, learned, and optimized by multi-layer neural networks (Section 3.2). Poligras is able to identify supernode pairs with significantly reduced summarization cost, thus achieving improved performance for graph summarization especially in large graphs.

3.1 Summarization Reward

In Poligras, graph summarization is driven by a sequence of n supernode-merging operations, each of which is denoted by (μ_i, ν_i) $(0 \le i \le n-1)$, indicating merging supernodes μ_i and ν_i selected from the supergraph $\mathcal{G}_i$:

$$\mathcal{G}_0, C_0 \xrightarrow{(\mu_0,\nu_0)} \mathcal{G}_1, C_1 \xrightarrow{(\mu_1,\nu_1)} \dots \xrightarrow{(\mu_{n-1},\nu_{n-1})} \mathcal{G}_n, C_n$$

At the beginning, the input graph $G = (V, E)$ is abstracted as an initial supergraph $\mathcal{G}_0 = (\mathcal{V}_0, \mathcal{E}_0)$, where $\mathcal{V}_0 = \{\{v\}|v \in V\}$ and $\mathcal{E}_0 = \{(\{u\},\{v\})|(u,v) \in E\}$; Correspondingly, the edge correction set C_0 is initialized to $\emptyset$. We then go through an iterative process of supernode merging. At each step, we select a pair of supernodes (μ_i, ν_i) from $\mathcal{G}_i$ (The details of supernode selection are elaborated in Section 3.2), and merge them to a new supernode. As a consequence, the supergraph $\mathcal{G}_i$ is summarized to $\mathcal{G}_{i+1}$, and the edge correction set C_i is updated to C_{i+1}. This supernode-merging process is detailed in Algorithm 2. First of all, we create a new supernode ρ that encompasses all the plain vertices within μ_i and ν_i (Line 1). Consider an arbitrary supernode μ of $\mathcal{G}_i$. We define its *neighboring supernode set*, $N_{\mathcal{G}_i}[\mu] = \{\eta|(\eta,\mu) \in \mathcal{E}_i, \eta \in \mathcal{V}_i\}$, which includes all μ's adjacent supernodes in $\mathcal{G}_i$. While merging supernodes μ_i and ν_i into ρ, the graph summary $\mathcal{S}_i$, including $\mathcal{G}_i$ and C_i, is updated accordingly. The following theorem guarantees that updates to $\mathcal{G}_i$ and C_i are only confined to the adjacent supernodes of μ_i or ν_i:

Theorem 3.1. *While merging supernodes μ_i and ν_i of $\mathcal{G}_i$ into a new supernode ρ, only the adjacent supernodes of μ_i or ν_i in $N_{\mathcal{G}_i}[\mu_i] \cup N_{\mathcal{G}_i}[\nu_i]$ are potentially updated in the graph summary.* □

Algorithm 2 Supernode_Merging (μ_i, ν_i)

Input: The supergraph $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$, the edge correction set C_i, and a pair of pre-selected supernodes $\mu_i, \nu_i \in \mathcal{V}_i$
Output: The supergraph $\mathcal{G}_{i+1} = (\mathcal{V}_{i+1}, \mathcal{E}_{i+1})$, the edge correction set C_{i+1}

1: $\rho \leftarrow \{u|u \in \mu_i, u \in V\} \cup \{v|v \in \nu_i, v \in V\}$ ▷ *A new supernode $\rho = \mu_i \cup \nu_i$*
2: **for** $\eta \in N_{\mathcal{G}_i}[\mu_i] \cup N_{\mathcal{G}_i}[\nu_i]$ **do**
3: $\quad E_{\rho,\eta} \leftarrow \{(u,v)|u \in \rho, v \in \eta\} \cap E$
4: $\quad$ **if** $|E_{\rho,\eta}| > (1 + |\rho| \cdot |\eta|)/2$ **then**
5: $\quad\quad \mathcal{E}_i \leftarrow \mathcal{E}_i \cup \{(\rho,\eta)\}$ ▷ *Create a superedge (ρ,η)*
6: $\quad\quad C_i \leftarrow C_i \setminus E_{\rho,\eta}$ ▷ *Remove existing $+(u,v)$ from C_i*
7: $\quad\quad C_i \leftarrow C_i \cup (\{(u,v)|u \in \rho, v \in \eta\} \setminus E_{\rho,\eta})$ ▷ *Add $-(u,v)$ to C_i*
8: $\quad$ **else**
9: $\quad\quad C_i \leftarrow C_i \setminus (\{(u,v)|u \in \rho, v \in \eta\} \setminus E_{\rho,\eta})$ ▷ *Remove $-(u,v)$ from C_i*
10: $\quad\quad C_i \leftarrow C_i \cup E_{\rho,\eta}$ ▷ *Add $+(u,v)$ to C_i*
11: $\mathcal{E}_{i+1} \leftarrow \mathcal{E}_i \setminus (\{(\mu_i,\eta)|\eta \in N_{\mathcal{G}_i}[\mu_i]\} \cup \{(\nu_i,\eta)|\eta \in N_{\mathcal{G}_i}[\nu_i]\})$
12: $\mathcal{V}_{i+1} \leftarrow (\mathcal{V}_i \setminus \{\mu,\nu\}) \cup \{\rho\}$
13: $C_{i+1} \leftarrow C_i$
14: **return** $\mathcal{G}_{i+1}(\mathcal{V}_{i+1}, \mathcal{E}_{i+1}), C_{i+1}$

Proof. Consider, otherwise, a supernode τ such that $\tau \notin N_{\mathcal{G}_i}[\mu_i]$ and $\tau \notin N_{\mathcal{G}_i}[\nu_i]$; that is, there exist no superedges (μ_i, τ) and (ν_i, τ) in $\mathcal{G}_i$. Therefore,

$$|E_{\mu_i,\tau}| \le (1 + |\mu_i| \cdot |\tau|)/2, \qquad |E_{\nu_i,\tau}| \le (1 + |\nu_i| \cdot |\tau|)/2.$$

Adding up both sides of the above inequalities, we have

$$|E_{\rho,\tau}| = |E_{\mu_i,\tau}| + |E_{\nu_i,\tau}| \le \frac{1 + (|\mu_i| + |\nu_i|) \cdot |\tau|}{2} = \frac{1 + |\rho| \cdot |\tau|}{2};$$

That is, there exists no superedge between τ and the new supernode ρ in $\mathcal{G}_{i+1}$. Therefore, the superedges incident to τ will not change during the supernode merging for μ_i and ν_i. Likewise, all the edges in the edge correction set C_i that are incident to the vertices within τ will not change as well. □

According to Theorem 3.1, when updating $\mathcal{G}_i$ and C_i due to a supernode merging between μ_i and ν_i, it suffices to examine the supernodes only in μ_i's or ν_i's neighboring set, as opposed to the entire supergraph $\mathcal{G}_i$ (Lines 2-10). After merging μ_i and ν_i into ρ, we remove from $\mathcal{G}_i$ the incident superedges of μ_i (resp. ν_i) (Line 11), followed by μ_i and ν_i, which are substituted by the new supernode ρ (Line 12). As a result, the supergraph $\mathcal{G}_i$ is summarized to $\mathcal{G}_{i+1}$, and the edge correction set C_i is updated to C_{i+1} (Line 13). This iterative supernode merging process culminates in the final graph summary $\mathcal{S}_n$ comprising the supergraph $\mathcal{G}_n$ and the edge correction set C_n.

