Task-Oriented Genetic Activation for Large-Scale Complex Heterogeneous Graph Embedding

Zhuoren Jiang
Department of Information Resources Management, Zhejiang University, Hangzhou, China
jzr1986@gmail.com

Zheng Gao
Indiana University Bloomington
Bloomington, IN, USA
gao27@indiana.edu

Jinjiong Lan
Alibaba Group
Hangzhou, China
jinjiong.ljj@alibaba-inc.com

Hongxia Yang
Alibaba Group,
Hangzhou, China
yang.yhx@alibaba-inc.com

Yao Lu*
Sun Yat-sen University
Guangzhou, China
luyao23@mail.sysu.edu.cn

Xiaozhong Liu*
Indiana University Bloomington
Bloomington, IN, USA
liu237@indiana.edu

ABSTRACT

The recent success of deep graph embedding innovates the graphical information characterization methodologies. However, in real-world applications, such a method still struggles with the challenges of heterogeneity, scalability, and multiplex. To address these challenges, in this study, we propose a novel solution, **G**enetic h**E**terogeneous g**R**aph e**M**bedding (**GERM**), which enables flexible and efficient task-driven vertex embedding in a complex heterogeneous graph. Unlike prior efforts for this track of studies, we employ a task-oriented genetic activation strategy to efficiently generate the "**E**dge **T**ype **A**ctivated **V**ector" (**ETAV**) over the edge types in the graph. The generated ETAV can not only reduce the incompatible noise and navigate the heterogeneous graph random walk at the graph-schema level, but also activate an optimized subgraph for efficient representation learning. By revealing the correlation between the graph structure and task information, the model interpretability can be enhanced as well. Meanwhile, an activated heterogeneous skip-gram framework is proposed to encapsulate both topological and task-specific information of a given heterogeneous graph. Through extensive experiments on both scholarly and e-commerce datasets, we demonstrate the efficacy and scalability of the proposed methods via various search/recommendation tasks. GERM can significantly reduces the running time and remove expert-intervention without sacrificing the performance (or even modestly improve) by comparing with baselines.

CCS CONCEPTS

• **Computing methodologies** → *Learning latent representations.*

KEYWORDS

Heterogeneous Graph Embedding; Edge Type Activation

*Corresponding author

ACM Reference Format:
Zhuoren Jiang, Zheng Gao, Jinjiong Lan, Hongxia Yang, Yao Lu, and Xiaozhong Liu. 2020. Task-Oriented Genetic Activation for Large-Scale Complex Heterogeneous Graph Embedding. In *Proceedings of The Web Conference 2020 (WWW '20), April 20–24, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3366423.3380230

1 INTRODUCTION

Graph[34] provides a ubiquitous non-linear data structure to characterize the real-world information, i.e., the objects and the interactions between the objects [10]. While the data instances/properties with higher complexity can be interconnected, graph mining and analysis methods are playing an increasingly important role to probe the "deep dependence" among different types of information, which has been successfully employed to address a variety of emerging applications across many disciplines, such like social graphs, citation graphs, biological graphs and e-commerce graphs [31]. Graph embedding algorithms that aim to learn the low-dimensional representations of the vertexes in a graph, are attracting attention recently and have proved very useful for a wide variety of prediction and graph analysis tasks [11, 14, 15, 17–19, 27]. However, graph embedding could be a challenging problem in the real world due to the following reasons:

Heterogeneity. Compared with the homogeneous graph, the heterogeneous graph has been demonstrated as a more effective way to model complex data for many applications, and it provides important potentials to characterize various kinds of interactions among the objects [16, 20, 21]. However, classical graph embedding methods, e.g., [12, 19, 27, 35], show their incompatibility for the heterogeneous graphs, while ignoring the heterogeneous type information of the vertexes/edges on the graph. For instance, one vertex may associate with different types of edges, and its context (vertexes) sampling probability should depend on both transition probability and the associated edge-type(s). When the graph is becoming increasingly complex, noisy information (e.g., incompatible types of edges) may pollute the context vector and hurt the embedding result as well [32].

Scalability. As shown in Table 1, the sizes of benchmark datasets for graph embedding experiments are usually small, and most of the existing graph embedding methods haven't been validated through

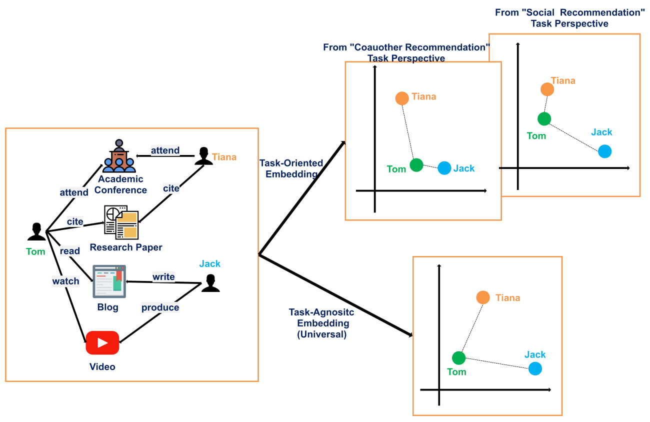

Figure 1: A toy example of different embedding spaces for a given heterogeneous graph.

Table 1: The sizes of several popular graph datasets for evaluating graph embedding algorithms

Type	Dataset	#Vertex	#Edge
Web Graph	WebKB [23]	877	1,608
	Polblogs [2]	1,222	16,715
Word Graph	Wikipedia-word [24]	4,777	184,812
Citation Graph	Cora [30]	2,708	5,429
	Citeseer [30]	3,312	4,732
Biological Graph	PPI [7]	3,890	76,584
Social Graph	Amherst [37]	2,021	81,492
	Hamilton [37]	2,118	87,486
	Mich [37]	2,933	54,903
	Rochester [37]	4,145	145,305
	BlogCatalog [41]	10,312	333,983

This table doesn't cover all datasets.

very large graphs, mainly because they are computationally expensive and often memory intensive [42, 43]. Unfortunately, the cost of generating heterogeneous graph embedding can be significantly higher, especially when expert knowledge enabled approaches, e.g., metapath [11, 34], is not available.

Multiplex. To the best of our knowledge, most existing graph embedding algorithms are trying to learn the universal (one space) embedding for a given graph. On a heterogeneous graph, however, different downstream tasks may need different embedding spaces (multiple spaces), because various kinds of edges may contribute to different tasks diversely. Figure 1 depicts a toy example of a heterogeneous graph, which hosts different types of information, i.e., *Tom* and *Tiana* attend the same academic conference and cite the same paper in their research papers; meanwhile, *Tom* likes reading/watching the blog and video generated by *Jack*. While a universal embedding tells the "general distances" among these people (vertexes); for different search/recommendation scenarios, the task-oriented embeddings should be utilized to optimize the outcomes. For instance, as Figure 1 shows, for "social recommendation", *Tom* and *Jack* should be closer, and *Tom* should be more similar to *Tiana* in a "scholarly recommendation" context. Behind each task, the graph embedding model should be able to sample the informative edges while ignoring the noisy ones in terms of task-oriented edge-type suitability. Note that, in a complex graph, the algorithm should be able to learn the task-oriented multiplex representations automatically.

To address these challenges, we propose a novel solution, **G**enetic h**E**terogeneous g**R**aph e**M**bedding (**GERM**), which can auto-navigate the random walker to sample the context vertexes concerning a target (search/recommendation) task. We hypothesize that,

- For the accuracy reason, on a complex heterogeneous graph, not all the edge types are equally important for the target task, and a task-oriented edge-type-navigation can be significant for graph representation learning.
- For the efficiency reason, it can be intractable to perform complex inferences on the entire graph, and a "local mining" on the subgraph can be more efficient.

Based on task-annotation information, we use the genetic activation strategy [33] to generate an "Edge Type Activated Vector" (ETAV) for a heterogeneous graph. ETAV can not only filter the incompatible noise and navigate the heterogeneous graph random walk, but also implicitly activate an optimized subgraph for efficient (local) representation learning. Meanwhile, the activated graph structure (vertexes and edges) information can improve the model interpretability by revealing the task-oriented edge-type suitability. An activated heterogeneous skip-gram framework is proposed to abstract the geometrical and task-driven information from given heterogeneous graph. In more detail, we develop an activated heterogeneous negative sampling based method for accurate prediction of vertex's heterogeneous graph neighborhood (context). To the best of our knowledge, this is the first study to investigate the task-oriented heterogeneous graph representation learning along with graph embedding efficiency problems.

