Pattern Recognition 164 (2025) 111495

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Extremely fast graph integration for semi-supervised learning via Gaussian fields with Neumann approximation

Taehwan Yun [a], Myung Jun Kim [c], Hyunjung Shin [a,b,*]

[a] Department of Artificial Intelligence, Ajou University, 206, World cup-ro, Yeongtong-gu, Suwon-si, 16499, Gyeonggi-do, Republic of Korea
[b] Department of Industrial Engineering, Ajou University, 206, World cup-ro, Yeongtong-gu, Suwon-si, 16499, Gyeonggi-do, Republic of Korea
[c] Soda, INRIA Saclay, 1 Rue Honoré d'Estienne d'Orves, Palaiseau, 91120, Île-de-France, France

ARTICLE INFO

Keywords:
Graph-based semi-supervised learning
Multiple graphs
Graph integration
Graph fusion
Neumann series

ABSTRACT

With the rapid growth in data availability, it has become more important to utilize multiple data sources containing different but complementary information for a given task. Using multiple graphs can technically be interpreted as finding the optimal combination of each graph. There have been various approaches for graph integration or graph fusion, but most of them have suffered from scalability issues as data size increases due to long computation time. This makes them difficult to utilize in the current trend of data size becoming huge. To circumvent this difficulty, our approach introduces a fast graph integration method based on semi-supervised learning (SSL), which incorporates the Neumann approximation during the maximum likelihood estimation process. Empirical studies show that the proposed method significantly reduces computation time by at least a factor of two compared to state-of-the-art methods, while still performing competitively with other methods. This advantage becomes more apparent as the size of the data increases, since the complexity of the proposed method depends mostly on the number of graphs to be integrated and not on the number of nodes, unlike other methods. Experimental results demonstrate the scalability and efficiency of the proposed method for graph integration.

1. Introduction

In many real-world situations, data emerges from diverse, heterogeneous sources. For instance, within social networks, the nature of an edge is defined by various types of interactions such as friendships, expressions of emotion, comments on shared posts, or even 'likes' between users as in [1]. In computational biology, the functions of genes or proteins are elucidated through various lenses, like genetic interactions or joint participation in protein complexes, leading to the creation of independent graphs. Each graph source potentially holds unique, complementary information, and making the combination of available graphs beneficial for comprehensive analyses and improving performance in specific tasks as shown in [2]. This process, known as graph integration or graph fusion, aims to amalgamate these diverse data sources.

The most intuitive way is to estimate a linear combination of graphs [3,4]. In previous works, various approaches have been proposed to estimate the optimal linear combination. Among these, the method named multiple kernel learning (MKL) has mainly been studied in conjunction with support vector machines (SVM) or graph-based semi-supervised learning (SSL) [5–8]. In [9], semi-definite programming for multiple graph kernels was proposed to integrate heterogeneous sources of data within the SVM framework. In [10,11], the Lagrange multiplier for the dual problem of the graph-based regularization was used to obtain the optimal combining coefficients. On the other hand, works of [12] attempted to find the combination by minimizing the least squares error between the integrated graph and the target graph. In [13], expectation–maximization (EM) style algorithm was proposed to estimate the optimal values for the combining coefficients. There are also ways to perform graph integration non-linearly. In [14,15], the graphs were integrated using the diffusion process. In [16,17], a fused graph based on spectral decomposition was proposed. Recently, there has been a growing trend of using graph neural networks (GNNs) to graph integration [18,19]. In [20,21], they proposed a GNN model that spans the number of edge features as many as the number of data sources. And in [22–25], a peculiar GNN model was proposed that can be applied to tie together heterogeneous types of graphs, such as movie, actor, and director networks.

* Corresponding author at: Department of Industrial Engineering, Ajou University, 206, World cup-ro, Yeongtong-gu, Suwon-si, 16499, Gyeonggi-do, Republic of Korea.
E-mail addresses: youndh0101@ajou.ac.kr (T. Yun), myung.kim@inria.fr (M.J. Kim), shin@ajou.ac.kr (H. Shin).
URL: https://alphaminers.ai/ (H. Shin).

https://doi.org/10.1016/j.patcog.2025.111495
Received 23 August 2024; Received in revised form 16 February 2025; Accepted 19 February 2025
Available online 1 March 2025

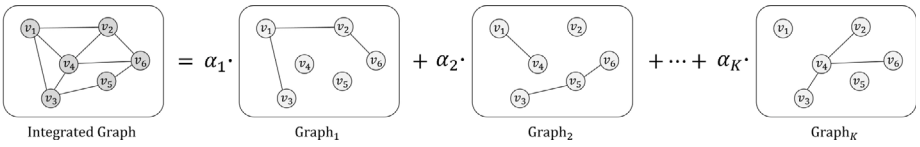


Fig. 1. A schematic overview of graph integration: K graphs, each with different edge sets but sharing the same node set, are combined linearly using the coefficients α's. A single graph may not reliably predict node labels, but multiple graphs from diverse data sources provide both independent and complementary information. By integrating these graphs, predictive performance can be significantly improved, with the key challenge being the determination of optimal combining coefficients α's.

As such, graph integration has been studied in various ways to efficiently utilize multiple data sources. Despite promising results, a major limitation of existing research has been scalability issues due to the enormous computational demands on large data sets. This is typically caused by the need for iterative optimization or computational processes, and the problem becomes more severe as data size increases. This problem can be mitigated if iterative optimization can be avoided.

To address this issue of scalability, we introduce a straightforward and rapid algorithm to approximate the optimal linear combination of multiple graphs. Drawing upon the concepts of graph-based SSL [26], we have reformulated the regularized objective for combining multiple graphs within the context of a Gaussian field framework. More specifically, the SSL integration for multiple graphs is modeled as a Gaussian distribution with zero mean and a covariance matrix that incorporates the combining coefficients. In this setup, these combining coefficients are estimated through maximum likelihood estimation (MLE), using the initially given labels. Essentially, this process identifies the most probable values for the coefficients in a manner that best fits the available labeled data. During the MLE, we utilize the Neumann approximation to simplify the process, allowing for a closed-form solution.

Our experimental results demonstrate the effectiveness of this proposed method, highlighting its strengths in both performance and scalability. The proposed method not only improves computational efficiency but also maintains, and in some cases enhances, the accuracy of the integration. Using the Neumann approximation instead of iterative optimization provides significant gains in computational speed without sacrificing much accuracy. This is particularly valuable for applications requiring real-time or near-real-time processing. Experimental comparisons show that our method is at least two orders of magnitude faster than the best existing methods, suggesting it could serve as a highly efficient and practical alternative for large-scale graph integration tasks.

2. Gaussian fields for graph-based semi-supervised learning

In this section, we reinterpret the conventional graph-based SSL as an energy function within a Gaussian field framework. Our approach begins with a foundational case using just a single graph and evolves to incorporate the integration of multiple graphs. And it conforms to the approach, which is represented by a linear combination of graphs, as illustrated in Fig. 1.