To precisely quantify the summarization effectiveness incurred by a supernode merging to the graph summary, we introduce in Poligras a new notion of *summarization reward*:

Definition 4. **[SUMMARIZATION REWARD]** *Consider a supernode pair (μ_i, ν_i), where $\mu_i, \nu_i \in \mathcal{V}_i, 0 \le i \le n-1$. The summarization reward, $r(\mu_i, \nu_i)$, of merging μ_i and ν_i that transforms the graph summary $\mathcal{S}_i(\mathcal{G}_i, C_i)$ to $\mathcal{S}_{i+1}(\mathcal{G}_{i+1}, C_{i+1})$ is defined as*

$$r(\mu_i, \nu_i) = \max\{0, (|\mathcal{E}_i| + |C_i|) - (|\mathcal{E}_{i+1}| + |C_{i+1}|)\}.$$ □

As $r(\mu_i, \nu_i) \ge 0$, the summarization reward $r(\mu_i, \nu_i)$ specifies an *absolute* size reduction to graph summaries incurred by a supernode merging of μ_i and ν_i. Therefore, at each step of summarization, we aim to identify a pair of supernodes, μ_i^* and ν_i^*, from $\mathcal{G}_i$ (more specifically, from each partitioned group of $\mathcal{V}_i$), such that

$$(\mu_i^*, \nu_i^*) = \arg\max\{r(\mu_i, \nu_i)\}, \qquad \mu_i, \nu_i \in \mathcal{V}_i. \tag{3}$$

When there exist no supernode pairs at step $n(\geq 0)$ satisfying $r(\mu_n^*, \nu_n^*) > 0$, it means $\mathcal{G}_n$ can no longer be summarized for further size reduction. We define the *cumulative summarization reward* for the graph summary $\mathcal{S}_n$ as

$$r(\mathcal{S}_n) = \sum_{i=0}^{n-1} r(\mu_i^*, \nu_i^*) \qquad \mu_i^*, \nu_i^* \in \mathcal{V}_i \tag{4}$$

As each $r(\mu_i^*, \nu_i^*) > 0$, it is immediate that $r(\mathcal{S}_{i+1}) > r(\mathcal{S}_i)$; that is, the cumulative summarization reward $r(\mathcal{S}_i)$ is monotonically increasing *w.r.t. i*. When $i = n$ and $\nexists \mu_n^*, \nu_n^* \in \mathcal{V}_n$ such that $r(\mu_n^*, \nu_n^*) > 0$, we terminate the supernode merging process, which ends up with the ultimate graph summary $\mathcal{S}_n = (\mathcal{G}_n, C_n)$.

A direct benefit of quantifying the supernode merging effectiveness based on the notion of summarization reward is that it is intrinsically related to the graph summary cost (Equation 1), as shown in the following theorem.

Theorem 3.2. *Given a graph $G = (V, E)$, a sequence of n supernode-merging operations, and the ultimate graph summary $\mathcal{S}_n = (\mathcal{G}_n, C_n)$, the cumulative summarization reward of $\mathcal{S}_n$, $r(\mathcal{S}_n)$, is equivalent to $|E| - (|\mathcal{E}_n| + |C_n|)$, the absolute reduction of graph summary cost of $\mathcal{S}_n$ w.r.t. G.* □

Proof. Based on the definition of cumulative summarization reward, we note that

$$\begin{aligned} r(\mathcal{S}_n) &= \sum_{i=0}^{n-1} r(\mu_i^*, \nu_i^*) = \sum_{i=0}^{n-1} \max\{0, (|\mathcal{E}_i| + |C_i|) - (|\mathcal{E}_{i+1}| + |C_{i+1}|)\} \\ &= \sum_{i=0}^{n-1} \{(|\mathcal{E}_i| + |C_i|) - (|\mathcal{E}_{i+1}| + |C_{i+1}|)\} \\ &= (|\mathcal{E}_0| + |C_0|) - (|\mathcal{E}_n| + |C_n|) = |E| - (|\mathcal{E}_n| + |C_n|) \end{aligned}$$

Specifically, the second line holds because $r(\mu_i^*, \nu_i^*) > 0$ for all $0 \leq i \leq n-1$; The third line holds because the telescopic sums cancel consecutive terms, leaving only the initial and final ones. In particular, $\mathcal{E}_0$ is equivalent to the edge set E of the original graph G, and C_0 is initialized to $\emptyset$, so $|C_0| = 0$. □

According to Theorem 3.2, the objective of identifying a min-cost graph summary $\mathcal{S}^*$ (Definition 3) translates to finding an optimal sequence of n supernode pairs (μ_i^*, ν_i^*) determined by Equation 3. As a consequence, the iterative merging of μ_i^* and ν_i^* $(0 \leq i \leq n-1)$ will lead to the optimal graph summary $\mathcal{S}^*(= \mathcal{S}_n)$ with the maximum cumulative summarization reward, $r(\mathcal{S}_n)$. It is worth noting that existing graph summarization solutions, such as SWeG [37] and LDME [45], adopt approximate metrics for supernode pair selection, which result in non-optimal graph summaries. Poligras instead considers the new notion of summarization reward, the maximization of which directly leads to the min-cost graph summary.

3.2 Supernode Selection Policy

To this end, our prime goal is to select the optimal pair of supernodes, μ_i^* and ν_i^*, from $\mathcal{G}_i$, where $i = 0, 1, 2, \ldots$, based on Equation 3. In practice, given the supergraph $\mathcal{G}_i$, its supernode set $\mathcal{V}_i$ is first partitioned to a set $\mathcal{P}$ of supernode groups, and supernode pairs are chosen from within each group $P \in \mathcal{P}$. A straightforward solution is to explore every supernode pair within P, the number of which can be $\binom{|P|}{2} = O(|P|^2)$, and for each such pair (μ, ν), we tentatively merge them in order to compute the summarization reward, $r(\mu, \nu)$, based on Algorithm 2. The pair with the maximum summarization reward will be selected for merging. This brute-force approach, however, is computationally infeasible especially in real-world large graphs (See details in Section 4).

To address this challenge, we propose in Poligras a learning-enhanced supernode selection policy. Intuitively, this policy is a probabilistic function aiming to model the pairwise likelihood of supernodes that could potentially be selected as the optimal pair for merging: Given any supernode pair (μ, ν) in a partitioned group P, the higher the summarization reward $r(\mu, \nu)$, the greater the probability assigned for this pair (μ, ν) by the supernode selection policy. To this end, the supernode selection policy is essentially a probabilistic approximation for summarization rewards. In addition, the learning and optimization of such a policy must be efficient, thus making the process of supernode pair selection computationally feasible in real-world graphs.

The learning and computation of the supernode selection policy consists of two components: (1) supernode embedding and (2) policy function learning and optimization, as elaborated below.

3.2.1 **Supernode Embedding**. Based on Definition 4, we note that the summarization reward $r(\mu, \nu)$ quantifies an absolute size reduction to the graph summaries incurred by merging supernodes μ and ν. According to Theorem 3.1 and Algorithm 2, however, the merge of μ and ν causes potential updates for a sub-portion of the graph summary, which is only related to the adjacent supernodes of μ or ν. In order to capture the neighborhood information of supernodes during the summarization reward computation, we propose to embed each supernode into a low-dimensional space with reduced complexity, high efficiency, and scalability.

At the beginning, each vertex u of the input graph G is treated as a special supernode $\{u\}$ in $\mathcal{G}_0$, and its supernode-embedding, $\mathbf{H}_{\{u\}}$, is a $|V|$-dimensional vector based on the *one-hot* encoding of u's adjacent neighbors in G: The i-th entry of $\mathbf{H}_{\{u\}}$ is 1 if $(u, v_i) \in E$, or 0 otherwise, where $v_i \in V (1 \leq i \leq |V|)$. To further reduce its dimensionality, we divide the one-hot encodings into d groups of sub-encodings, each of which is with a length of $\lceil |V|/d \rceil$, except for the last group whose length is $(|V| - (d-1) * (\lceil |V|/d \rceil))$ without padding. Here $d (\ll |V|)$ is a user-specified parameter for the final embedding dimension. We then aggregate each of the d groups by summing up its sub-encoding. This way, the dimension of the supernode embedding $\mathbf{H}$ is reduced from $|V|$ to d.

In the subsequent iterative process of graph summarization, when two supernodes μ and ν are merged to a new supernode ρ, the embedding $\mathbf{H}$ is updated accordingly. As the neighboring set of ρ is solely determined by the union of neighboring sets of μ and ν, its embedding can be computed as

$$\mathbf{H}_\rho = \mathbf{H}_\mu \oplus \mathbf{H}_\nu, \tag{5}$$

where $\oplus$ is the element-wise addition between d-dimensional vectors, and $\mathbf{H}_\rho \in \mathbb{R}^d$. This embedding scheme encodes the salient neighborhood information of supernodes, which is essential for supernode merging in Algorithm 2. Furthermore, it also provides the low-dimensional input for the policy function learning and optimization as discussed below.