The major contributions of this paper can be summarized as follows:

(1) We propose a novel solution, GERM, to address the challenges of heterogeneity, scalability, and multiplex for ranking tasks in a large-scale complex heterogeneous graph (e.g., the vertex number is over 1 million and the edge-type number is over 10).
(2) A task-oriented ETAV formulation is proposed to efficiently preserve both the structures and semantics of a heterogeneous network for a given search or recommendation task. The learned ETAV can enhance the interpretability by revealing the correlation between the activated graph structure and task information.
(3) A genetic evolutionary activation algorithm and an activated heterogeneous skip-gram framework are proposed for task-oriented salable heterogeneous graph representation learning. Meanwhile, the proposed genetic activation algorithm can be applied as a flexible "plug-in" to existing graph embedding models without extra modification.
(4) Through extensive experiments on both scholarly and e-commerce datasets[1], we demonstrate the efficiency and scalability of the proposed methods for various tasks. We find

[1]To help other scholars reproduce the experiment outcome, we will release the datasets via GitHub. https://github.com/GraphEmbedding/GERM.

GERM can significantly reduces the running time and remove expert-intervention without sacrificing the performance (or even modestly improve) by comparing with baselines.

2 PRELIMINARIES

Definition 1. ***Heterogeneous Graph***. *$G = (V, E, \mathbb{N}, \mathbb{Z})$, where V denotes the vertex set, and $E \subseteq V \times V$ denotes the edge set. $\mathbb{N}$ denotes the set of vertex-types and $\mathbb{Z}$ denotes the set of edge-types. $|\mathbb{N}| + |\mathbb{Z}| > 2$ [11, 34].*

Each heterogeneous graph associates with a graph schema $S_G = (\mathbb{N}, \mathbb{Z})$, which can be used to specify type constraints on the sets of vertexes and edges of a heterogeneous graph. [21].

Definition 2. ***Heterogeneous Graph Complexity***. *In this study, heterogeneous graph complexity is defined as: $H = |\mathbb{Z}|$, where $|\mathbb{Z}|$ denotes the number of edge-types in the graph schema [5].*

In a simple graph, classical methods, e.g., metapath, can be efficient while the expert can propose a high-quality ranking hypothesis for the random walk. However, the search space of metapath solutions is $|\mathbb{Z}|^l$ (a permutation with repetition problem [28]), where l is the length of metapath. While $|\mathbb{Z}|$ increases, the expert-navigated random walk becomes increasingly expensive. For instance, when $|\mathbb{Z}| = 10$, there could be 10^5 potential metapaths (*length = 5*) on the graph schema, and the vertex context sampling (for graph embedding) can be unaffordable.

Definition 3. ***Heterogeneous Graph Embedding***. *Given a heterogeneous graph $G = (V, E, \mathbb{N}, \mathbb{Z})$, The goal of graph embedding is to obtain the latent vertex representations by mapping vertexes into a low-dimensional space $\mathbb{R}^d$, $d \ll |V|$. The mapping function from multi-typed vertexes to feature representations can be presented as: $f : V \to \mathbb{R}^d$.*

Unlike prior studies, graph embedding in this study is task-oriented. From the vertex (relevance) ranking viewpoint, a ranking task on the graph is a conditional probability $\Pr(v_i|v_j)$ between two vertexes (candidate node v_i and query node v_j)[13]. $\Pr(v_i|v_j) = \Delta\left(f(v_i), f(v_j)\right)$, where f is graph embedding function, and Δ is a probability scoring function based on the learned graph embeddings. The more relevant of v_i and v_j, the greater $\Pr(v_i|v_j)$ is. For the same graph, when facing multi-tasks, f can vary to optimize the ranking outcomes. Specifically, we propose a novel method to enhance the efficiency of representation learning function f when experiencing large-scale complex heterogeneous graphs (e.g., $|V| > 10^6$ and $H \geqslant 10$).

3 METHODOLOGY

3.1 Overall Framework

As aforementioned, classical graph embedding methods suffer from heterogeneity, scalability and multiplex problems. In this study, we propose GERM model[2] to address the search or recommendation tasks on a complex heterogeneous graph. Figure 2 depicts the framework of GERM and common notations are listed in Table 2. We summarize the features of GERM as follows:

[2]https://github.com/GraphEmbedding/GERM

Table 2: A summary of common notations

Symbol	Definition
G	Graph
S_G	Graph Schema
V, v	Vertex set and vertex instance
E, e	Edge set and edge instance
$\mathbb{N}$	Vertex-type set
$\mathbb{Z}$	Edge-type set
Z	Edge-type instance
H	Heterogeneous graph complexity
f	Graph embedding function
A_E	ETAV
a_z	Activation state for z_{th} edge-type
V_{A_E}	Activated vertex set
$\mathbb{N}_{A_E}$	Activated vertex-types
$\mathbb{Z}_{A_E}$	Activated edge-types
L	Labeled vertex pairs set (task information)
C	Chromosome, a candidate ETAV for a given task
g	Gene, activation state for each edge-type
P	Population, a set of chromosomes
F	Fitness function for evaluating the candidate ETAV
Gen	Generation, population in each iteration

Edge Type Activated Vector (ETAV). Unlike previous studies [18, 22] trying to learn the edge-type probability distribution, we aim to preserve the critical edge-types while removing the noisy (incompatible) ones given a specific ranking task. Namely, we try to learn a binary ETAV (a.k.a. a heterogeneous graph simplification model) at the graph schema level: $A_E = \left\{a_1, ..., a_z, ..., a_{|\mathbb{Z}|}\right\}$, a_z is the "0/1" activation state for z_{th} edge-type. Note that, given the same graph, when the ranking tasks are different, ETAV could be nonidentical.

Optimization and Efficiency Balance. The binary edge-type activation strategy is proposed for juggling optimized performance and efficiency, which is based on the following assumptions:

- For a given ranking task on a complex heterogeneous graph, not all the edge-types are equally important. As Figure 1 illustrates, some incompatible edge-types could pollute the ranking results.
- For a large-scale complex heterogeneous graph, it could be very time-consuming to learn the edge-type fitness distribution for the target task. For instance, [18] utilized a k-shortest path based approach to learn the edge-type fitness distribution and its computation cost is $O(|E| + |V|\log|V| + k)$. The computational complexity could be greatly reduced if we only infer a "0/1" activation switch for each edge-type concerning the target task.
- The size of the graph could be further reduced based on the generated ETAV. In other words, by activating a subgraph, the graph embedding cost can be significantly reduced.

Activated Random Walk. As Figure 2 (b) shows, the learned ETAV can guide the graph navigation at the graph schema level, i.e., assist the walker to walk on the most informative types of edges

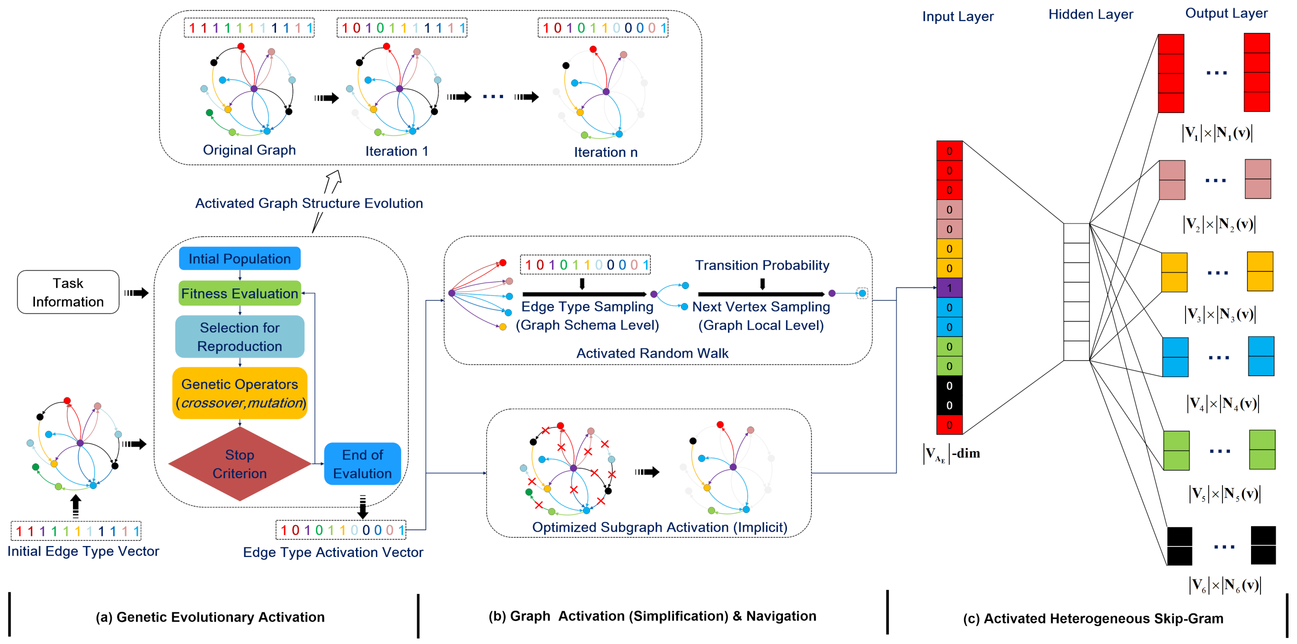
Original Graph
Iteration 1
Iteration n
Activated Graph Structure Evolution
Task Information
Intial Population
Fitness Evaluation
Selection for Reproduction
Genetic Operators (crossover,mutation)
Stop Criterion
End of Evalution
Initial Edge Type Vector
Edge Type Activation Vector
Edge Type Sampling (Graph Schema Level)
Transition Probability
Next Vertex Sampling (Graph Local Level)
Activated Random Walk
Optimized Subgraph Activation (Implicit)
Input Layer
Hidden Layer
Output Layer
$|V_{A_E}|$-dim
$|V_1|\times|N_1(v)|$
$|V_2|\times|N_2(v)|$
$|V_3|\times|N_3(v)|$
$|V_4|\times|N_4(v)|$
$|V_5|\times|N_5(v)|$
$|V_6|\times|N_6(v)|$
(a) Genetic Evolutionary Activation
(b) Graph Activation (Simplification) & Navigation
(c) Activated Heterogeneous Skip-Gram