Graph-based SSL is a powerful algorithm that utilizes the structural information of the graph to predict the values of unlabeled nodes with only a small number of labeled nodes [27–29]. A dataset can be represented by a graph $G(V,W)$ composed of the node set $V=\{v_1,v_2,\dots,v_n\}$ and the edge set $W\in\mathbb{R}^{n\times n}$ in which the elements w_{ij} represent the strength of connection between node i and j. For the problem of semi-supervised learning framework, each node is divided into labeled node $\{(v_1,y_1),(v_2,y_2),\dots,(v_l,y_l)\}$ and unlabeled node $\{v_{l+1},v_{l+2},\dots,v_{l+u=n}\}$, and usually $u\gg l$. Given a graph $G(V,W)$ with binary labels $y_i\in\{-1,+1\}$, node classification is performed by minimizing the following energy function [30,31]

$$\min_f\ E(f)=\alpha\sum_{i,j}w_{ij}(f_i-f_j)^2+\sum_{i\in I}(f_i-y_i)^2. \tag{1}$$

The predicted value $f_i\in\mathbb{R}$ of each node i is treated as real number. The parameter α is for the trade-off between the two terms, and I is an index set of labeled nodes. The first term in (1) is the *smoothness* that enforces similar data points to have more similar outputs, and the second term is the *loss* for consistency with labeled data. Now we interpret (1) as an energy function, in accordance with the Gaussian random fields outlined in [27,32]. Firstly, given that $f_i\equiv y_i$ for every labeled node i, the *loss* term is no longer present. Consequently, (1) can be simplified as follows

$$E(f)=\alpha f^\top Lf \tag{2}$$

where $L=D-W$ represents the graph Laplacian, with $D=diag(d_i)$ where $d_i=\sum_j w_{ij}$ [33]. Although (2) might not qualify as a proper energy function due to that the Laplacian L includes at least one zero eigenvalue, a trivial modification can rectify this. The revised energy function can be expressed as

$$E(f)=f^\top(I+\alpha\tilde{L})f, \tag{3}$$

where $\tilde{L}=D^{(-1/2)}LD^{(-1/2)}$ is the normalized graph Laplacian [32]. Now we suppose that the value f is derived from a multivariate Gaussian distribution $f\sim\mathcal{N}(0,C)$ which is

$$p(f)=\frac{1}{(2\pi)^{n/2}\,|C|^{1/2}}\exp(-\frac{1}{2}f^\top C^{-1}f). \tag{4}$$

From this, it becomes apparent that the density function $p(f)$ is in proportion to the energy function,

$$p(f)\propto\exp(-\frac{1}{2}E(f))=\exp(-\frac{1}{2}f^\top(I+\alpha\tilde{L})f). \tag{5}$$

Note that $(I+\alpha\tilde{L})^{-1}$ is interpreted as the covariance C of Gaussian density (4).

Analogously, the process described thus far can be extended to cases for the integration of multiple graphs. Consider that there are K graphs G_k $(k=1,\dots,K)$, each sharing the same set of nodes V, but differing in their similarity matrices $W_k\in\mathbb{R}^{n\times n}$.

$$\{G_1,G_2,\dots,G_K\},\quad G_k=G(V,W_k),$$

Similar to the case of a single graph, the conventional SSL formulation for integrating multiple graphs [2,10]

$$\min_f\quad(f-y)^\top(f-y)+\sum_{k=1}^{K}\alpha_k f^\top\tilde{L}_k f, \tag{6}$$

can be developed into an energy function using the normalized Laplacians

$$E(f)=f^\top(I+\sum_{k=1}^{K}\alpha_k\tilde{L}_k)f, \tag{7}$$

where $\tilde{L}_k$ denotes the normalized Laplacian of graph k and, α_k plays the role of *smoothness* parameter while simultaneously acting as the combining coefficient for graph k. The only difference between the two energy functions (3) and (7) is that the single Laplacian matrix is replaced by a linear combination of multiple Laplacian matrices. It is important to note that estimating α_k equates to the process of integrating multiple graphs. As a result, the density function $p(f)$ is in proportion to the energy function as follows

$$p(f)\propto\exp(-\frac{1}{2}E(f))=\exp(-\frac{1}{2}f^\top(I+\sum_{k=1}^{K}\alpha_k\tilde{L}_k)f). \tag{8}$$

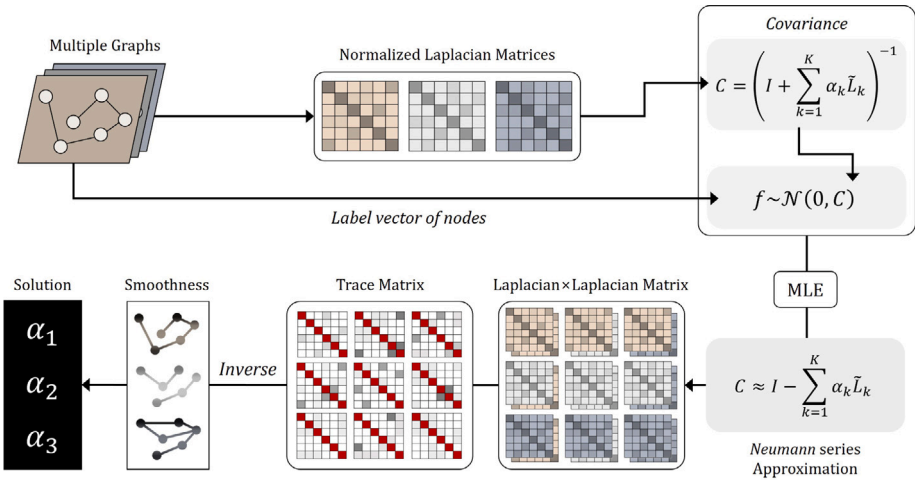


Fig. 2. Overall process of the proposed method. The figure presents the case where there are three graphs is taken as an example. The proposed framework first constructs the covariance matrix C as a linear combination of the graph Laplacians. With the covariance, maximum likelihood estimation is employed along with the Neumann expansion of C for approximation. The closed-form solution of α is obtained by solving the system of linear equation.

The density function (8) is a conversion of the energy function (7) into Gaussian field form. Within this Gaussian field framework, the SSL integration for multiple graphs is characterized as a Gaussian distribution with a mean of 0 and a covariance matrix denoted by

$$C = (I + \sum_{k=1}^{K} \alpha_k \tilde{L}_k)^{-1}. \tag{9}$$

With this setup, we start with known f_i values for labeled nodes and estimate the parameter α_k via maximum likelihood estimation. This means finding the most likely values for the α_k's, which are crucial for integrating multiple graphs, in a way that best fits the available labeled data.