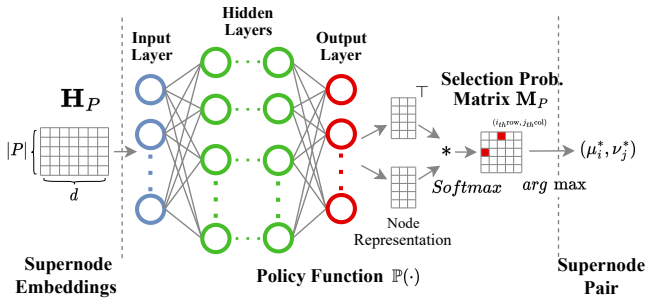

Figure 3: Supernode Pair Selection by the Policy $\mathbb{P}(\cdot)$.

3.2.2 **Policy Function Learning and Optimization**. The supernode selection policy is a learnable probabilistic function $\mathbb{P}(\cdot)$ that models the summarization reward for supernode pairs. Specifically, given as input the embeddings $\mathbf{H}_P$ of supernodes in a partitioned group P, the policy function $\mathbb{P}(\cdot)$, which is learned and optimized by a multi-layer neural network, estimates the pairwise supernode selection probability $\mathbb{P}(\mu, \nu)$, for any $\mu, \nu \in P$. The overall process for policy-based supernode pair selection is illustrated in Figure 3. In particular, given a partitioned group $P \in \mathcal{P}$ (from the group partitioning stage in Section 2.2), we embed all the supernodes of P into a d-dimensional space. The resultant supernode embedding matrix is denoted by $\mathbf{H}_P \in \mathbb{R}^{|P| \times d}$ with each row $\mathbf{H}_P[i]$ representing a d-dimensional vector for the i-th supernode μ_i of P. Given as input the supernode embedding matrix $\mathbf{H}_P$, the supernode selection policy $\mathbb{P}(\cdot)$ is formulated as

$$\mathbb{P}(\mathbf{H}_P) := \text{softmax}(\text{NN}(\mathbf{H}_P) \cdot \text{NN}(\mathbf{H}_P)^\top), \tag{6}$$

where $\text{NN}(\cdot)$ is a multi-layer neural network whose weights are to be learned and optimized. In essence, the policy function $\mathbb{P}(\cdot)$ characterizes the pairwise similarity of supernodes in terms of their neighborhood proximity. It generates as output a symmetric probability matrix $\mathbf{M}_P \in [0,1]^{|P| \times |P|}$ with $\mathbf{M}_P[i,j]$ representing the selection probability of the supernode pair (μ_i, ν_j), where $\mu_i, \nu_j \in P$. Consequently, the supernode pair with the largest selection probability is chosen for supernode merging:

$$(\mu_i^*, \nu_j^*) = \underset{1 \le i < j \le |P|}{\text{argmax}} \ \mathbf{M}_P[i,j]. \tag{7}$$

To enforce the intrinsic property that the supernode selection probability is an stochastic estimation of summarization rewards, we learn and optimize the policy $\mathbb{P}(\cdot)$ in an unsupervised, online-learning fashion. In particular, for each iteration of graph summarization, the supernode set $\mathcal{V}$ is partitioned to a set $\mathcal{P}$ of disjoint groups, and we identify from each group $P_i \in \mathcal{P}$ a supernode pair, $(\mu_{P_i}^*, \nu_{P_i}^*)$, with the largest supernode-pair selection probability, $\mathbf{M}_{P_i}[\mu_{P_i}^*, \nu_{P_i}^*]$. We further compute its exact summarization reward, $r(\mu_{P_i}^*, \nu_{P_i}^*)$, by merging $\mu_{P_i}^*$ and $\nu_{P_i}^*$ based on Algorithm 2. As a result, we have $|\mathcal{P}|$ such supernode pairs, together with their summarization rewards, which constitute the training instances toward learning the policy, $\mathbb{P}(\cdot)$, within an iteration. We define the loss function for training $\mathbb{P}(\cdot)$ as

$$\mathcal{L} = \frac{1}{|\mathcal{P}|} \sum_{i=1}^{|\mathcal{P}|} \{-\log(M_{P_i}[\mu_{P_i}^*, \nu_{P_i}^*]) \cdot \tilde{r}(\mu_{P_i}^*, \nu_{P_i}^*)\}, \tag{8}$$

Algorithm 3 Poligras Training (In Lines 4-6 of Algorithm 1)

```
1:  best_sum ← 0
2:  while True do
3:      sp_list ← [], sr_list ← []      ▷ Maintain M_{P_i}[μ*_{P_i}, ν*_{P_i}] and r(μ*_{P_i}, ν*_{P_i})
4:      for P_i ∈ 𝒫 do
5:          (μ*_{P_i}, ν*_{P_i}) ← argmax M_{P_i} from ℙ(H_{P_i})      ▷ Supernode selection
6:          sp_list.push(M_{P_i}[μ*_{P_i}, ν*_{P_i}])
7:          sr_list.push(r(μ*_{P_i}, ν*_{P_i}))
8:      ℒ = 1/|𝒫| Σ_{i=1}^{|𝒫|} {−log(M_{P_i}[μ*_{P_i}, ν*_{P_i}]) · r̃(μ*_{P_i}, ν*_{P_i})}      ▷ Loss function
9:      w ← w − α · Adam(∂ℒ/∂w)      ▷ Back-propagation via the Adam optimizer
10:     if sum(sr_list) ≤ best_sum then
11:         Break
12:     best_sum ← sum(sr_list)
```

where $\tilde{r}(\mu_{P_i}^*, \nu_{P_i}^*) = (r(\mu_{P_i}^*, \nu_{P_i}^*) - \mu_r)/\sigma_r$ is the *normalized* summarization reward for the supernode pair $(\mu_{P_i}^*, \nu_{P_i}^*)$, with μ_r and σ_r being the sample mean and the sample standard deviation, respectively, of the summarization rewards for the aforementioned collection of $|\mathcal{P}|$ supernode pairs, $(\mu_{P_i}^*, \nu_{P_i}^*)$, where $1 \le i \le |\mathcal{P}|$. In the training phase, our goal is to minimize the loss $\mathcal{L}$. As a result, the policy $\mathbb{P}(\cdot)$ is optimized in the following manner:

(1) For a supernode pair (μ_{P_i}, ν_{P_i}) with a high summarization reward, its supernode selection probability, $\mathbf{M}_{P_i}[\mu_{P_i}, \nu_{P_i}]$, is inclined to become large. In other words, in order to decrease the loss $\mathcal{L}$, the policy $\mathbb{P}(\cdot)$ tends to enhance the supernode selection probabilities for the supernode pairs with high summarization rewards;
(2) On the other hand, for supernode pairs with low summarization rewards (*e.g.*, $\tilde{r}(\mu_{P_i}, \nu_{P_i}) < 0$), the policy $\mathbb{P}(\cdot)$ tends to output small supernode selection probabilities.

In Poligras, the training and optimization of the policy function $\mathbb{P}(\cdot)$ is accomplished by back-propagation. Different from the conventional machine learning process where the model training and usage for inference are separated to different stages, the training and optimization of $\mathbb{P}(\cdot)$ follows an online learning paradigm that is integrated to the graph summarization process. In particular, within each iteration of summarization (Lines 4-6 of Algorithm 1), the back-propagation algorithm is executed to minimize the loss function $\mathcal{L}$ (Equation 8) by the Adam optimizer [17].

The training and optimization of the policy $\mathbb{P}(\cdot)$ is detailed in Algorithm 3. Specifically, in each execution of back-propagation, a supernode pair, $(\mu_{P_i}^*, \nu_{P_i}^*)$, is selected from each partitioned group $P_i \in \mathcal{P}$. Its supernode selection probability, $\mathbf{M}_{P_i}[\mu_{P_i}^*, \nu_{P_i}^*]$, and the summarization reward, $r(\mu_{P_i}^*, \nu_{P_i}^*)$, are maintained in two data structures, *sp_list*, and *sr_list*, respectively (Lines 4-7). These $|\mathcal{P}|$ supernode pairs constitute a mini-batch for training $\mathbb{P}(\cdot)$ and updating the weights, $\mathbf{w}$, of the neural network (Lines 8-9). After each run of back-propagation, a new mini-batch of $|\mathcal{P}|$ supernode pairs are selected for training and optimization. If the total summarization reward of the current batch is greater than that of the previous one, back-propagation will continue; Otherwise, we terminate the optimization in the current iteration of graph summarization, and the mini-batch with the greatest total summarization reward will be chosen for supernode merging (Lines 10-12).