Figure 2: An Illustration of Genetic hEterogeneous gRaph eMbedding (GERM) Framework. (a) In task-oriented genetic edge-type activation, the ETAV will evolve based on task information. (b) Activated random walk is employed on a optimized subgraph. (c) In activated heterogeneous skip-gram, $|V_{A_E}|$ denotes the size of activated vertexes, $|V_i|$ denotes the size of i_{th} type of vertexes. $N_i(v)$ specifies the size of i_{th} type vertexes in v's neighborhood.

for the given task. The activated random walk process is defined as follows[3]:

(1) For the i_{th} vertex v_i in the walk:
   (a) Probabilistically draw an edge-type Z from A_E
   (b) For the generated edge-type Z
      (i) Based on the transition probability $\Pr(Z(v_{i+1}|v_i))$, probabilistically draw one vertex v_{i+1} as the destination;
      (ii) Move forward to v_{i+1}.

The time complexity is $O(|\mathbb{Z}_{A_E}| + |N_Z(v)|)$ per walk, where $|\mathbb{Z}_{A_E}|$ is the number of activated edge-types, and $|N_Z(v)|$ is the number of activated neighbor vertexes. Through a series of experiments, we demonstrate $|\mathbb{Z}_{A_E}|$ and $|N_Z(v)|$ can be considerably reduced with ETAV. More detailed information can be found in the experiment section.

Activated Heterogeneous Skip-Gram. The representation learning is based on an activated heterogeneous skip-gram architecture. For this study, we optimize the following objective function:

$$\max_f \sum_{v\in V_{A_E}} \sum_{n\in\mathbb{N}_{A_E}} \sum_{v_n^c\in N_n(v)} \log\Pr(v_n^c|f(v)) \tag{1}$$

where $N_n(v)$ denotes v's graph neighborhood ("context") with the n_{th} type of vertexes. We leverage activated random walk to generate activated graph neighborhood, which can eliminate the noisy vertexes and make neighborhood relevant to the given task. Furthermore, as shown in Figure 2 (c), the activated heterogeneous skip-gram specifies different sets of multinomial distributions for different (activated) vertex-types in the output layer. On the contrary, the ordinary skip-gram model, utilized in [12, 27], only generates one set of multinomial distributions regardless of vertex types. More specifically, for the activated heterogeneous skip-gram, the n_{th} type multinomial distribution dimension is determined by the number of vertexes with n_{th} type [11, 18]. The activated vertex-type set $\mathbb{N}_{A_E}$ is inferred from ETAV. In objective function (1), $\Pr(v_n^c|f(v))$ can be calculated as:

$$\Pr(v_n^c|f(v)) = \frac{\exp(f(v_n^c)\cdot f(v))}{\sum_{u_n\in V_n}\exp(f(u_n)\cdot f(v))} \tag{2}$$

However, optimizing the above Equation (2) is computationally expensive, which requires the summation over the entire V_n to calculate $\Pr(v_n^c|f(v))$. To address this problem, we adopt a novel activated negative sampling function:

$$\log\sigma(f(v_n^c)\cdot f(v)) + \sum_{k=1}^{K}\mathbb{E}_{u_n^k\sim\Pr_{A_E}^n}\Big[\log\sigma(-f(u_n^k)\cdot f(v))\Big] \tag{3}$$

where σ is the sigmoid function; $\Pr_{A_E}^n$ ensures the negative vertexes u_n^k are sampled from the activated sub-graph according to their type information. Equation (3) can be optimized by using the stochastic gradient descent algorithm. The time complexity for activated heterogeneous skip-gram is $O(\log(|V_{A_E}|))$. In practice, the number of activated vertexes $|V_{A_E}|$ can be much smaller than the total vertex number $|V|$. For instance, in the experiment, GERM only activated 23.7% of the total vertexes, while the task performance outperformed the classical methods utilizing the whole

[3]To avoid walking into a dead-end, the directions of edges are ignored in the activated random walk algorithm.

graph; more detailed information can be found in Section 4.2. Compare to the original skip-gram algorithms (whose time complexity is $O(\log(|V|))$, GERM can be much more efficient. Meanwhile, the space complexity of GERM is $O|E|$.

3.2 Genetic Activation Strategy

In this section, we focus on the task-oriented "0/1" edge-type activation problem. Theoretically, the search space for this problem is $2^{|\mathbb{Z}|}$. For a large complex heterogeneous graph (e.g., $|\mathbb{Z}| \geqslant 10$), the cost is still high.

Gradient Descent vs. Genetic Algorithm. As described in section 2, a ranking/recommendation task on the graph can be defined as a conditional probability $\Pr(v_i|v_j) = \Delta\left(f(v_i), f(v_j)\right)$. Intuitively, a straightforward strategy is to compare the qualities of graph embeddings for all candidate activation solutions. It needs to explicitly compute the graph embeddings, which requires Gradient Descent (GD) based optimization. However, the computational cost of GD is decided by the objective function [6], which is always expensive for large scale graph embedding (e.g., $O(\log|V|)$ for skip-gram model). In this study, we proposed a more efficient graph optimization/simplification method by using the Genetic Algorithm (GA) [33], which is a robust/stochastic search technique modeled on the principles and concepts of natural selection and evolution. Unlike GD to directly calculate graph embedding $f(\cdot)$, GA defines a fitness function $F(v_i, v_j)$ to estimate $\Pr(v_i|v_j)$, which can be much more efficient for graph computation (e.g., $O(1)$ for direct neighborhood comparison). Meanwhile, while GD is not directly targeting to find an optimal ETAV, GA is able to learn the optimized solution in the search space. When the graph schema is complex, GD may be stuck on the local minimum; but GA is global optimum [3], which can be particularly efficient and effective for the proposed task.

Genetic Edge-type Activation Algorithm. In this study, as Figure 2 (a) shows, we solve the edge-type activation problem via a genetic algorithm framework. The Algorithm 1 is the pseudo-code for genetic edge-type activation. The task information comes from the labeled vertex (training) pairs set L, which is generated based on the task-specified relevance (e.g., two collaborators for co-author recommendation task).

In Algorithm 1, selection operations (line 9-12) are based on the elite principle [4], ensuring that the high-quality ETAV candidates have a higher probability of passing its information to the next generation for further navigation. Crossover operations (line 13-16) and mutation operations (line 17-20) are both used to explore the possible ETAV spaces. Finally, via an iterative evolution process, the best chromosome (solution) is selected as the final ETAV. Theoretically, the proposed genetic edge-type activation algorithm is ***transparent*** to the underlying graph embedding techniques, which has the potential to be treated as a ***"plug-in"*** for various graph embedding models.