3. MLE with Neumann approximation for graph integration

3.1. Maximum likelihood estimation

The process of maximum likelihood estimation for α involves minimizing the negative log-likelihood of the Gaussian distribution $\mathcal{N}(0, C)$ in (4),

$$\text{NLL}(\alpha) = \frac{1}{2} f^\top C^{-1} f + \frac{1}{2} \log |C| + \frac{n}{2} \log 2\pi. \tag{10}$$

Note that α_k is included as in C(9). To delve deeper into how α_k affects C, we look at the partial derivative of C with respect to α_k, as given by

$$\frac{\partial C}{\partial \alpha_k} = -C L_k C, \quad k = 1, \dots, K. \tag{11}$$

From this, the partial derivative of (10) with respect to each α_k is derived as follows

$$\begin{aligned} \frac{\partial \text{NLL}}{\partial \alpha_k} &= -\frac{1}{2} f^\top C^{-1} \frac{\partial C}{\partial \alpha_k} C^{-1} f + \frac{1}{2} \text{tr}\left(C^{-1} \frac{\partial C}{\partial \alpha_k}\right) \\ &= \frac{1}{2} f^\top \tilde{L}_k f - \frac{1}{2} \text{tr}(\tilde{L}_k C), \quad k = 1, \dots, K. \end{aligned} \tag{12}$$

However, it is not practical to obtain a closed-form solution of (12) because α_k is embedded in the covariance matrix C, which requires an inversion operation. This complication necessitates the use of gradient-based optimization methods. Unfortunately, the computational demand is quite high, primarily because it involves inverting a matrix, a task that becomes particularly challenging when dealing with very large matrices. In situations where obtaining an exact solution is complex and computationally demanding, an alternative strategy is to use an approximation, which can simplify the process and reduce the computational load, offering a viable solution.

3.2. Approximation

From (12), we can replace the covariance matrix C with the first-order approximation of $(I + \alpha\tilde{L})^{-1}$ using the Neumann expansion. The Neumann series for a matrix S is defined as follows [34]:

$$(I + S)^{-1} = \sum_{m=0}^{\infty} (-1)^m S^m,$$

where $S^m \to 0$ for $m \to \infty$. Since $\tilde{L}_i^m \to 0$ for $m \to \infty$ and by setting $0 < \alpha_i < 1$ with $\sum_{k=1}^{K} \alpha_k \le 1$, we can approximate C as the following:

$$C = \left(I + \sum_{k=1}^{K} \alpha_k \tilde{L}_k\right)^{-1} \approx I - \sum_{k=1}^{K} \alpha_k \tilde{L}_k. \tag{13}$$

Using this approximation, (12) can be rephrased as:

$$\begin{aligned} \frac{\partial \text{NLL}}{\partial \alpha_k} &= \frac{1}{2} f^\top \tilde{L}_k f - \frac{1}{2} \text{tr}\left(\tilde{L}_k \left(I - \sum_{i=1}^{K} \alpha_i \tilde{L}_i\right)\right) \\ &= \frac{1}{2} f^\top \tilde{L}_k f - \frac{n}{2} - \frac{1}{2} \text{tr}\left(\tilde{L}_k \sum_{i=1}^{K} \alpha_i \tilde{L}_i\right), \end{aligned} \tag{14}$$

which leads to a system of equations obtained by setting the partial derivative of the NLL to zero:

$$\begin{bmatrix} \text{tr}(\tilde{L}_1\tilde{L}_1) & \cdots & \text{tr}(\tilde{L}_1\tilde{L}_K) \\ \text{tr}(\tilde{L}_2\tilde{L}_1) & \cdots & \text{tr}(\tilde{L}_2\tilde{L}_K) \\ \vdots & \ddots & \vdots \\ \text{tr}(\tilde{L}_K\tilde{L}_1) & \cdots & \text{tr}(\tilde{L}_K\tilde{L}_K) \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_K \end{bmatrix} = \begin{bmatrix} n - f^\top \tilde{L}_1 f \\ n - f^\top \tilde{L}_2 f \\ \vdots \\ n - f^\top \tilde{L}_K f \end{bmatrix} \tag{15}$$

Based on this, we are able to derive the optimal solution for the combining coefficient, denoted as α^*, in a closed form. For better understanding of the proposed framework, Fig. 2 summarizes the overall process. Subsequently, we can also compute the predicted values f in (6) using this optimal α^*. The problem being convex allows for a straightforward computation of the optimal solution for f, which is depicted in the below:

$$f^* = \left(I + \sum_{k=1}^{K} \alpha_k^* \tilde{L}_k\right)^{-1} y. \tag{16}$$

Integrating multiple graphs using the optimally determined coefficients α^* enables a more precise calculation of f, effectively incorporating diverse information from multiple sources.

3.3. Remarks on the Gaussian field formulation of SSL

When $f_i \equiv y_i$ for every labeled node i and the remaining nodes are unlabeled $y \in \{0\}$, the predicted values f_i's in SSL tend to cluster around zero. This behavior suggests that f_i follows a Gaussian distribution with a mean of zero and an unknown covariance. The covariance of the Gaussian random field provides valuable predictive information, analogous to the role of the Laplacian in SSL. Through the energy function framework [27,32], it can be shown that the inverse of $(I + \alpha L)$ corresponds to the covariance C. This relationship highlights that a more effective Laplacian results in a smaller covariance, leading to more confident predictions.

This observation motivates the reformulation of SSL as a Gaussian field. Such a reformulation brings additional advantages, including natural parameter estimation via MLE. In a single-graph setting, the smoothness parameter α in SSL can be estimated using MLE. Similarly, in a multi-graph setting, the integration coefficients for multiple graphs α_k's $(k = 1, \ldots, K)$ can also be determined through MLE, where a larger value of α_k indicates that the corresponding graph contributes more to the prediction.

3.4. Remarks on complexity

The use of multiple graphs brings into focus the aspect of computational complexity. Typically in graph integration applications, the number of graphs (K) is much smaller than the number of nodes $(n)(K \ll n)$. Consequently, inverting the $K \times K$ matrix in (15), is relatively straightforward and not computationally intensive. However, the primary computational cost arises from the operation $\mathrm{tr}(\tilde{L}_i \tilde{L}_j)$, which depends on the number of non-zero edges in both G_i and G_j. When dealing with two graphs, the cases of having non-zero entries in both simultaneously are generally rare. Consequently, the computational complexity involved in calculating the coefficients α, is $O(K^3)$, which is quite manageable in most cases. On the other hand, for solving f, we address a sparse linear system given by $y = \left(I + \sum_{k=1}^{K} \alpha_k^* \tilde{L}_k\right) f$, rather than using (16) which involves inverting an $n \times n$ matrix. This approach significantly reduces computational demands, and is nearly linear to the number of non-zero entries in $\sum_{k=1}^{K} \alpha_k^* \tilde{L}_k$ [35].