3.3 Computational Complexity of Poligras

In Algorithm 1, we note that Poligras is an iterative method with the number of iterations, T, as an input. It can terminate early if summarization rewards at iteration $T' (< T)$ are no longer positive. In each iteration, Poligras comprises the following key steps: (1) Supernode partitioning: We use minhashing to partition the supernode set $\mathcal{V}$ into $|\mathcal{P}|$ disjoint groups based the node connectivity of G with the time complexity of $O(|V|)$ and the space complexity of $O(|E|)$; (2) Supernode embedding: All the supernodes are embedded into a low-dimensional space $\mathbf{H}$ represented as a $|V| \times d$ matrix, where d is the dimensionality for supernode embeddings. The space cost is $O(d \cdot |V|)$, and the time complexity for computing and updating $\mathbf{H}$ is $O(d \cdot |V|)$; (3) Policy learning and optimization, which further contains three computational components:

(1) Mini-batch supernode selection: This step mainly involves the computation for the neural network. We maintain an intermediate data structure for the probability matrix $\mathbf{M}_P$, where $P \in \mathcal{P}$ is a partitioned group with an average size of $|P| = \lceil |V|/|\mathcal{P}| \rceil$. Therefore, the space cost is $O(|P|^2)$. The neural network computation is $O(\eta \cdot (d \cdot |P| + |P|^2))$, where η is a small, constant factor related to tiny-size matrix multiplication once the neural network structure is determined and fixed; For all the partitions, the time complexity is $O(\eta(|V| + |V|^2/|\mathcal{P}|))$;
(2) Summarization reward computation: According to Algorithm 2, an exact computation of $r(\mu, \nu)$ needs a tentative supernode merging, whose complexity is $O(d_m)$, where d_m denotes the maximum vertex-degree of G. In each mini-batch, we need to compute $r(\mu_P^*, \nu_P^*)$ for every partitioned group $P \in \mathcal{P}$, so the time complexity is $O(d_m \cdot |\mathcal{P}|)$;
(3) Back-propagation optimization: In each round of optimization, the time complexity stems from updates to the parameters of the neural network, which is related to parameters α (the learning rate) and d, so the complexity for this step is $O(\alpha \cdot d)$.

Therefore, the total time complexity for policy learning and optimization is $O(\eta(|V| + |V|^2/|\mathcal{P}|) + d_m \cdot |\mathcal{P}| + \alpha \cdot d)$. In our setting, the size of each partitioned group, $|P| = \lceil |V|/|\mathcal{P}| \rceil$, is typically bounded by a small constant (*e.g.*, 200 or 400), so it is safe to assume $|\mathcal{P}| = O(|V|)$. The above complexity can be further simplified to $O(\beta \cdot |V| + d_m \cdot |V|)$, where β is a constant factor. After the supernode set $\mathcal{V}$ is determined, we encode plain edges of G into the superedge set $\mathcal{E}$ and the edge correction set C with the time and space complexity of $O(|E|)$. As a result, the total time complexity of Poligras is $O((\beta + d_m) \cdot |T| \cdot |V| + |E|)$, indicating it is a linear-time method for graph summarization. In addition, its space complexity is $O(|E| + d \cdot |V|)$, which is also space-efficient.

4 EXPERIMENTS

We report our experimental studies and key findings for graph summarization in real-world graphs. In particular, we compare our policy-based graph summarization method, Poligras, with the state-of-the-art solutions, SWeG [37] and LDME [45], by evaluating both the summarization effectiveness and efficiency. We also examine the learning-enhanced mechanisms of Poligras for supernode selection and merging and its potential benefits for graph summarization. All our experiments are carried out on a Linux server running Ubuntu 20.04 with two Intel 2.3GHz ten-core CPUs and 256GB memory.

Table 2: Statistics of Graph Datasets

Datasets	|V|	|E|	Avg. Degree
astro-ph (**AS**)	18,772	198,110	21.11
cnr-2000 (**CN**)	325,557	2,738,969	16.83
skitter (**SK**)	1,696,415	11,095,298	13.08
in-2004 (**IN**)	1,382,908	13,591,473	19.65
eu-2005 (**EU**)	862,664	16,138,468	37.42
patent (**PA**)	3,774,768	16,518,948	8.75
tweibo (**TW**)	1,397,020	41,145,547	58.90
hollywood-2011 (**HW**)	2,180,759	114,492,816	105.00
indochina-2004 (**IC**)	7,414,866	150,984,819	40.72
uk-2002 (**UK**)	18,520,486	261,787,258	28.27

4.1 Datasets

We consider ten real-world graph datasets [3–6, 23, 24] in our experimental studies, and their key statistics, including the number of nodes, $|V|$, the number of edges, $|E|$, and the average node-degrees of graphs, are reported in Table 2. These datasets have been extensively used to evaluate existing graph summarization techniques, such as SWeG and LDME. Concretely, **astro-ph** (**AS**) is a collaboration network between the authors who submitted papers to the Astro-Physics domain; **cnr-2000** (**CN**), **in-2004** (**IN**), and **eu-2005** (**EU**) are hyperlink networks in different Internet domains; **skitter** (**SK**) is an Internet topology graph extracted by `traceroute` run daily in 2005; **patent** (**PA**) is a citation network for U.S. utility patents published between 1975 and 1999; **tweibo** (**TW**) is a social network from the Tencent Weibo platform; **hollywood-2011** (**HW**) is a Hollywood movie-actor social network, in which nodes represent actors and edges indicate the co-occurrence relationship for two actors in the same movie; **indochina-2004** (**IC**) is a sub-Internet crawled in the country domains of Indochina; **uk-2002** (**UK**) is an Internet graph crawled in the `.uk` domain in 2002.

4.2 Experimental Settings

4.2.1 **Baseline Methods**. Besides Poligras, we consider in our experimental studies two state-of-the-art graph summarization solutions: (1) SWeG [37] is a scalable graph summarization method that employs *shingling* and *minhashing* techniques for supernode partitioning, and the *SuperJaccard* metric for supernode selection and merging; (2) LDME [45] adopts the weighted locality sensitive hashing (LSH) technique for supernode partitioning, and achieves significant speedup in supernode merging based on hash indexing. Both methods, however, are approximate graph summarization solutions with no learning-enabled mechanisms proposed. We also develop another baseline method, Baseline, which explores every supernode pair, (μ, ν), in a partitioned group to compute the *exact* summarization reward, $r(\mu, \nu)$, and selects the one, (μ^*, ν^*), with the highest summarization reward for supernode merging. Baseline guarantees that the optimal supernode pairs can always be identified for merging. However, Baseline is extremely time-consuming: It fails to summarize the smallest graph, **CN**, within 120 hours. We thus ignore its results in the following experimental studies.

4.2.2 **Evaluation Metrics**. We consider both the summarization effectiveness and the runtime efficiency to evaluate different graph summarization methods. For the effectiveness metric, we consider

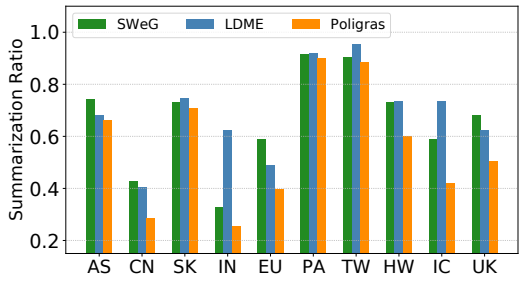

Figure 4: Graph Summarization Ratios on All Datasets.

the *summarization ratio* of the graph summary $\mathcal{S}_n$ *w.r.t.* the input graph G as

$$\frac{cost(\mathcal{S}_n)}{|E|} = \frac{|\mathcal{E}_n| + |C_n|}{|E|}, \tag{9}$$

where $|E|$ represents the size of the original graph G. The graph summarization ratio quantifies the *percentage* of the graph summary size *w.r.t.* the original graph size; The lower the graph summarization ratio, the lower the graph summary cost (Equation 1), and the better the graph summarization effectiveness. For the efficiency metric, we report the overall runtime cost (in seconds) for different graph summarization algorithms.