As Figure 2 (a) and Algorithm 1 demonstrate, for the evolution iterations, it is important to accurately and effectively evaluate the fitness of a candidate solution. Then the ETAV can evolve in the right direction. To address this problem, as aforementioned, we explore three efficient fitness functions to approximate the task

Algorithm 1 Graph Edge-Type Activation Algorithm: a Genetic Evolutionary Approach

```
Input: heterogeneous graph $G = (V, E, \mathbb{N}, \mathbb{Z})$, labeled vertex pairs set $L$,
    population size $|P|$, crossover rate $r_{co}$, mutation rate $r_{mu}$.
Output: optimized edge-type activation vector $A_E$.
 1: Initialization[4]: chromosome population $P^0 = \left\{C_1^0, C_2^0, \ldots, C_{|P|}^0\right\}$; generation $Gen = 0$; $A_E = \{g_1, \ldots, g_z, \ldots, g_{|\mathbb{Z}|}\}$, $g_z = 1$.
 2: repeat
 3:     for all $C_c^{Gen} \in P^{Gen}$ do
 4:         for all $\{v_i, v_j\} \in L$ do
 5:             $S_c$ +=FitnessEvaluation($G, C_c^{Gen}, \{v_i, v_j\}$)
 6:         end for
 7:     end for
 8:     Assign $C_{best}$ (the chromosome with best fitness score) to $A_E$
 9:     Select $PA_1^{Gen} =< C_{1,i}^{Gen}, C_{1,j}^{Gen} >$, $C_{1,i}^{Gen} = C_{best}$ and $C_{1,j}^{Gen}$ is
        probabilistically sampled based on fitness scores.
10:     for $p = 2$ to $\frac{|P|}{2} - 1$ do
11:         Select $PA_p^{Gen} =< C_{p,i}^{Gen}, C_{p,j}^{Gen} >$, $C_{p,i}^{Gen}$ and $C_{p,j}^{Gen}$ are both
            probabilistically sampled based on fitness scores.
12:     end for
13:     for all $PA_p^{Gen} \in PA^{Gen}$ do
14:         Randomly sample $z$ gene positions
15:         Between $C_{p,i}^{Gen}$ and $C_{p,j}^{Gen}$, exchange the genes on sampled posi-
            tions with the probability of $r_{co}$, to generate $C_{p,i}^{Gen'}$ and $C_{p,j}^{Gen'}$
16:     end for
17:     for all $C_c^{Gen'} \in PA^{Gen}$ do
18:         Randomly sample $z$ gene positions
19:         In $C_c^{Gen'}$, reverse the genes' values on sampled positions with
            the probability of $r_{mu}$, to generate $C_c^{Gen''}$.
20:     end for
21:     The new generate $P^{Gen+1} = \left\{C_1^{Gen''}, C_s^{Gen''}, \ldots, C_{|P|}^{Gen''}\right\}$.
22: until Converge
```

conditional probability $\Pr(v_i|v_j)$ for a labeled vertex pair $\{v_i, v_j\}$. The basic fitness evaluation criterion is to ensure the labeled pair of vertexes deserve the highest possible score.

Greedy, this function is based on the empirical evidence in the academic environment[26], where $\Gamma_C^G(v)$ is activated neighbor set of v in G given C (a candidate ETAV).

$$F_{greedy}(v_i, v_j) = \left|\Gamma_C^G(v_i)\right| \cdot \left|\Gamma_C^G(v_j)\right| \tag{4}$$

Refine, this function is based on a natural intuition that vertexes that share similar neighbors tend to be similar to each other. It refines the simple counting of common neighbors by boosting the wright of the rarer neighbors [1].

$$F_{refine}(v_i, v_j) = \sum_{v_z \in \Gamma_C^G(v_i) \bigcap \Gamma_C^G(v_j)} \frac{1}{\log\left|\Gamma_C^G(v_z)\right|} \tag{5}$$

Edge, this "edge-level" function is proposed by weighting the important edges (with large edge-weight) to the common neighbors more heavily. In this function, v_z denotes the common neighbours of $\{v_i, v_j\}$ in the activated subgraph. $E_{i,j}^C$ denotes the activated edge set (between v_i and v_j) given C. $E_C(v)$ represents the activated edge set connecting to v given C. $\omega(e)$ is the edge-weight of e.

$$F_{edge}(v_i, v_j) = \frac{\sum_{e_{i,z} \in E_{i,z}^C}[\omega(e_{i,z})] + \sum_{e_{j,z} \in E_{j,z}^C}[\omega(e_{j,z})]}{\sum_{e_n \in E_C(v_i)}[\omega(e_n)] + \sum_{e_m \in E_C(v_j)}[\omega(e_m)]} \tag{6}$$

[4]In experiments, we hold the constraint that the activated subgraph must include testing vertex.

These fitness assessments all focus on the local topology comparison in the activated subgraph, where the immediate neighbors' generation can be $O(1)$ by using a hash mapping structure. The performance of different fitness functions will be compared in Section 4.2.

4 EXPERIMENT

4.1 Dataset and Experiment Setting

Dataset[5]. As aforementioned, we are targeting the large-scale complex heterogeneous graph (i.e., $|V| > 10^6$ and $H \geqslant 10$). To the best of our knowledge, no public graph benchmark datasets meet this requirement. To address this problem, we collected and constructed both scholarly and e-commerce datasets to verify the performance and efficiency of the proposed GERM. The statistics of both datasets are shown in Table 3.

Table 3: Statistics of the datasets

Dataset	#Vertex	#V-Type	#Edge	#E-Type
Scholarly	1,613,872	3	10,827,944	10
E-commerce	1,007,051	8	7,315,490	22

Scholarly Dataset (10 types of edges). This dataset is collected from AMiner Computer Science (CS) Dataset[36]. Based on the cleaned vertex set provided by [11], we explore the relations from raw data and generate an academic heterogeneous graph with 3 types of vertexes (authors, papers, and venues) and 10 types of edges (e.g., citation, coauthorship, publish, etc).

E-commerce Dataset (22 types of edges). This dataset is collected from a world-leading online consumer-to-consumer platform. Except for user and product, this dataset also provides shopping feeds, e.g., text content focusing on product descriptions, fashion, entertainment, pricing, and special promotions, that users could read and share while they shop online. The platform needs to recommend the relevant feeds to the target users. The interests of users can be learned via shopping history. The constructed graph has 8 types of vertexes (e.g., user, feed, product, seller, review, etc.) and 22 types of edges (e.g., share, follow, buy and comment, etc.).

Task and Evaluation Metric. For this study, we validated the proposed algorithm via 3 tasks: *citation recommendation*, *coauthor recommendation* (both on the scholarly graph) and *feeds recommendation* (on the e-commerce graph). Different models were compared by using Mean Average Precision (MAP), Normalized Discounted Cumulative Gain (NDCG), Precision (P) and Mean Reciprocal Rank (MRR). For each task, a number of edges of the target edge-type were randomly selected as the ground truth for the algorithm evaluation, i.e., "paper citation" edges for citation recommendation task, "coauthorship" edges for coauthor recommendation task, and "user-browse-feed" edges for feeds recommendation task. The selected edges were removed from the graphs before the experiment. For citation and feeds recommendation tasks, the relevance of the target paper or feed would be 0 or 1. For coauthor recommendation, a score of 0-4 was utilized to characterize the importance of the recommended author[6].

Baselines. We compared with two groups of graph embedding algorithms:

Homogeneous Graph Embedding: (1) DeepWalk[7] [27], denoted as **DW**. (2) LINE[7] [35], aimed at preserving first-order and second-order proximity in concatenated embeddings and denoted as **LINE**. (3) Node2vec[7] [12], denoted as **N2V**, where we tuned return parameter p and in-out parameter q and picked up a best-performed parameter setting for the experiment, as suggested by [12]. (4) Graph Convolutional Networks[8] [19], we trained convolutional neural networks on an adjacency matrix of the graph and a topological feature matrix of vertexes in a vertex classification task (semi-supervised task), as suggested by [19], denoted as **GCN**.

Heterogeneous Graph Embedding: (5) Metapath2vec++[9] [11]. This algorithm learned heterogeneous graph embeddings via metapath based random walk and heterogeneous negative sampling in the graph. It required a human-defined metapath scheme to guide random walks. For the citation and coauthor recommendation tasks, we followed [11], utilized 2 different metapaths, denoted as **M2V(A)** (Author $\overset{have}{\leftarrow}$ Paper $\overset{have}{\rightarrow}$ Author) and **M2V(B)** (Author $\overset{have}{\leftarrow}$ Paper $\overset{publish-in}{\rightarrow}$ Venue $\overset{publish-in}{\leftarrow}$ Paper $\overset{have}{\rightarrow}$ Author), respectively. For feeds recommendation task, we utilized 2 commonly used metapaths, denoted as **M2V(C)** (User $\overset{follow}{\rightarrow}$ Author $\overset{belong}{\leftarrow}$ Feed $\overset{browse}{\leftarrow}$ User) and **M2V(D)** (User $\overset{browse}{\rightarrow}$ Feed $\overset{collect}{\leftarrow}$ User), respectively. (6) The recent state-of-the-art heterogeneous graph embedding model that utilized edge representations with learned heterogeneous metrics, denoted as **HEER**[10] [32].