4. Experiments

For validation of the proposed method, we evaluated the classification performance on nine datasets by measuring the area under the receiver operating characteristic curve (AUC) and the computation time. Prior to comparison with other algorithms, we tested our approximate solution against a single graph and the exact solution denoted by I-ACT (Integration with exACT solution) was obtained through gradient-based optimization to minimize the NLL for multiple graphs. For a single graph, we recorded results for one graph with the best performance. Also, we observed the changes in computation time of the proposed method on a randomly generated toy dataset to test the dependence on the number of graphs and number of nodes. For comparative validation, we considered the following major integration algorithms

- *Fast Integration for Multiple Networks (FIMN)*: Our method is fundamentally based on the work of FIMN [10]. This method employs convex optimization for the estimation of the optimal combining coefficient.
- *Robust Label Propagation on Multiple Networks (RLPMN)*: The met hod is based on the EM algorithm, and determines the combining coefficients converged through iterative updates [13].
- *Gene Multiple Association Network Integration Algorithm (GMANIA)*: This method estimates the optimal combining coefficients by minimizing the gap between the integrated graph and the target graph constructed with label vector [12].
- *Gaussian Process Classification (GPC)*: In a similar way to our method, a sheer work of Gaussian Process (GP) [36] to graph integration was considered for comparison.
- *Affinity Network Fusion (ANF)*: ANF [15] uses an r-step random walk ($r = 2$ in the typical case) to fuse multiple graphs into a linear combination of their respective affinity matrices after smoothing them. The first step is a random walk on a particular graph, and the second step is a random walk on the aggregated complementary graph, and so on.
- *Multiplex Heterogeneous Graph Convolutional Network (MHGCN)*: The MHGCN [22] is a deep learning model designed to capture meta-path information across multi-relations in heterogeneous graphs using a multi-layer graph convolution module.

A more detailed description of each algorithm is provided in Appendix. In the following subsections, we present brief summary of data, experimental settings and detailed discussions of the results.

4.1. Datasets

We used a total of nine datasets for our experiments. Table 1 is summary of the datasets used in the experiment, respectively. In this table, the first five datasets are provided as multiple networks. And the remaining four datasets are given in tabular form.

4.1.1. Network datasets

The followings are brief descriptions for the five network datasets.

- ***MIPS*** is protein network dataset for predicting functional class of yeast proteins. The function of each protein is labeled according to the MIPS Comprehensive Yeast Genome Database [10]. In out dataset, Each protein may belong to 13 functional classes, and we performed experiments on 11 of these classes. This dataset has five types of edge (W_{ST}, W_{BP}, W_{PPI}, W_{GI}, and W_{GE}) derived from five different measures.
- ***Cora*** is a citation network for scientific publication classification task. Nodes in the graph represent publications and each publication can belong to only one of the seven categories. The original dataset has only one type of edge (W_{CIT}), but we make an additional type (W_{SWV}) with the node feature vector using the cosine similarity.
- ***IMDB*** contains information about movies, such as genres, actors, and directors. We created two types of edge from this dataset. (1) W_{MDM}: connect movies made by the same director; (2) W_{MAM}: connect movies featuring at least one same actor. Each movie can belong to one of four genres (*Action*, *Comedy*, *Drama*, *Thriller*) [23].
- ***Amazon*** is a benchmark graph for fraudulent user detection task. Nodes in the graph represent users and there are two types of edge useful for classification. (1) W_{USV}: connect users having at least one same star rating within one week; (2) W_{UVU}: connect users with top 5% text similarities (measured by TF-IDF) among all users [37].
- ***Yelp*** is a benchmark graph for spam review detection task. Nodes in the graph represent reviews written by users. There are three types of edge. (1) W_{RSR}: connect reviews under the same product with the same star rating (1–5 stars); (2) W_{RTR}: connect two reviews under the same product posted in the same month; (3) W_{RUR}: connect reviews posted by the same user [37].

4.1.2. Tabular datasets

Tabular datasets include biological profiles of patients obtained from TCGA (The Cancer Genome Atlas) [14]. The four datasets are information on patients with breast cancer, brain cancer (glioblastoma multiforme), lung cancer, and kidney cancer, respectively. Each dataset provides gene expression and the time period from diagnosis of cancer

Table 1
Summary of nine datasets.

Dataset	# Nodes	# Classes	W	Description	Density (%)
MIPS	3,588	11	W_{ST}	Similarity of protein structure	0.7807
			W_{BP}	Co-participation	0.0570
			W_{PPI}	Protein–protein interactions	0.0565
			W_{GI}	Genetic interactions	0.0435
			W_{GE}	Gene expression measurements	0.0920
Cora	2,708	7	W_{CIT}	Original citation network	0.1809
			W_{SWV}	Similarity of word vector	0.6429
IMDB	4,919	4	W_{MDM}	Same director	0.1081
			W_{MAM}	Same actor	0.4323
Amazon	11,944	2	W_{USV}	Same star rating within one week	5.0088
			W_{UVU}	Similarity of text (TF-IDF)	1.4619
Yelp	45,954	2	W_{RSR}	Same product with same star rating	0.3244
			W_{RTR}	Same product posted in same month	0.0565
			W_{RUR}	Same user	0.0068
Breast	105	2	W_{DNA}	Similarity of DNA methylation	10.5128
			W_{mRNA}	Similarity of mRNA expression	10.3846
			W_{miRNA}	Similarity of miRNA expression	10.7875
			W_{time}	Similarity of survival duration	7.7106
Brain	215	2	W_{DNA}	Similarity of DNA methylation	4.5729
			W_{mRNA}	Similarity of mRNA expression	5.0815
			W_{miRNA}	Similarity of miRNA expression	5.0120
			W_{time}	Similarity of survival duration	3.7383
Lung	106	2	W_{DNA}	Similarity of DNA methylation	10.3324
			W_{mRNA}	Similarity of mRNA expression	9.7754
			W_{miRNA}	Similarity of miRNA expression	10.7278
			W_{time}	Similarity of survival duration	7.5292
Kidney	122	2	W_{DNA}	Similarity of DNA methylation	8.5219
			W_{mRNA}	Similarity of mRNA expression	8.9012
			W_{miRNA}	Similarity of miRNA expression	8.9148
			W_{time}	Similarity of survival duration	6.5980