4.2.3 **Algorithm and Parameter Settings for Poligras**. We choose minhashing for group partitioning in Poligras, like SWeG. Specifically, in the **TW** and **HW** datasets, the size of partitioned groups (*i.e.*, the average number of supernodes in each group) is 400, and in all the other datasets, the size is 200; This setting is consistent with the group-size settings as reported in SWeG and LDME. For supernode embeddings in Poligras, we set the embedding dimension d between 200 and 1,700 in different graph datasets. To train the supernode selection policy $\mathbb{P}(\cdot)$, we consider a two-layer Perceptron with hidden sizes of 64 and 32, respectively, as the underlying neural network (We carry out experimental studies by varying the number of layers and the size of each layer in the neural network. Marginal changes in experimental results are witnessed, and thus omitted for reporting in the paper). In the online learning process, Poligras is optimized by back-propagation with the Adam optimizer [17], and the default learning rate α is set to 0.001.

4.3 Experimental Results

4.3.1 **Graph Summarization Effectiveness**. In the first experiment, we evaluate three graph summarization methods, Poligras, SWeG, and LDME, upon ten real-world graph datasets, and report the graph summarization ratio results in Figure 4. We set in this experiment the number of iterations, T, for graph summarization to be an excessively large value, such that each summarization method terminates only when the iterative summarization process can no longer contribute further size reductions to graph summaries; that is, the results reported here are the best possible graph summarization ratios different methods can achieve in the graphs. First of all, Poligras can losslessly summarize real-world graphs of varying sizes and structures into succinct summaries that are significantly smaller than original graphs. For instance, the graph summaries derived from the **CN** (small), **IN** (medium), and **IC** (large) datasets are merely 28%, 24%, and 43% the sizes of the original graphs, respectively, thereby demonstrating its excellent summarization capabilities in real-world graphs.

Next, we note that Poligras outperforms SWeG and LDME by consistently achieving the smallest graph summarization ratios in every graph dataset; that is, Poligras can obtain the most succinct graph summaries for all these graphs. In particular, the gains of graph summarization effectiveness by Poligras are notable: when compared with SWeG, Poligras offers 1.72×, 1.55×, and 1.64× improvements in the **CN** (small), **EU** (medium), and **IC** (large) datasets, respectively; when compared with LDME, Poligras achieves 1.64×, 2.74×, and 1.71× improvements in the **CN** (small), **IN** (medium), and **IC** (large) datasets, respectively. Even for the largest graph, **UK**, Poligras gains 1.42× and 1.33× improvements in graph summarization ratios compared with SWeG and LDME, respectively, and the graph summary obtained by Poligras is less than half the size of the original graph without information loss. Such performance gains are due in particular to the learning-enhanced supernode selection policy proposed in Poligras, which can be further validated in the experiments in Section 4.3.5.

In the second experiment, we consider another application scenario for graph summarization: Instead of waiting for the ultimate graph summaries where no supernodes can be further merged for improved summarization effectiveness, we early terminate the graph summarization process by fine tuning the number of iterations, T, as long as (1) the current graph summary has reached a desirable size, or (2) the difference of graph summarization ratios between adjacent iterations becomes marginal. In this scenario, users would accept sufficiently compact, not necessarily the smallest, graph summaries in order to trade for summarization efficiency.

By regulating the number of iterations, T, we report the graph summarization ratios for all the graph datasets in Figure 5. We note that, in each dataset and for different values of T, Poligras consistently outperforms SWeG and LDME, and the performance gains are significant. For example, when $T = 100$, Poligras achieves 1.73× and 2.45× improvements for graph summarization ratios in the graph **IN**, when compared with SWeG and LDME, respectively. In the largest graph, **UK**, if users want a graph summary with 60% the size of the original graph, it only takes Poligras $T = 60$ iterations of computation. However, neither SWeG nor LDME can attain this goal after $T = 120$ iterations of summarization, which is time-consuming. The reason that Poligras outperforms SWeG and LDME is twofold: First, Poligras uses the notion of summarization reward to quantify the effectiveness of supernode selection and merging, which helps find the min-cost graph summaries; Second, and more importantly, the learning-enhanced supernode selection policy $\mathbb{P}(\cdot)$ in Poligras is more effective for optimal supernode pair selection than the approximation or randomized algorithms used in SWeG and LDME.

To further validate the advantages of Poligras in supernode selection and merging, we report the *average* summarization rewards for the supernode pairs identified in the first 50 iterations of different graph summarization methods, and the results are illustrated in Figure 6. According to Definition 4, the summarization reward, $r(\mu, \nu)$, quantifies the absolute size reduction to graph summaries incurred by merging the pre-selected supernodes μ and ν, so the average summarization reward signifies the *quality* of supernode pairs selected by different summarization methods. In all graph

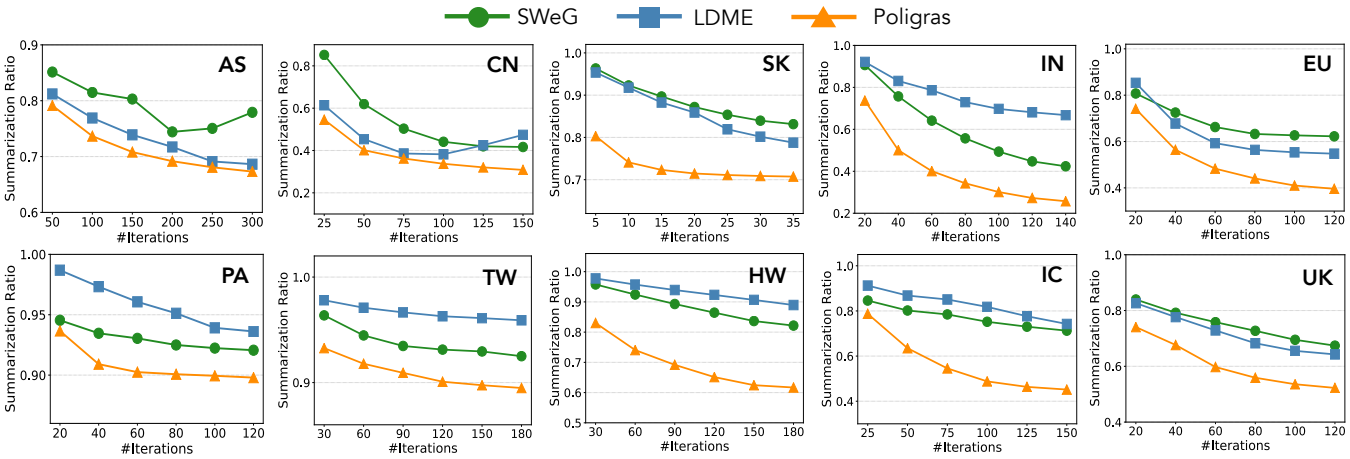

Figure 5: Graph Summarization Ratio *w.r.t.* Number of Iterations, $|T|$, for Different Graph Summarization Methods.

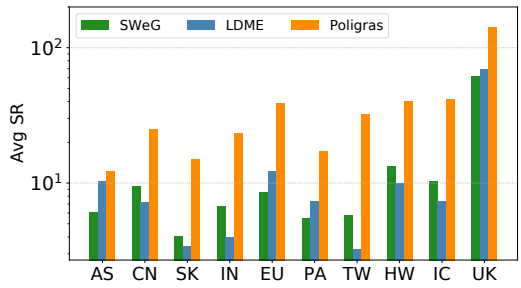

Figure 6: Average Summarization Rewards on All Datasets.