Experimental Set-up. As DW, LINE, N2V, and GCN are originally designed for homogeneous graphs, for a fair comparison, we have to construct a homogeneous graph based on the heterogeneous one. We first integrated all edges between two vertexes into one edge, then estimated the edge weight (transition probability) by summing all integrated edges. Recall the ranking task definition in section 2, $\Pr(v_i|v_j) = \Delta\left(f(v_i), f(v_j)\right)$. For better comparing and more focusing on the graph embedding function f, in the experiment, we simply used cosine similarity as measurement Δ. The only exception was the baseline HEER [32], which produced an edge-type-specific score for each vertex pair. In order to maximize its potential, we followed the scoring method proposed in the original paper [32].

Meanwhile, for comparison fairness, all the random walk based embedding methods used the same parameters as follows: (1) The number of walks per vertex: 10; (2) the walk length: 80; (3) the vector dimension: 128; (4) the neighborhood size: 10. Most graph embedding methods reported the above parameter settings in their original papers [12, 27]. In the experiment, the negative instances for each testing vertex were 20, 5-fold cross-validation was applied to avoid the evaluation bias.

[5]https://github.com/GraphEmbedding/GERM

[6]We defined a as the number of coauthorships between a candidate author and a testing author. If $a = 0$, score = 0; if $0 < a \leq 2$, score = 1; if $2 < a < 5$, score = 2; if $5 \leq a < 10$, score = 3; if $a \geq 10$, score = 4.

[7]https://github.com/thunlp/OpenNE

[8]https://github.com/tkipf/pygcn

[9]https://ericdongyx.github.io/metapath2vec/m2v.html

[10]https://github.com/GentleZhu/HEER

Table 4: Measures of different graph embedding algorithms

Graph	Task	Algorithm	Map@10	Map@30	NDCG@10	NDCG@30	P@10	P@30	MRR
Scholarly	**Citation Recommendation**	DW [27]	0.5340	0.5642	0.6410	0.6901	0.1823	0.0802	0.6186
		LINE [35]	0.3706	0.4150	0.4880	0.5764	0.1501	0.0793	0.4760
		N2V [12]	0.6966	0.7220	0.7810	0.8133	0.1949	0.0814	**0.7896**
		GCN [19]	0.1968	0.2589	0.3036	0.4483	0.1160	0.0805	0.2943
		M2V(A) [11]	0.4188	0.4655	0.5156	0.6160	0.1433	0.0803	0.5502
		M2V(B) [11]	0.3348	0.3841	0.4496	0.5526	0.1391	0.0806	0.4417
		HEER [32]	0.2000	0.2623	0.3040	0.4523	0.1122	0.0803	0.3093
		GERM$_g$	**$0.7060^{\dagger}$**	**0.7293**	**$0.7887^{\dagger}$**	**0.8167**	**$0.1998^{\dagger\dagger\dagger}$**	**$0.0815^{\dagger\dagger\dagger}$**	0.7877
		GERM$_r$	**0.6985**	**0.7225**	0.7762	0.8088	**$0.1998^{\dagger\dagger\dagger}$**	**$0.0819^{\dagger\dagger\dagger}$**	0.7802
		GERM$_e$	0.2470	0.3016	0.3612	0.4840	0.1262	0.0803	0.3402
	Coauthor Recommendation	DW [27]	0.9293	0.9344	0.9271	0.9307	0.1743	0.0613	0.9520
		LINE [35]	0.6921	0.7077	0.7420	0.7690	0.1562	0.0611	0.7541
		N2V [12]	0.9340	0.9391	0.9305	0.9341	0.1747	**0.0614**	0.9557
		GCN [19]	0.2675	0.3079	0.3618	0.4579	0.1084	0.0608	0.3381
		M2V(A) [11]	0.7833	0.7996	0.7990	0.8314	0.1545	0.0612	0.8427
		M2V(B) [11]	0.6746	0.6891	0.7206	0.7475	0.1587	0.0613	0.7299
		HEER [32]	0.7549	0.7654	0.7886	0.8060	0.1642	0.0612	0.7950
		GERM$_g$	**$0.9406^{\dagger\dagger\dagger}$**	**$0.9452^{\dagger\dagger\dagger}$**	**$0.9348^{\dagger\dagger\dagger}$**	**$0.9373^{\dagger\dagger\dagger}$**	**$0.1755^{\dagger\dagger\dagger}$**	0.0613	**$0.9601^{\dagger\dagger\dagger}$**
		GERM$_r$	**$0.9483^{\dagger\dagger\dagger}$**	**$0.9527^{\dagger\dagger\dagger}$**	**$0.9400^{\dagger\dagger\dagger}$**	**$0.9424^{\dagger\dagger\dagger}$**	**$0.1757^{\dagger\dagger\dagger}$**	**0.0614**	**$0.9662^{\dagger\dagger\dagger}$**
		GERM$_e$	0.8772	0.8833	0.8898	0.8947	0.1727	0.0613	0.9077
E-commerce	**Feeds Recommendation**	DW [27]	0.3548	0.4712	0.5332	0.6692	0.3140	0.2026	0.6368
		LINE [35]	0.1271	0.2598	0.2666	0.4961	0.1892	0.1932	0.3635
		N2V [12]	0.3809	0.4948	0.5597	0.6854	0.3299	0.2038	0.6528
		GCN [19]	0.1213	0.2652	0.2619	0.4894	0.1963	0.1986	0.2702
		M2V(C) [11]	0.1565	0.2894	0.3115	0.5207	0.2204	0.1966	0.3719
		M2V(D) [11]	0.1264	0.2726	0.2803	0.4943	0.2160	0.2008	0.2707
		HEER [32]	0.3223	0.4373	0.5061	0.6385	0.3133	0.1999	0.5709
		GERM$_g$	**$0.4011^{\dagger\dagger\dagger}$**	**$0.5147^{\dagger\dagger\dagger}$**	**$0.5868^{\dagger\dagger\dagger}$**	**$0.6981^{\dagger\dagger\dagger}$**	**$0.3482^{\dagger\dagger\dagger}$**	**$0.2059^{\dagger\dagger\dagger}$**	**0.6588**
		GERM$_r$	**$0.3955^{\dagger\dagger\dagger}$**	**$0.5093^{\dagger\dagger\dagger}$**	**$0.5807^{\dagger\dagger\dagger}$**	**$0.6942^{\dagger\dagger}$**	**$0.3454^{\dagger\dagger\dagger}$**	**$0.2057^{\dagger\dagger}$**	**0.6542**
		GERM$_e$	**$0.3907^{\dagger\dagger\dagger}$**	**$0.4984^{\dagger\dagger\dagger}$**	**$0.5737^{\dagger\dagger\dagger}$**	0.6824	**$0.3355^{\dagger\dagger\dagger}$**	0.1987	**$0.6692^{\dagger\dagger\dagger}$**

T-test: GERM vs baselines; $^{\dagger}p < 0.05$, $^{\dagger\dagger}p < 0.01$, $^{\dagger\dagger\dagger}p < 0.001$

We also compared the performances of fitness variants of the proposed method, denoted as **GERM$_g$** (using "greedy" fitness function for genetic activation), **GERM$_r$** ("refine" fitness function) and **GERM$_e$** ("edge" fitness function). In the experiment, for genetic activation, the population size was 10, the crossover rate was 1.0, the mutation rate was 0.1. We applied sensitivity analysis for these genetic-related parameters in section 4.2.

4.2 Experiment Results and Analyses

Task Performances. The three task performances are summarized in Table 4. Based on the experiment results, we have the following observations and discussions:

• **GERM vs. Baselines.** GERM had modest improvements comparing with the other baselines for almost all the evaluation metrics for all tasks. These results indicate that: (1) GERM is able to generate high-quality vertex representations for complex heterogeneous graphs; (2) GERM can achieve a good task-oriented multiplex learning performance for various ranking tasks.