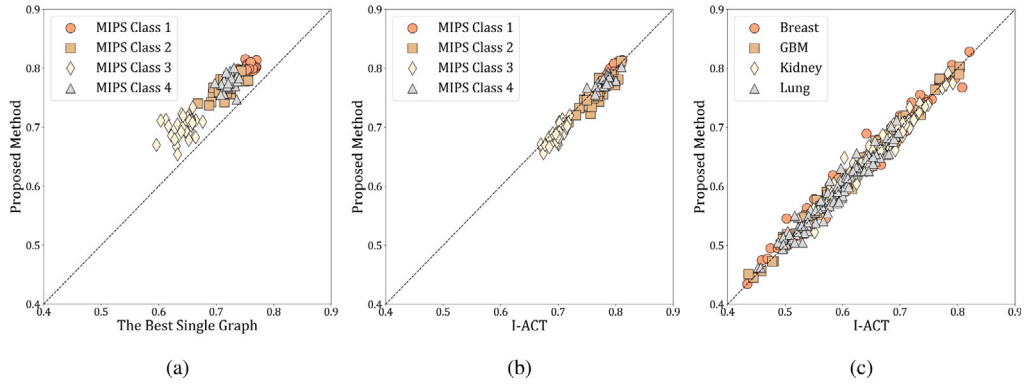


Fig. 3. AUC performance comparison of (a) integrated graph derived from proposed method vs. the best performance of single graph for MIPS (class 1–4), (b) proposed method vs. I-ACT for MIPS, and (c) for four cancer dataset per each repetition. Each marker represents one repetition. (a) and (b) were repeated 30 times per class and (c) repeated 100 times.

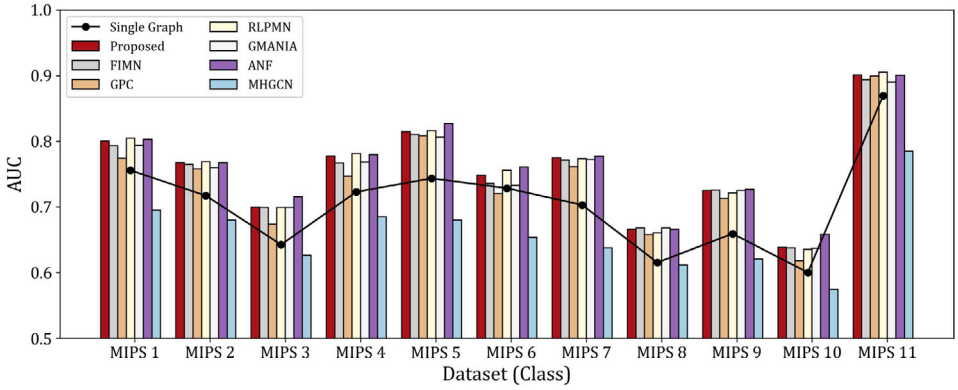
to data collection (or to death for patients who had already died) for each patient. Gene expression is divided into three modalities: DNA, mRNA, and miRNA. We derived four graphs for each dataset using survival duration and three modalities of gene expression. (1) W_{DNA}: similarity of expression for DNA methylation between patients; (2) W_{mRNA}: similarity of mRNA expression between patients; (3) W_{miRNA}: similarity of miRNA expression between patients; (4) W_{time}: similarity of survival duration between patients. We used the Gaussian kernel as a similarity measure and exploited k-nearest neighbor ($k = 5$) which set the similarities between non-neighboring points (in terms of the pairwise similarity) to zero. And the label information is survival or death of the patient at the time of data measurement.

4.2. Experimental setting

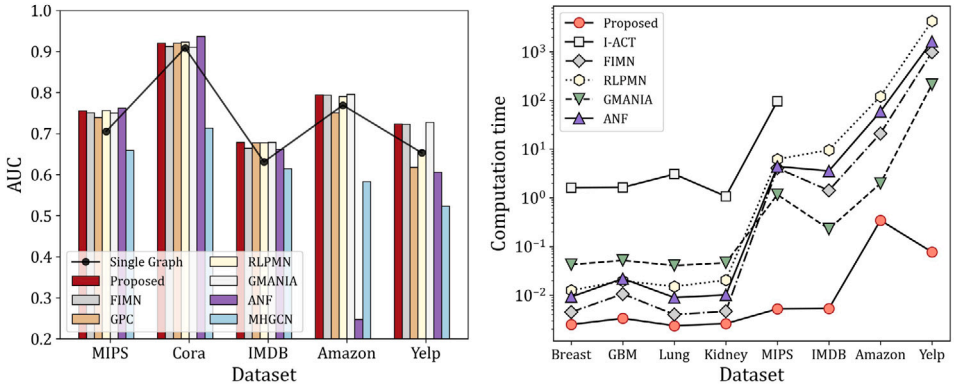
The proposed method was compared with the competing algorithms for node classification tasks. For multi-class problems, we performed binary classification for each of the classes. For the label setting, we randomly assigned 20% of labels to induce the SSL setting. In addition, for speed comparison, the time required for graph integration was measured. Experiment for each dataset was repeated 30 times for network datasets and 100 times for tabular datasets. All experiments were carried out on a AMD Ryzen Threadripper 3960X 24-Core Processor 3.79 GHz PC with 256 GB memory.

4.3. Single vs. Integrated, exact vs. Approximate

Fig. 3 shows the validity of the proposed method in terms of 'single vs. integrated graph' and 'exact vs. approximated integration'. The x-axis represents the AUC performance of the competing methods, and the y-axis represents the AUC performance of the proposed method. The AUC pairs of the competing and the proposed methods are plotted on xy-coordinates. The AUCs were measured at every repetition.



(a)



(b) (c)

Fig. 4. The overall AUC performance of comparing algorithms on five datasets (MIPS, Cora, IMDB, Amazon, Yelp). (a) shows the AUC for each class in the MIPS dataset. The x-axis in (a) represents the dataset and the number corresponding to the class. (b) shows the average AUC for the five datasets. And (c) shows the comparison of average computation time (second) between comparison methods. All graph integration methods have better AUC performance than the best single graph and the proposed method demonstrates competitiveness for the graph integration problem because it achieved similar AUC performance to the comparing methods with the fastest speed.

Fig. 3(a) shows the AUC performance of a best single graph and that of the proposed method for MIPS's protein functional class 1 to 4. In this figure, all points are located on the top of the diagonal line, which means that using the integrated graph is always better than using only a single graph. That is, the information in multiple graphs is more comprehensive than the information in a single graph, which improves performance.

On Figs. 3(b) and 3(c), the pairs of AUCs of I-ACT and the proposed method are scattered, for MIPS's dataset and the four tabular datasets, respectively. In both figures, we observed that the points are located near the diagonal line. And this means that there is little difference in AUC performance between the two methods. These result implies that the proposed method well approximates the exact solution for integration for multiple graphs.

4.4. Comparison with existing methods

The overall AUC performance comparison over the five network datasets is shown in Fig. 4. Fig. 4(a) is the result for each class in MIPS. And Fig. 4(b) is the result of the average AUC for the five datasets. First of all, it can be confirmed that all graph integration methods outperform a single graph in most cases. And it can be seen that the proposed method does not lag behind in terms of performance across the five datasets. Table 2 shows the average of the AUC for each dataset. The proposed method had the highest or second highest performance for all datasets. This result strengthens the competitiveness of the proposed method in problems of graph integration. Existing methods also showed good performance overall, but ANF showed poor performance for large-sized datasets. This is because the r-step random walk was not sufficient for information fusion in large-sized graphs. In addition, MHGCN showed low AUC performance overall due to the absence of node features.