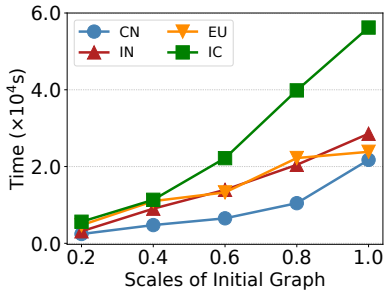

Figure 7: Scalability Tests for Poligras

datasets, the average summarization rewards of Poligras are significantly higher than those of SWeG and LDME. In the **TW** dataset, for instance, Poligras achieves 5.6× and 9.8× improvements when compared with SWeG and LDME, respectively, thanks to the supernode selection policy of Poligras, which helps identify and merge supernode pairs with high summarization rewards, and therefore result in low-cost graph summaries for real-world graphs.

4.3.2 **Graph Summarization Efficiency**. In this experiment, we examine the efficiency of different graph summarization methods in real-world graphs. In the first setting, we run different algorithms until the ultimate graph summaries stabilize (no supernode merging arises for further summarization), and the overall runtime costs (in seconds) are reported in Figure 8. We note that Poligras is consistently faster than LDME, and the improvements on summarization efficiency ranges from 1.2× to 2.2× across different datasets. When compared with SWeG, Poligras has similar runtime costs in small and medium-size graphs, such as **CN**, **SK**, **IN**, and **EU**. However, when the graphs get larger, such as in **TW**, **HW**, **IC**, and **UK**, Poligras runs faster than SWeG: the improvements on summarization efficiency range from 1.5× to 1.8×.

For Poligras, we also report in Figure 8 the subdivided runtime (in log-scale) for the three main computational components: (1) supernode partitioning, (2) supernode selection based on policy learning and optimization, and (3) edge encoding. Note that, on average, 86.2% of the running time for Poligras is consumed for supernode selection and merging; The supernode partitioning takes 9.3% of the overall time; For edge encoding, it takes only 4.5% of the overall time for graph summarization.

In another setting, we report the runtime costs of different methods by limiting the number of iterative executions of graph summarization up to T, and the results are shown in Figure 9. Note that Poligras has similar runtime costs to SWeG; Both are faster than LDME in small and medium-size graphs. When graphs get larger, Poligras is significantly faster: in the largest **UK** dataset, for instance, the speedup ranges from 2.4× to 3.7× compared to SWeG, and 2.1× to 2.8× compared to LDME. In SWeG and LDME, for each step of supernode selection, it is imperative to compute the SuperJaccard coefficient for all the supernodes in a partitioned group, which is time-consuming. In contrast, supernode selection in Poligras is accomplished by the learnable policy $\mathbb{P}(\cdot)$, whose training and optimization in mini-batches is efficient.

4.3.3 **Scalability Analysis**. To examine if the proposed method, Poligras, can scale up in real-world graphs, we extract a series of connected subgraphs with varying sizes of $x\% \times |E|$, where $x\%$ denotes the percentage of edges retained in subgraphs, and evaluate the runtime cost for graph summarization. The results for four graph datasets are presented in Figure 7. In this experiment, we set five different size ratios ranging from 20% to 100%. With an increase of graph sizes, Poligras exhibits excellent scalability: The runtime costs of graph summarization in the graphs of varying

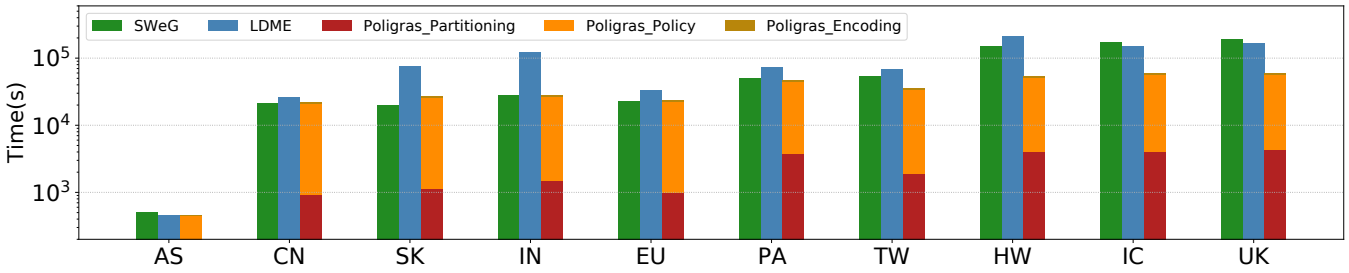

Figure 8: Graph Summarization Efficiency for Different Methods in Ten Graph Datasets.

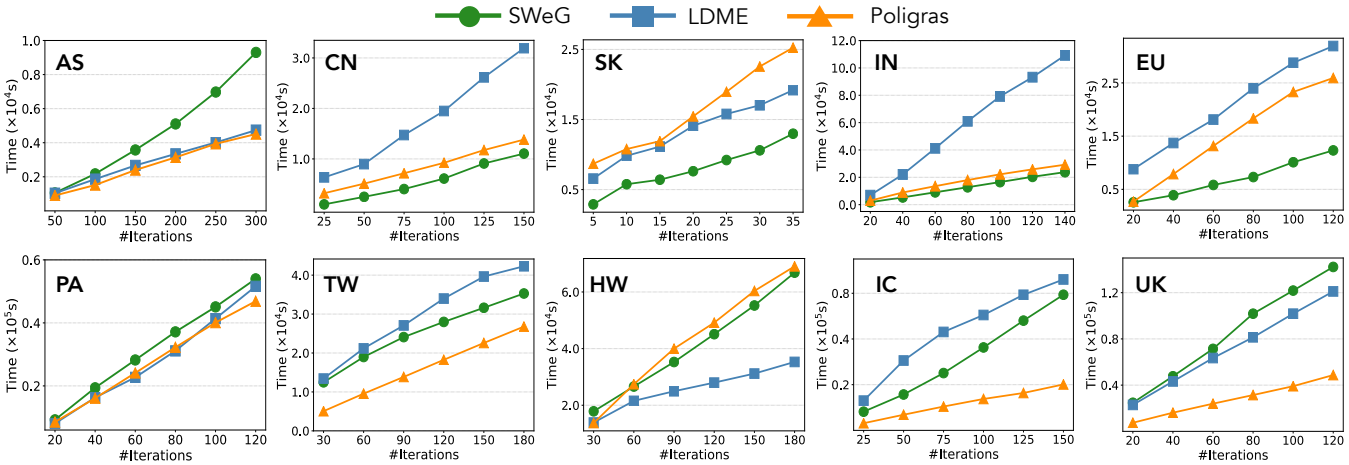

Figure 9: Graph Summarization Efficiency *w.r.t.* Number of Iterations, $|T|$, for Graph Summarization Algorithms.

sizes present nearly linear trends. These results are consistent with the time-complexity analysis in Section 3.3. Consequently, Poligras is a scalable graph summarization method that can be employed in real-world, large-scale graphs for efficient graph summarization.

4.3.4 **Parameter Sensitivity Analysis.** Poligras has several algorithmic parameters that can be regulated. In the following experiments, we examine how these parameters affect the performance of graph summarization. We first regulate the group size (the average number of supernodes within each partitioned group) for supernode partitioning, and report graph summarization ratios and runtime costs in Figure 10. While changing the group sizes from 50 to 800, the graph summarization ratios remain roughly stable with an exception of a steady decrease in the large graph **IC**. The reason is that in large partitioned groups of a large graph, chances are higher to select the supernode pairs with larger summarization rewards, thus resulting in lower-cost graph summaries. On the other hand, with an increase of the group sizes, the runtime costs of Poligras grow proportionally, as more time is needed for supernode selection and merging in larger partitioned groups.

In Poligras, supernodes of each partitioned group are first embedded into a d-dimensional space before fed into the neural network for policy learning. In this experiment, by regulating the dimensionality d in supernode embeddings, we report graph summarization ratios and runtime costs of Poligras in Figure 11. In general, the summarization ratios remain stable given different values of d in four graphs. Minor decreases are witnessed when we increase the embedding dimensions from $d/4$ to d, as more neighborhood information surrounding supernodes and more learnable weights in neural networks have to be encoded in Poligras, both of which help improve the effectiveness for supernode selection and merging. On the other hand, the runtime costs grow proportionally with an increase of d, as it takes more time for Poligras to train and optimize the policy $\mathbb{P}(\cdot)$ when the supernode embeddings have a larger dimensionality.