• **GERM vs. N2V & DW.** Although designed for homogeneous graphs, by integrating multiple edges with their weights between two vertexes, the random walk based baselines N2V and DW can achieve good performance (best-performed and runner-up baselines). However, in this study, we focus on not only the performance but also efficiency (running time). The theoretical time complexity of DW and N2V is $O(\log|V|)$. Correspondingly, the time complexity of GERM is $O(\log(|V_{A_E}|))$. As the number of activated vertexes decreases, the computational cost can be significantly reduced. Experiment results confirmed this assumption. Empirically, the **computational costs** of N2V and DW were quite expensive, as shown in Figure 3 (a). For instance, in feeds recommendation, GERM$_e$ only used 9.1% of the running time while the performance had a 2.5% improvement (Map@10, NDCG@10, P@10, and MRR), comparing to N2V. More detailed efficiency analysis see Figure 3.

• **Heterogeneous Graph Embedding Baselines**. (1) M2V tried to address the task-oriented multiplex problem by using the human-defined metapath. However, a single metapath cannot completely cover the semantic and structural correlations between different types of vertexes. Meanwhile, human knowledge is not always available or reliable, especially for a complex heterogeneous graph. Therefore, M2V didn't achieve a significant performance in the experiment. (2) HEER employed the edge representations to enhance the heterogeneous graph embedding. From a task-oriented perspective, although HEER considered the edge type incompatibility, it cannot eliminate task-irrelevant noisy information. This may be the reason why HEER didn't perform well and stably in the experiment.

• **GCN Performance**. Deep neural network model GCN had a poor performance in the experiment. There may be two possible reasons. First, GCN is designed for an end-to-end fashion. Though we train the model following the original paper [19], the graph embedding information from the representation layer can only exhibit partial capabilities of this model. We consider it's a limitation of GCN: it required a specific model design for different tasks, and its graph embedding layer information can't be directly integrated into other models (it needs to be re-trained and re-tuned). Correspondingly, GERM is designed for graph embedding, and its genetic edge-type activation strategy is flexible for any downstream vertex ranking task. Second, though we have constructed a homogeneous graph based on the heterogeneous one, the heterogeneity problem may still decrease GCN's representation ability. Meanwhile, as a successful extension of GCN, R-GCN [29] can achieve an improvement for the multi-relational data. We used the original GCN in the experiment because of its efficiency. It's worthwhile to compare GERM with R-GCN. We left it as an impotent future work.

• **LINE Performance**. Local proximity preserving model LINE didn't perform well in the experiment either. A possible explanation is that LINE only uses first-order and second-order neighborhood information for embedding, which is not enough for a complex heterogeneous graph. This may also explain why random walk based models like GERM, DW and N2V can achieve relatively good performance, because they have a higher order neighborhood information.

• **Variants of GERM**. To compare the variants of the proposed method: (1) GERM$_g$ and GERM$_r$ can maintain a good and stable performance in all tasks. Correspondingly, GERM$_e$ didn't have steady performance. It performed poorly in citation recommendation and coauthor recommendation tasks (on the scholarly graph), while achieved a good performance in feeds recommendation task (on the e-commerce graph). These results reveal that the genetic activation methods (with different fitness functions) are closely related to the quality of the final graph embedding. (2) Furthermore, as Table 5 shows, compared to the other two functions, "edge" fitness evaluation can be relatively "aggressive" for heterogeneous graph deactivation. For instance, in the scholarly graph, it deactivated 90.3% edges for citation recommendation task and 79.5% for coauthor recommendation task. Meanwhile, as Table 3 shows, the scholarly graph only has 10 edge-types and 3 vertex-types (relatively simple) while the e-commerce graph has 22 edge-types and 8 vertex-types (relatively complex). It is reasonable to assume that for a relatively simple heterogeneous graph, an aggressive activation function may threaten the quality of the graph embeddings.

Simplification Effect. Table 5 summarizes the comparison of implicitly activated vertexes and edges based on the genetic activation algorithm with different fitness functions. Overall, the proposed algorithm can effectively simplify the heterogeneous graph for different tasks. From the simplification effect viewpoint, "edge" fitness evaluation was most aggressive for deactivation, "greedy" can be the most conservative one, and "refine" was in the middle. As aforementioned in section 3, a successful graph simplification can significantly reduce the computational cost of the graph representation learning.

Empirical Running Time. In the experiment, the best-performed and runner-up (N2V and DW) baselines were both random walk based algorithms, which shared a similar framework with GERM. Figure 3 (a) depicts the running time comparisons[11]. In the experiment, when using proposed GERM models, the running time decreased while the embedding quality was even improved. For instance, in feeds recommendation task, comparing to N2V, GERM$_e$ used only 9.1% running time, and performance (MAP@10) had an

Table 5: Statistics of the implicitly activated graphs using fitness evaluation variants in different tasks. (Task 1: citation recommendation; Task 2: coauthor recommendation; Task 3: feeds recommendation)

Task	Graph	#Vertex	#Edge
Task 1	Original	1,613,872	10,820,534
	greedy	1,613,763 (1×10^{-2}% ↓)	9,945,628 (8.1% ↓)
	refine	1,266,307 (21.5% ↓)	4,532,952 (57.8% ↓)
	edge	1,054,153 (34.7% ↓)	1,053,938 (90.3% ↓)
Task 2	Original	1,613,872	10,735,777
	greedy	1,613,763 (7×10^{-3}% ↓)	5,979,102 (44.3% ↓)
	refine	1,082,922 (32.9% ↓)	6,247,120 (42.3% ↓)
	edge	568,559 (64.8% ↓)	2,205,450 (79.5% ↓)
Task 3	Original	1,007,049	7,288,466
	greedy	824,776 (18.1% ↓)	6,357,547 (12.8% ↓)
	refine	824,917 (18.1% ↓)	5,135,990 (29.5% ↓)
	edge	239,070 (76.3% ↓)	2,554,528 (65.0% ↓)

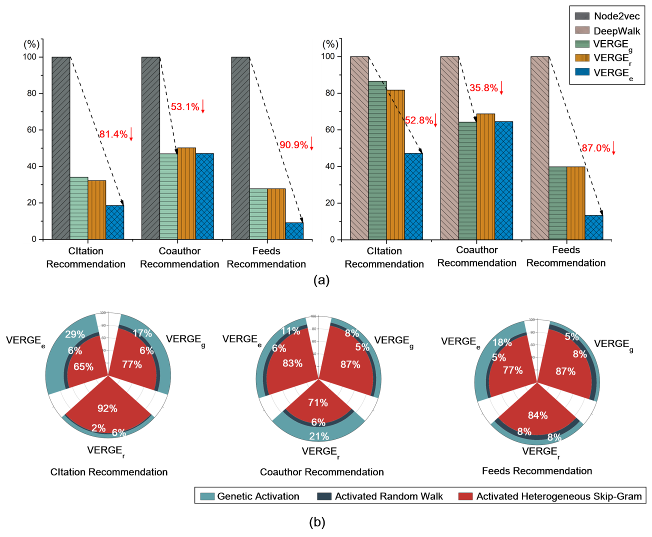

Figure 3: (a) Empirical running time comparison of random walk based algorithms. The left part is N2V vs. GERM variants; The right part is DW vs. GERM variants. For each task, we set the model with longest running time to 1 (100%). (b) Detailed running time (percentage) of each component in GERM variants in tasks. We consider the total running time of each GERM variant as 1 (100%).

[11]For a fair comparison, we ran all the algorithms in the same environment. The compared algorithms were implemented by python. Specifically, N2V and DW were using OpenNE implementation (https://github.com/thunlp/OpenNE). Averaged running time for 5-fold cross-validation was reported.

Table 6: Comparison of models without(w/o) or with(w/) applying Edge-type Activation (EA)[1] as a "plug-in". (Task 1: citation recommendation on academic graph; Task 2: feeds recommendation on e-commerce graph)

Task	Model[2]	Performance (MRR)		Time (s)	
		w/o EA	w/ EA	w/o EA	w/ EA
Task 1	GCN	0.2943	0.3060 (4% ↑)	11,213	3,105 (72% ↓)
	LINE	0.4760	0.6293 (32% ↑)	54,372	14,798 (73% ↓)
	HEER	0.3093	0.3490 (13% ↑)	24,110	6,356 (74% ↓)
Task 2	GCN	0.2677	0.2878 (8% ↑)	3,516	1,097 (69% ↓)
	LINE	0.3607	0.5775 (60% ↑)	7,790	3,140 (60% ↓)
	HEER	0.5709	0.6043 (6% ↑)	19,550	11,226 (43% ↓)

1. In edge-type activation, we use "refine" fitness function for genetic algorithm.
2. All models using exactly same settings in the same running environment.