In terms of computation time, Fig. 4(c) shows the computation time (second) of each integration method in log scale for the eight datasets. Note that the computation time of GPC and MHGCN is significantly longer than the others, so it is excluded from the plot, but has a record in Table 3. Among the others, I-ACT consumed the most time, while the proposed method using the Neumann approximation dramatically shortened the time required. We also see that the important factor affecting the computation time of the proposed method is the network density of the involved graphs rather than the number of graphs, since the data sources are usually very small. This can be seen from the fact that the Amazon dataset is smaller than the Yelp dataset but requires more computation time. This issue will be revisited in a later section on empirical time complexity. Overall, the comparison results in performance and computation bolster that the proposed integration framework is competitive against comparing algorithms and has a huge advantage of scalability when time becomes a critical factor. Table 3 summarizes the average computation time on the five network datasets.

The proposed method not only matches but frequently exceeds the performance of existing methods across five datasets. In particular, the use of Neumann approximation dramatically reduces computation time. Unlike previous works such as RLPMN, ANF, and MHGCN, which face scalability challenges with large-sized graphs, our approach excels in both speed and accuracy. This makes it particularly advantageous for large-scale graph integration tasks.

Table 2
Comparison of average AUC with existing methods for five datasets. The experiments were conducted 30 times for each method, and the results are reported as mean ± standard deviation. Statistical significance was assessed using a t-test, with p-values denoted as follows: $*: < 0.01$, and $**: < 0.001$. *n.s.*: not significant.

	MIPS	Cora	IMDB	Amazon	Yelp
GMANIA	0.75 ± 0.02 (*n.s.*)	0.91 ± 0.01 (**)	0.68 ± 0.01 (*n.s.*)	0.80 ± 0.01 (*n.s.*)	0.73 ± 0.01 (*n.s.*)
RLPMN	0.76 ± 0.01 (*n.s.*)	0.92 ± 0.01 (*n.s.*)	0.68 ± 0.01 (*n.s.*)	0.79 ± 0.01 (*n.s.*)	0.66 ± 0.00 (**)
GPC	0.74 ± 0.02 (*)	0.92 ± 0.01 (*n.s.*)	0.68 ± 0.01 (*n.s.*)	0.75 ± 0.00 (**)	0.62 ± 0.05 (**)
FIMN	0.75 ± 0.02 (*n.s.*)	0.91 ± 0.01 (*)	0.66 ± 0.01 (*n.s.*)	0.79 ± 0.01 (*n.s.*)	0.72 ± 0.00 (*n.s.*)
ANF	0.76 ± 0.01 (*n.s.*)	0.93 ± 0.01 (**)	0.66 ± 0.01 (*)	0.25 ± 0.00 (**)	0.61 ± 0.00 (**)
MHGCN	0.66 ± 0.01 (**)	0.71 ± 0.03 (**)	0.61 ± 0.01 (**)	0.58 ± 0.09 (**)	0.52 ± 0.03 (**)
Proposed	0.76 ± 0.02	0.92 ± 0.01	0.68 ± 0.01	0.79 ± 0.01	0.72 ± 0.00

Table 3
Comparison of computation time (*s*) with existing methods for five datasets. We repeated the experiment 30 times and reported mean ± standard deviation. It was confirmed that our proposed method showed the fastest speed.

	MIPS	Cora	IMDB	Amazon	Yelp
GMANIA	1.1548 ± 0.0947	0.1802 ± 0.1646	0.2271 ± 0.0109	1.9630 ± 1.2117	212.8583 ± 251.5614
RLPMN	6.1869 ± 0.2415	2.8716 ± 0.0294	9.6062 ± 0.0914	121.0524 ± 7.1263	4269.9623 ± 149.2684
GPC	31.1244 ± 15.3637	5.9858 ± 0.7860	8.0670 ± 0.0462	1084.2671 ± 220.7073	16220.4962 ± 4058.6283
FIMN	4.0471 ± 0.7729	0.7448 ± 0.1157	1.4109 ± 0.0343	20.8813 ± 0.9692	982.6038 ± 69.2546
ANF	4.3869 ± 0.0519	0.9083 ± 0.0220	3.5470 ± 0.0506	58.9387 ± 0.2758	1633.0658 ± 8.3567
MHGCN	43.9726 ± 0.2018	32.0222 ± 0.2164	344.3978 ± 3.9224	704.0761 ± 4.5182	8335.2831 ± 45.2805
I-ACT	96.4449 ± 22.6714	–	–	–	–
Proposed	**0.0052 ± 0.0074**	**0.0016 ± 0.0047**	**0.0053 ± 0.0073**	**0.3434 ± 0.0183**	**0.0772 ± 0.0014**

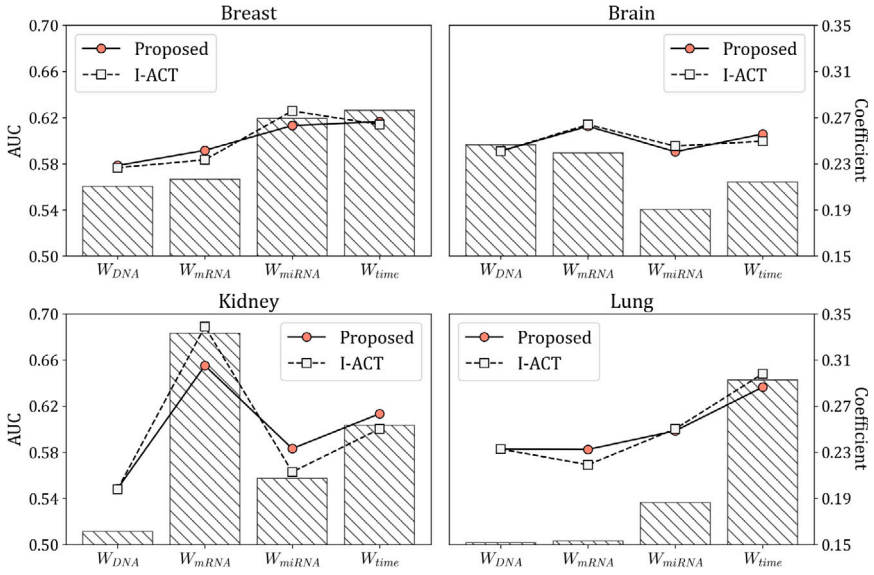


Fig. 5. Comparison with the AUC performance of a single graph and the combining coefficient of an integrated graph for four tabular datasets (Breast, Brain, Kidney, Lung). The bar in the figure is the AUC performance of node classification when only a single graph on the x-axis is used. The line plot is the combining coefficient of the integrated graph derived from the proposed method and I-ACT.