4.3.5 **Effectiveness of Policy Optimization**. In this experiment, we evaluate the effectiveness of the optimization mechanism in the supernode selection policy, $\mathbb{P}(\cdot)$. We first design an alternative method for the policy function computation, denoted as Non-optimized, by opting out the back-propagation optimization in Poligras (Line 9 in Algorithm 3). In other words, the approach Non-optimized does not optimize the objective loss function (Equation 8) during policy training. We then compare Non-optimized to Poligras with the optimization enabled (our default setting), which is denoted as Optimized, and the graph summarization ratios in different graph datasets are presented in Table 3. Note this experiment directly demonstrates the correctness of the proposed objective loss function, and also the effectiveness of the optimization mechanism in $\mathbb{P}(\cdot)$. From Table 3, we recognize that the optimization-enabled method, Optimized, achieves consistently lower summarization ratios than Non-optimized in different graph datasets, indicating

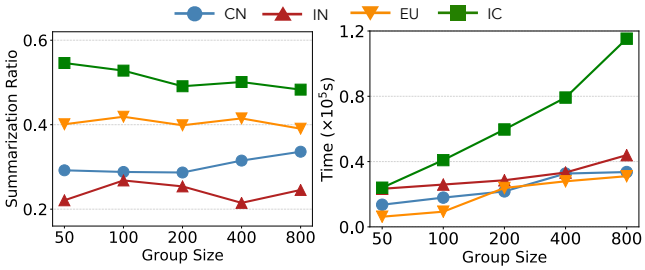

Figure 10: Poligras with Different Group Sizes.

the optimization mechanism of Poligras indeed helps improve the effectiveness for graph summarization.

Table 3: Policy Optimization in Poligras

Methods	AS	CN	SK	IN	EU	PA	TW	HW	IC	UK
Non-optimized	0.717	0.348	0.753	0.501	0.693	0.930	0.901	0.928	0.721	0.661
Optimized	**0.661**	**0.286**	**0.649**	**0.254**	**0.398**	**0.891**	**0.884**	**0.603**	**0.489**	**0.506**

5 RELATED WORKS

Graph Summarization. As an effective means for graph data reduction and simplification, graph summarization aims to identify concise representations for graphs [15, 28]. Existing graph summarization methods can be broadly classified into two categories: *lossy summarization* and *lossless summarization*. Lossy graph summarization techniques, such as summarization with utility loss [12, 20], summarization with construction error [33], graph sparsification [21], and graph structure summary [22], seek to optimize the compression for effective graph storage and representation, yet some graph-specific information may become inaccurate or even lost during summarization. In contrast, lossless graph summarization [31, 37, 45] strives to reduce the storage requirement for graphs, while the complete information of vertices and edges in the graphs can be well preserved and fully restored if needed. The graph summarization problem studied in this paper belongs to the category of lossless graph summarization.

The edge-correction based graph summarization [31, 37, 45] is one of the widely used lossless graph summarization techniques. GREDDY [31] proposes a novel framework that reduces an input graph into a summary graph and an edge correction set: The summary graph consists of supernodes and superedges, and the edge correction set aims to rectify incorrect connections in the summary graph. RANDOMIZED [31] follows the same summarization framework as GREDDY. However, during supernode merging, it first picks a supernode uniformly at random, and then select another supernode in order to maximize the number of edges thus saved in the summary. SAGS [16] and LDME [45] apply the locality sensitive hashing technique for supernode selection and merging. SWeG [37] uses SuperJaccard similarity as an approximation metric for supernode selection, and it is further implemented in the shared-memory and MapReduce environment for parallel graph summarization. Other related topics, such as summarizing dynamic graphs or graph streams [18, 26, 32, 36, 41], query- or utility-driven graph summarization [8, 10, 13], and graph summarization in the distributed environment [27], have also been studied.

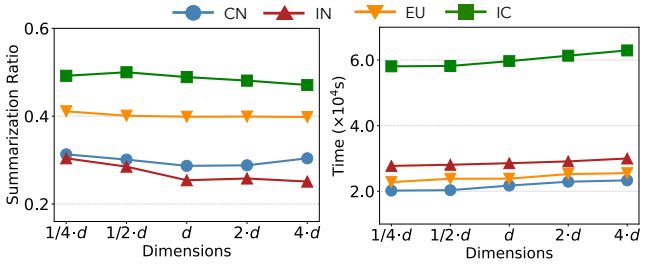

Figure 11: Poligras with Different Embedding Dimensions d.

Policy Learning and Optimization. Policy gradient [25, 29, 38, 40, 44] is one of the widely used models for online or deep reinforcement learning [1, 30, 39], which trains an agent interacting with the environment in order to achieve the highest overall rewards. In the policy gradient framework, there is a policy agent π (with parameter θ) employed to interact with the environment and make a series of decisions over time. At each timestamp t, the agent receives the state s_t from the environment, performs an action a_t following the policy $\pi(a_t|s_t;\theta)$, and receives a corresponding reward r_t. The environment then transits to the next state s_{t+1}. This iterative process continues until the policy agent reaches a termination state. The objective of policy gradient is to optimize the policy agent (*i.e.*, θ) to maximize total rewards, or alternatively, to maximize accumulated rewards after certain steps [29]. In this framework, the policy $\pi(a_t|s_t;\theta)$ can be optimized by updating parameter θ with the gradient ascent method [44]. Inspired by the idea of policy gradient, we transform in this paper the problem of graph summarization into a policy optimization problem, where the graph summarization process translates to a sequence of supernode selection and merging actions determined by the supernode selection policy, $\mathbb{P}(\cdot)$. Each supernode selection decision is similar to an agent performing actions to interact with the environment. By optimizing $\mathbb{P}(\cdot)$, the cumulative summarization rewards of selected supernode pairs can be maximized and, as a result, the cost of graph summaries is minimized.

6 CONCLUSION

With the prevalence of large graphs, it becomes indispensable and imperative to concisely represent them for efficient and cost-effective storage, analysis, and visualization in real-world networked applications. In this paper, we proposed a learning-enhanced approach, Poligras, for summarizing large graphs. We reformulated the problem of graph summarization as a sequential decision-making problem, where optimal supernode pairs could be identified and merged based on the novel notion of summarization rewards and the learning-enhanced policy that was trained and optimized by neural networks. Poligras has proven to be highly efficient, effective, scalable, and achieved significantly better graph summarization performance than state-of-the-art solutions.

ACKNOWLEDGMENTS

This work was supported in part by the Army Research Office (ARO Award No. W911NF1810395). Any opinions, findings, and conclusions in this paper are those of the authors and do not necessarily reflect the funding agencies.

REFERENCES

[1] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. 2017. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine* 34, 6 (2017), 26–38.

[2] Till Blume, David Richerby, and Ansgar Scherp. 2020. Incremental and parallel computation of structural graph summaries for evolving graphs. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management (CIKM'20)*. 75–84.

[3] Paolo Boldi, Bruno Codenotti, Massimo Santini, and Sebastiano Vigna. 2004. UbiCrawler: a scalable fully distributed web crawler. *Software: Practice & Experience* 34, 8 (2004), 711–726.

[4] Paolo Boldi, Andrea Marino, Massimo Santini, and Sebastiano Vigna. 2018. BUbiNG: massive crawling for the masses. *ACM Trans. Web* 12, 2 (2018), 1–26.

[5] Paolo Boldi, Marco Rosa, Massimo Santini, and Sebastiano Vigna. 2011. Layered label propagation: a multi-resolution coordinate-free ordering for compressing social networks. In *Proceedings of the 20th international conference on World Wide Web (WWW'11)*. 587–596.

[6] Paolo Boldi and Sebastiano Vigna. 2004. The WebGraph framework I: compression techniques. In *Proceedings of the Thirteenth International World Wide Web Conference (WWW'04)*. 595–601.

[7] Andrei Z Broder, Moses Charikar, Alan M Frieze, and Michael Mitzenmacher. 2000. Min-wise independent permutations. *J. Comput. System Sci.* 60, 3 (2000), 630–659.