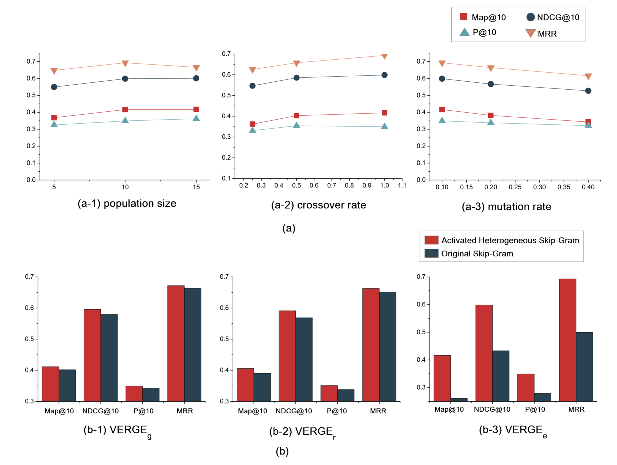

Figure 4: (a) Parameter sensitivity analysis. (b) Activated heterogeneous skip-gram vs. Original skip-gram in GERM variants.

2.5% improvement ($p < 0.001$); while comparing to DW, GERM$_e$ used 13.3% running time, and performance (MAP@10) had a 10% improvement ($p < 0.001$). As N2V required pre-calculation for its 2^{nd} order random walk, its running time was the longest in all random walk based models. Note that, for running time comparison, we didn't count the tuning time of N2V[12].

Among three variants, though GERM$_e$ was the most efficient model, its performance was not stable. Generally speaking, GERM$_g$ and GERM$_r$ can achieve a better balance between efficiency and effectiveness.

In Figure 3 (b), we took a closer look at the running time (percentage) of each component in GERM variants. Comparing to the activated heterogeneous skip-gram (65% - 92% in all tasks), the time-consuming ratio of genetic activation (5% - 29% in all tasks) is not high. Comparing to baselines, *the efficiency improvement of GERM mainly came from the graph simplification.*

[12]For each task, we had to manually tune the p and q of N2V and pick the best-performed settings, the tuning time was much longer than running time.

"Plug-in" Effect. In order to demonstrate that the proposed genetic activation method can preserve the desired edge-types correctly and have a *flexible task-oriented* learning ability. We considered the Edge-type Activation (EA) as a "plug-in" for multiple algorithms (GCN, LINE, and HEER). Note that, in this experiment, we focus on the improvement after applied the EA "plug-in", not the absolute value of each model's performance. For different tasks, the optimized subgraphs (activated by ETAV) were input to graph embedding algorithms. As Table 6 shows, the lifts are very significant. In two tasks (on both graphs), the performances of all algorithms were improved while the running time was significantly reduced. For instance, in citation recommendation task, with EA as a "plug-in", LINE reduced 73% running time, and its performance (MRR) had a 32% improvement; while in feeds recommendation task, EA helped LINE reduced 60% running time, and improved its performance (MRR) by 60%. These results further confirmed our hypothesis that by efficiently learning the task-oriented ETAV, we can improve the quality of the graph embedding to best benefit the different target tasks, while reducing the embedding costs as well.

Parameter Sensitivity Analysis. We also conducted a parameter sensitivity analysis of GERM by tailoring the population size, crossover rate and mutation rate in feeds recommendation task. Figure 4 (a) depicts their impacts on the task performance. Based on the comparison, we find that the proposed method was not very sensitive to these parameters. While a population size of 10, a crossover rate of 1.0, and a mutation rate of 0.1 can achieve optimal performance.

Component Validation. Meanwhile, to validate the effectiveness of the proposed activated heterogeneous skip-gram framework, we also compared the performance of GERM under the original skip-gram framework. As Figure 4 (b) shows, when the activated heterogeneous skip-gram framework was replaced by the ordinary skip-gram, the task performance declined significantly. It is clear that activated heterogeneous skip-gram framework contributes to heterogeneous graph representation learning and task performance significantly. This could be one of the reasons why GERM can outperform the other random walk based algorithms, i.e., N2V and DW.

Activation Visualization. To gain a deeper understanding of the characteristics of the proposed genetic activation algorithm, we conduct an in-depth case study in feeds recommendation task. As shown in Figure 5 (a), we visualized the optimal ETAV in each iteration of the genetic activation algorithm. Figure 5 (b) further depicts the change of activated edge-type number with evolving. It is clear that the evolution of the activated edge-type number was showing a downward trend and eventually stabilized at a relatively small level (from 22 to 10). For the feeds recommendation task, in optimized ETAV, the task-irrelevant edge-types were deactivated and task-relevant edge-types were successfully activated. For instance, edge-type 2 (item $\overset{belong}{\longrightarrow}$ category) and edge-type 16 (user $\overset{dislike}{\longrightarrow}$ tag) were removed, while edge-type 11 (user $\overset{collect}{\longrightarrow}$ feed) and edge-type 19 (user $\overset{buy}{\longrightarrow}$ item) were kept. The activated/deactivated edge-types can enhance the interpretability of the model, helping us to better understand the correlation between the graph structure and a specific task.

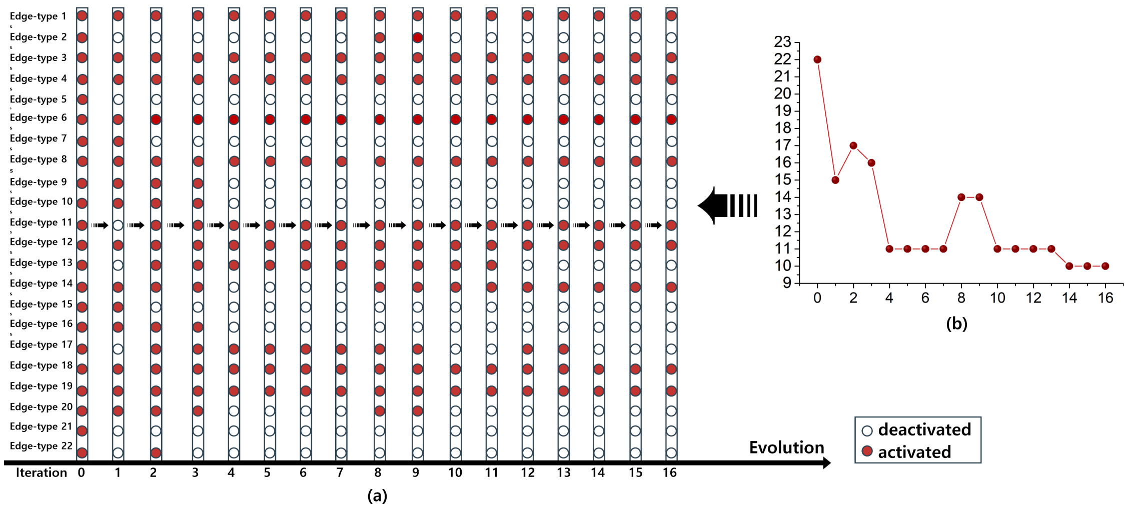

Figure 5: (a) Visualization of the best ETAV in each iteration of the proposed genetic activation algorithm (with "edge" fitness evaluation) in feeds recommendation task. (b) The activated edge-type number of best ETAV in each iteration. The y-axis is the activated edge-type number, whereas the x-axis is the evolution iterations.

5 RELATED WORK

Graph embedding [8] is attracting increasing attention recently due to its broad applicability. Prior works mainly focused on graph factorization, e.g., GraRep [9]. However, the scalability of this kind model could be a potential problem. For instance, the space complexity of GraRep is $O(|V|^2)$ and the memory requirement for embedding a large scale graph will be unacceptable.

Following the idea of representation learning and the success of word embedding [25], DeepWalk [27], LINE [35], Node2Vec [12], SDNE [39] were proposed for learning representation of vertexes in a graph. Specifically, DeepWalk and Node2Vec are considered as the random walk based models. This kind of model is space-efficient, and could be used in the extremely large-scale graph. For instance, [40] applied a billion-scale commodity embedding using the random walk based graph embedding approach. However, this random walk based model requires time to generate a sufficient number of walks and train the embedding model. This problem could affect the efficiency of this kind of model.

A series of deep neural network based models [19, 38, 44] have been proposed and achieved breakthroughs in recent years. However, most of these models are designed for a specific task, the graph representation learning is the intermediate product, not the target goal. We have to change the neural network structure (especially the output layer) for different tasks.