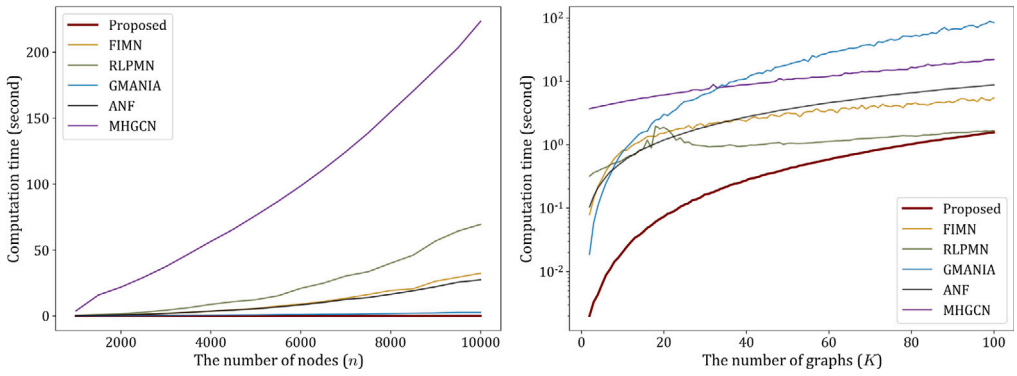


Fig. 6. Comparison of computation time between four graph integration methods according to the number of (a) nodes and (b) graphs.

4.5. Combining-coefficient estimation

Each coefficient of the graph linear combination can be regarded as the importance of each graph. And simply, we can assume that important graph will have high performance of node classification. Fig. 5 shows comparison with the AUC performance of a single graph and the combining coefficient of an integrated graph for Breast, Brain, Kidney, and Lung datasets. In this figure, the coefficient of the integrated graph either by I-ACT or by the proposed method tended to have a high value for a single graph with a high AUC, and vice versa, and the difference between the two methods was small. Therefore, we conclude that the solution of the proposed method approximates the coefficients well.

4.6. Empirical time complexity

To perform an empirical validation of time complexity, toy graph dataset was created using a gaussian kernel and k-nearest neighbor ($k = 3$) from random noise. Labels were also randomly assigned to 20% of the data. Fig. 6 shows the computation time of the six graph integration methods, FIMN, RLPMN, GMANIA, ANF, MHGCN including our method. And Fig. 6(a) represents the computation time according to the increase in the number of nodes (with three graphs, $K = 3$). The computation time of FIMN, RLPMN, ANF and MHGCN increases significantly as the number of nodes increases, whereas the proposed method and GMANIA are not significantly affected. In particular, the proposed method appears to be almost unchanged. With a fixed number of nodes ($n = 1000$), on the other hand, Fig. 6(b) shows the computation time in log scale according to the increase in the number of graphs. Contrary to Fig. 6(a), the computation time of the proposed method and GMANIA significantly increased as the number of graphs increases. The computation time of FIMN, RLPMN, ANF and MHGCN, which are relatively less affected, also significantly increased. The proposed method is the fastest when the number of graphs is less than 100. For the cases where there are more than that (which are very cases in practice), all comparison methods require much longer computation time. Consequently, the empirical study shows that the proposed method is much faster and more robust than other methods in general situations such as ($K \ll n$).

5. Conclusion

As the availability of data increases, it becomes more important to process multiple data sources containing different but complementary information for a given task. In particular, if the data source contains relationship information between data points, representing it as a graph composed of nodes and edges is an inevitable choice. Given multiple graphs, there have been existing approaches combining them, but they have struggled because of the hurdle of long computational time.

To overcome the scalability issue, we proposed an integration method that obtains the optimal values of combining coefficients in a closed-form solution. The proposed method utilizes the maximum likelihood estimation with the first-order approximation on the Neumann expansion of the covariance within a semi-supervised learning framework. Empirically, the proposed method exhibited competitiveness in prediction performance against other algorithms. In computational perspective, the proposed method showed superiority over comparing algorithms, showing its favoring characteristic of fast optimization.

However, there are some limitations, including future works, that need to be addressed. First, a closer look on the approximation should be conducted. In general, the justification of approximation on the Neumann expansion depends on the eigenvalue of S in the Neumann equation. A more rigorous and sophisticated design of approximation should be considered. Furthermore, the proposed integration may be sensitive to network density, which implies it may be vulnerable to fully connected or densely connected graphs, or graphs with many noisy edges. When the densities of the networks differ significantly, the proposed method tends to assign a larger value to a denser network. This is a problem with most methods for integrating multiple networks. However, a network with sparse connections between nodes may still be more valuable than a network with noisy or redundant connections. Deeper investigations into such problems are left for our future study.

CRediT authorship contribution statement

Taehwan Yun: Writing – original draft, Visualization, Validation, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Myung Jun Kim:** Methodology, Data curation, Conceptualization. **Hyunjung Shin:** Writing – review & editing, Writing – original draft, Supervision, Resources, Funding acquisition, Formal analysis.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2021R1A2C2003474), Institute of Information & communications Technology Planning & Evaluation (IITP) under the Artificial Intelligence Convergence Innovation Human Resources Development (IITP-2023-No. RS-2023-00255968) grant funded by the Korea government(MSIT) and the Ajou University research fund.

Appendix. Related works on graph integration

Various approaches have been attempted to estimate parameters for multiple graph integration. Here, some of the main existing approaches are summarized for comparison with ours.

A.1. Fast Integration for Multiple Networks (FIMN)

The work of FIMN [10] is the basis for the proposed method. However, unlike the proposed method that estimates the combining coefficient based on the maximum likelihood estimation, this method uses a convex optimization to estimate the combining coefficient. The formulation begins by constructing the following convex problem

$$\begin{aligned} \min_{f,\xi,\gamma} \quad & (f-y)^\top (f-y) + c\gamma + c_0 \sum_{k=1}^{K} \xi_k \\ \text{s.t.} \quad & f^\top L_k f \le \gamma + \xi_k,\ \xi_k \ge 0,\ \gamma \ge 0. \end{aligned}$$

where c and c_0 are positive constants, $\{\xi_k\}_{k=1}^K$ are slack variables, and γ is the upper bound of the smoothness term $f^T L_k f$. To obtain the optimal coefficients, the work utilize the dual problem which is

$$\begin{aligned} \min_{\alpha} \quad & y^\top \left(I + \sum_{k=1}^{K} \alpha_k L_k \right)^{-1} y \\ \text{s.t.} \quad & 0 \le \alpha_k \le c_0,\ \sum_{k=1}^{K} \alpha_k \le c. \end{aligned}$$

where α_k is the Lagrange multiplier [38] and can be interpreted as combining coefficient of graph k. To estimate the Lagrange multiplier α, the partial derivative with respect to α is used as the gradient. Its computational complexity is $O(n^3)$.