[8] Wenfei Fan, Jianzhong Li, Xin Wang, and Yinghui Wu. 2012. Query preserving graph compression. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data (SIGMOD'12)*. 157–168.

[9] Wenfei Fan, Yuanhao Li, Muyang Liu, and Can Lu. 2021. Making graphs compact by lossless contraction. In *Proceedings of the 2021 International Conference on Management of Data (SIGMOD'21)*. 472–484.

[10] Wenfei Fan, Yuanhao Li, Muyang Liu, and Can Lu. 2022. A hierarchical contraction scheme for querying big graphs. In *Proceedings of the 2022 International Conference on Management of Data (SIGMOD'22)*. 1726–1740.

[11] George Fletcher, Jan Hidders, and Josep Llus Larriba-Pey. 2018. *Graph data management: fundamental issues and recent developments* (1st ed.). Springer.

[12] Mahdi Hajiabadi, Jasbir Singh, Venkatesh Srinivasan, and Alex Thomo. 2021. Graph summarization with controlled utility loss. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining (KDD'21)*. 536–546.

[13] Ruoming Jin, Yang Xiang, Ning Ruan, and Haixun Wang. 2008. Efficiently answering reachability queries on very large directed graphs. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data (SIGMOD'08)*. 595–608.

[14] Xiangyu Ke, Arijit Khan, and Francesco Bonchi. 2022. Multi-relation graph summarization. *ACM Trans. Knowl. Discov. Data* 16, 5 (2022), 1–30.

[15] Arijit Khan, Sourav S. Bhowmick, and Francesco Bonchi. 2017. Summarizing static and dynamic big graphs. *Proc. VLDB Endow.* 10, 12 (2017), 1981–1984.

[16] Kifayat Ullah Khan. 2015. Set-based approach for lossless graph summarization using locality sensitive hashing. In *the 31st IEEE International Conference on Data Engineering Workshops (ICDEW'15)*. 255–259.

[17] Diederik P. Kingma and Jimmy Ba. 2015. Adam: a method for stochastic optimization. In *the 3rd International Conference on Learning Representations (ICLR'15)*.

[18] Jihoon Ko, Yunbum Kook, and Kijung Shin. 2020. Incremental lossless graph summarization. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD'20)*. 317–327.

[19] Danai Koutra, U Kang, Jilles Vreeken, and Christos Faloutsos. 2015. Summarizing and understanding large graphs. *Stat. Anal. Data Min.* 8, 3 (2015), 183–202.

[20] K Ashwin Kumar and Petros Efstathopoulos. 2018. Utility-driven graph summarization. *Proceedings of the VLDB Endowment* 12, 4 (2018), 335–347.

[21] Kyuhan Lee, Hyeonsoo Jo, Jihoon Ko, Sungsu Lim, and Kijung Shin. 2020. SSumM: sparse summarization of massive graphs. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD'20)*. 144–154.

[22] Kristen LeFevre and Evimaria Terzi. 2010. GraSS: Graph structure summarization. In *Proceedings of the 2010 SIAM International Conference on Data Mining (SDM'10)*. SIAM, 454–465.

[23] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2005. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining (KDD'05)*. 177–187.

[24] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2007. Graph evolution: densification and shrinking diameters. *ACM transactions on Knowledge Discovery from Data (TKDD)* 1, 1 (2007), 2–42.

[25] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2016. Continuous control with deep reinforcement learning. In *the 4th International Conference on Learning Representations (ICLR'16)*.

[26] Yu-Ru Lin, Hari Sundaram, and Aisling Kelliher. 2008. Summarization of social activity over time: people, actions and concepts in dynamic networks. In *Proceedings of the 17th ACM conference on Information and knowledge management (CIKM'08)*. 1379–1380.

[27] Xingjie Liu, Yuanyuan Tian, Qi He, Wang-Chien Lee, and John McPherson. 2014. Distributed graph summarization. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management (CIKM'14)*. 799–808.

[28] Yike Liu, Tara Safavi, Abhilash Dighe, and Danai Koutra. 2018. Graph summarization methods and applications: A survey. *Comput. Surveys* 51, 3 (2018), 1–34.

[29] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning (ICML'16)*. 1928–1937.

[30] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529–533.

[31] Saket Navlakha, Rajeev Rastogi, and Nisheeth Shrivastava. 2008. Graph summarization with bounded error. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data (SIGMOD'08)*. 419–432.

[32] Qiang Qu, Siyuan Liu, Christian S Jensen, Feida Zhu, and Christos Faloutsos. 2014. Interestingness-driven diffusion process summarization in dynamic networks. In *Machine Learning and Knowledge Discovery in Databases: European Conference (ECML/PKDD'14)*. 597–613.

[33] Matteo Riondato, David García-Soriano, and Francesco Bonchi. 2017. Graph summarization with quality guarantees. *Data mining and knowledge discovery* 31, 2 (2017), 314–349.

[34] Jorma Rissanen. 1978. Modeling by shortest data description. *Automatica* 14, 5 (1978), 465–471.

[35] Sherif Sakr, Angela Bonifati, Hannes Voigt, Alexandru Iosup, Khaled Ammar, Renzo Angles, Walid Aref, Marcelo Arenas, Maciej Besta, Peter A. Boncz, Khuzaima Daudjee, Emanuele Della Valle, Stefania Dumbrava, Olaf Hartig, Bernhard Haslhofer, Tim Hegeman, Jan Hidders, Katja Hose, Adriana Iamnitchi, Vasiliki Kalavri, Hugo Kapp, Wim Martens, M. Tamer Özsu, Eric Peukert, Stefan Plantikow, Mohamed Ragab, Matei R. Ripeanu, Semih Salihoglu, Christian Schulz, Petra Selmer, Juan F. Sequeda, Joshua Shinavier, Gábor Szárnyas, Riccardo Tommasini, Antonino Tumeo, Alexandru Uta, Ana Lucia Varbanescu, Hsiang-Yun Wu, Nikolay Yakovets, Da Yan, and Eiko Yoneki. 2021. The Future is Big Graphs: A Community View on Graph Processing Systems. *Commun. ACM* 64, 9 (2021), 62–71.

[36] Neil Shah, Danai Koutra, Tianmin Zou, Brian Gallagher, and Christos Faloutsos. 2015. Timecrunch: interpretable dynamic graph summarization. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining (KDD'15)*. 1055–1064.

[37] Kijung Shin, Amol Ghoting, Myunghwan Kim, and Hema Raghavan. 2019. SWeG: lossless and lossy summarization of web-scale graphs. In *The World Wide Web Conference (WWW'19)*. 1679–1690.

[38] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. 2014. Deterministic policy gradient algorithms. In *International conference on machine learning (ICML'14)*. 387–395.

[39] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

[40] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems (NIPS'99)* 12 (1999), 1057–1063.

[41] Nan Tang, Qing Chen, and Prasenjit Mitra. 2016. Graph stream summarization: From big bang to big crunch. In *Proceedings of the 2016 International Conference on Management of Data (SIGMOD'16)*. 1481–1496.

[42] Yuanyuan Tian, Richard A. Hankins, and Jignesh M. Patel. 2008. Efficient aggregation for graph summarization. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data (SIGMOD'08)*. 567–580.

[43] Ioanna Tsalouchidou, Francesco Bonchi, Gianmarco De Francisci Morales, and Ricardo Baeza-Yates. 2020. Scalable dynamic graph summarization. *IEEE Transactions on Knowledge and Data Engineering* 32, 2 (2020), 360–373.

[44] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8, 3 (1992), 229–256.

[45] Quinton Yong, Mahdi Hajiabadi, Venkatesh Srinivasan, and Alex Thomo. 2021. Efficient graph summarization using weighted LSH at billion-scale. In *Proceedings of the 2021 International Conference on Management of Data (SIGMOD'21)*. 2357–2365.

[46] Peixiang Zhao, Xiaolei Li, Dong Xin, and Jiawei Han. 2011. Graph Cube: on warehousing and OLAP multidimensional networks. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data (SIGMOD'11)*. 853–864.