There are several algorithms target for heterogeneous graph embedding. For instance, metapathvec++ [11] tries to learn the representation of heterogeneous graphs using pre-defined metapaths, which requires human knowledge and could result in excessive simplification of the graph. HEER [32] is a recent heterogeneous graph embedding model which tries to cope with semantic incompatibility of different edge-type. This model is an unsupervised model, hence it cannot be optimized for different tasks.

In this study, we address the task-oriented heterogeneous graph embedding problem and propose a novel method GERM. The proposed model can improve the efficiency of the heterogeneous graph embedding while optimizing the graph structure based on the task information.

6 CONCLUSION

In this paper, we propose GERM for a large-scale complex heterogeneous graph (e.g., $|V| > 10^6$ and $H \geqslant 10$). Unlike existing graph embedding methods, GERM employs a genetic activation algorithm to generate the task-driven ETAV over the edge types in the graph. An activated random walk (on the activated subgraph) and an activated heterogeneous skip-gram framework are proposed based on the learned ETAV.

Extensive experiments via different ranking tasks, on scholarly and e-commerce datasets, prove the efficacy and scalability of the proposed methods. In addition, the learned ETAV is able to reveal the latent task-specified knowledge, which is important to the interpretability of the proposed graph embedding model.

Although the binary edge-type activation is a very efficient way to simplify the large-scale complex heterogeneous graph, a more accurate edge-type probability distribution may further enhance the graph embedding quality. In the future, we will validate GERM with more graphs, and try to explore an efficient task-driven edge-type distribution learning approach to further improve the representation learning performance.

ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China (61876003, 81971691), Guangdong Basic and Applied Basic Research Foundation (2019A1515010837), Fundamental Research Funds for the Central Universities (18lgpy62), and the Opening Project of State Key Laboratory of Digital Publishing Technology (cndplab-2020-Z001).

REFERENCES

[1] Lada A Adamic and Eytan Adar. 2003. Friends and neighbors on the web. *Social networks* 25, 3 (2003), 211–230.

[2] Lada A Adamic and Natalie Glance. 2005. The political blogosphere and the 2004 US election: divided they blog. In *Proceedings of the 3rd international workshop on Link discovery*. ACM, 36–43.

[3] Antonio Albero Ortíz, Juan Monzó Cabrera, Alejandro Benedicto Díaz Morcillo, María Eugenia Requena Pérez, et al. 2006. Combined use of genetic algorithms and gradient descent optimization methods for accurate inverse permittivity measurement. (2006).

[4] Thomas Bäck, David B Fogel, and Zbigniew Michalewicz. 2018. *Evolutionary computation 1: Basic algorithms and operators*. CRC press.

[5] Danail Bonchev and Gregory A Buck. 2005. Quantitative measures of network complexity. In *Complexity in chemistry, biology, and ecology*. Springer, 191–235.

[6] Léon Bottou. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*. Springer, 177–186.

[7] Bobby-Joe Breitkreutz, Chris Stark, Teresa Reguly, Lorrie Boucher, Ashton Breitkreutz, Michael Livstone, Rose Oughtred, Daniel H Lackner, Jürg Bähler, Valerie Wood, et al. 2007. The BioGRID interaction database: 2008 update. *Nucleic acids research* 36, suppl_1 (2007), D637–D640.

[8] Hongyun Cai, Vincent W Zheng, and Kevin Chang. 2018. A comprehensive survey of graph embedding: problems, techniques and applications. *IEEE Transactions on Knowledge and Data Engineering* (2018).

[9] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 891–900.

[10] Haochen Chen, Bryan Perozzi, Yifan Hu, and Steven Skiena. 2018. HARP: Hierarchical Representation Learning for Networks. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*. AAAI Press.

[11] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 135–144.

[12] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 855–864.

[13] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. 2016. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, 55–64.

[14] Zhuoren Jiang, Liangcai Gao, Ke Yuan, Zheng Gao, Zhi Tang, and Xiaozhong Liu. 2018. Mathematics content understanding for cyberlearning via formula evolution map. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 37–46.

[15] Zhuoren Jiang, Zhe Gao, Guoxiu He, Yangyang Kang, Changlong Sun, Qiong Zhang, Luo Si, and Xiaozhong Liu. 2019. Detect Camouflaged Spam Content via StoneSkipping: Graph and Text Joint Embedding for Chinese Character Variation Representation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 6188–6197.

[16] Zhuoren Jiang, Xiaozhong Liu, and Liangcai Gao. 2015. Chronological Citation Recommendation with Information-Need Shifting. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 1291–1300.

[17] Zhuoren Jiang, Jian Wang, Lujun Zhao, Changlong Sun, Yao Lu, and Xiaozhong Liu. 2019. Cross-domain Aspect Category Transfer and Detection via Traceable Heterogeneous Graph Representation Learning. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 289–298.

[18] Zhuoren Jiang, Yue Yin, Liangcai Gao, Yao Lu, and Xiaozhong Liu. 2018. Cross-language Citation Recommendation via Hierarchical Representation Learning on Heterogeneous Graph. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM, 635–644.

[19] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*.

[20] Xiaozhong Liu, Zhuoren Jiang, and Liangcai Gao. 2015. Scientific information understanding via open educational resources (OER). In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 645–654.

[21] Xiaozhong Liu, Yingying Yu, Chun Guo, and Yizhou Sun. 2014. Meta-Path-Based Ranking with Pseudo Relevance Feedback on Heterogeneous Graph for Citation Recommendation. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. ACM, 121–130.

[22] Xiaozhong Liu, Yingying Yu, Zhuoren Jiang, Chun Guo, and Scott Jensen. 2017. Personalized Navigation and Random Walk on a Complex Heterogeneous Graph. In *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval*. ACM, 217–224.

[23] Qing Lu and Lise Getoor. 2003. Link-based classification. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*. 496–503.

[24] Matt Mahoney. 2011. Large text compression benchmark. *URL: http://www. mattmahoney. net/text/text. html* (2011).

[25] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[26] Mark EJ Newman. 2001. Clustering and preferential attachment in growing networks. *Physical review E* 64, 2 (2001), 025102.

[27] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 701–710.

[28] Kenneth H Rosen and Kamala Krithivasan. 2012. *Discrete mathematics and its applications: with combinatorics and graph theory*. Tata McGraw-Hill Education.

[29] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*. Springer, 593–607.

[30] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine* 29, 3 (2008), 93.

[31] Chuan Shi, Yitong Li, Jiawei Zhang, Yizhou Sun, and S Yu Philip. 2017. A survey of heterogeneous information network analysis. *IEEE Transactions on Knowledge and Data Engineering* 29, 1 (2017), 17–37.

[32] Yu Shi, Qi Zhu, Fang Guo, Chao Zhang, and Jiawei Han. 2018. Easing Embedding Learning by Comprehensive Transcription of Heterogeneous Information Networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2190–2199.

[33] Mandavilli Srinivas and Lalit M Patnaik. 1994. Genetic algorithms: A survey. *computer* 27, 6 (1994), 17–26.

[34] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S Yu, and Tianyi Wu. 2011. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *Proceedings of the VLDB Endowment* 4, 11 (2011), 992–1003.

[35] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 1067–1077.

[36] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. 2008. Arnetminer: extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 990–998.

[37] Amanda L Traud, Peter J Mucha, and Mason A Porter. 2012. Social structure of Facebook networks. *Physica A: Statistical Mechanics and its Applications* 391, 16 (2012), 4165–4180.

[38] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

[39] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1225–1234.

[40] Jizhe Wang, Pipei Huang, Huan Zhao, Zhibo Zhang, Binqiang Zhao, and Dik Lun Lee. 2018. Billion-scale Commodity Embedding for E-commerce Recommendation in Alibaba. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 839–848.

[41] Reza Zafarani and Huan Liu. 2009. Social computing data repository at ASU.

[42] Daokun Zhang, Jie Yin, Xingquan Zhu, and Chengqi Zhang. 2018. Network representation learning: a survey. *IEEE Transactions on Big Data* (2018).

[43] Rong Zhu, Kun Zhao, Hongxia Yang, Wei Lin, Chang Zhou, Baole Ai, Yong Li, and Jingren Zhou. 2019. AliGraph: A Comprehensive Graph Neural Network Platform. In *45th International Conference on Very Large Data Bases (VLDB)*.

[44] Chenyi Zhuang and Qiang Ma. 2018. Dual Graph Convolutional Networks for Graph-Based Semi-Supervised Classification. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 499–508.