A.2. Robust Label Propagation on Multiple Networks (RLPMN)

RLPMN [13] uses label propagation for graphs, and the framework for graph integration is proposed as

$$\begin{aligned} \min_{f} \quad & \beta_y (f-y)^\top G (f-y) + f^\top (\beta_{bias} I + \beta_{net} L^*) f \\ \text{s.t.} \quad & L^* = \sum_{k=1}^{K} \alpha_k L_k. \end{aligned}$$

where β_y, β_{bias}, β_{net} are constants, and G is the diagonal indicator matrix for the labeled data. And this graph integration method is an expectation maximization [39,40] style iterative algorithm that runs the following two equations until convergence:

$$\hat{f} = \left(G + \frac{\beta_{bias}}{\beta_y} I + \frac{\beta_{net}}{\beta_y} \sum_{k=1}^{K} \alpha_k L_k \right)^{-1} G y \tag{17}$$

$$\alpha_k = \frac{\nu + n}{\nu + \beta_{net} f^\top L_k f} \tag{18}$$

where υ is constants. Observing (17) and (18), this approach also penalizes for large values of smoothness. Also, this method requires an iterative computation process until α converges, and its computational complexity is $O(n^3)$.

A.3. Gene Multiple Association Network Integration Algorithm (GeneMANIA)

GeneMANIA [12] is based on the following linear regression problem

$$\begin{aligned} \min_{\alpha} \quad & \mathrm{tr}((T - W^*)^\top (T - W^*)) \\ \text{s.t.} \quad & W^* = \sum_{k=1}^{K} \alpha_k W_k,\ \alpha_k \ge 0. \end{aligned}$$

where W_k is the similarity matrix of graph k, T_{ij}, elements of the target graph T. The target graph T is a signed graph with edge values dependent on the number of positive and negative labels. By optimizing the above objective function, the integrated graph W^* get closer to the target graph. Its computational complexity is $O(n^2K)$ and it boasts a fast speed in general graph integration settings where the number of graphs is small.

A.4. Gaussian Process Classification (GPC)

Similarly to our method, one may consider a sheer work of Gaussian process (GP) [36] to graph integration. For comparison, the GP-based graph integration problem is formulated as

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2} y^\top M^{-1} y + \frac{1}{2} \log |M| + \frac{n}{2} \log 2\pi \\ \text{s.t.} \quad & M = \alpha_0 I + \sum_{k=1}^{K} \alpha_k C_k. \end{aligned}$$

And its time complexity is $O(n^3K)$.

A.5. Affinity Network Fusion (ANF)

ANF [15] is a graph integration method based on random walk. The r-step random walk can be interpreted as the probability of a path between nodes in a graph that passes through r edges. And it is expressed in the form of the state transition matrix W (each row sums to 1) multiplied itself by r times (W^r). This method utilizes an update function that represents r-step random walk in the case of multiple graphs. The update is performed only once for each graph, and we use the update function for the case $r = 2$ as follows:

$$W_k = \beta W_k \Big(\sum_{i \ne k} \frac{\alpha_i}{\sum_{j \ne k} \alpha_j} W_i \Big) + (1-\beta) \Big(\sum_{i \ne k} \frac{\alpha_i}{\sum_{j \ne k} \alpha_j} W_i \Big) W_k, \tag{19}$$

where β is a hyperparameter that determines the trade-off between the two terms in (19). And α_k set as a hyperparameter or uniform value ($\frac{1}{K}$). After the update, graph integration is performed via linear combination like other methods. And its computational complexity is $O(n^2K)$.

A.6. Multiplex Heterogeneous Graph Convolutional Network (MHGCN)

The MHGCN [22] is a deep learning model designed to handle multiplex heterogeneous graphs, which are graphs containing multiple types of nodes and edges. This model captures meta-path information across multi-relations in heterogeneous graphs using the following a multi-layer graph convolution module:

$$H^{(l)} = (W^*)^l H^{(0)} \underbrace{\Theta^{(1)} \cdots \Theta^{(l)}}_{l}, \quad W^* = \sum_{k=1}^{K} \alpha_k W_k,$$

where $H^{(l)}$ is the output of lth layer ($H^{(0)}$ is a node feature matrix, but it is replaced with $I \in \mathbb{R}^{n \times n}$ to utilize only the graph structure information), $\Theta^{(l)}$ is the learnable weight matrix for lth graph convolution layer. Additionally, α_k is also a learnable parameter tuned by the back propagation algorithm in this model. The final node representation matrix H is derived through an aggregation process for each layer as follows:

$$H = \frac{1}{l} \sum_{i=1}^{l} H^{(i)}$$

By fusing the outputs of all layers, meta-path information of various lengths is captured in H. For comparison, we trained a classifier on H to perform node classification.

Data availability

Data will be made available on request.

References

[1] K. Musiał, P. Kazienko, Social networks on the internet, World Wide Web 16 (1) (2013) 31–72.
[2] H. Shin, K. Tsuda, B. Schölkopf, Protein functional class prediction with a combined graph, Expert Syst. Appl. 36 (2) (2009) 3284–3292.
[3] G. Lin, K. Liao, B. Sun, Y. Chen, F. Zhao, Dynamic graph fusion label propagation for semi-supervised multi-modality classification, Pattern Recognit. 68 (2017) 14–23.
[4] S. Bahrami, F. Dornaika, A. Bosaghzadeh, Joint auto-weighted graph fusion and scalable semi-supervised learning, Inf. Fusion 66 (2021) 213–228.
[5] H. Jiang, C. Tao, Y. Dong, R. Xiong, Robust low-rank multiple kernel learning with compound regularization, European J. Oper. Res. 295 (2) (2021) 634–647.
[6] Y. Tang, Z. Pan, X. Hu, W. Pedrycz, R. Chen, Knowledge-induced multiple kernel fuzzy clustering, IEEE Trans. Pattern Anal. Mach. Intell. (2023).
[7] J. Liu, X. Liu, Y. Yang, Q. Liao, Y. Xia, Contrastive multi-view kernel learning, IEEE Trans. Pattern Anal. Mach. Intell. 45 (8) (2023) 9552–9566.
[8] W. Guo, Z. Wang, W. Du, Robust semi-supervised multi-view graph learning with sharable and individual structure, Pattern Recognit. 140 (2023) 109565.
[9] G.R. Lanckriet, T. De Bie, N. Cristianini, M.I. Jordan, W.S. Noble, A statistical framework for genomic data fusion, Bioinformatics 20 (16) (2004) 2626–2635.
[10] K. Tsuda, H. Shin, B. Schölkopf, Fast protein classification with multiple networks, Bioinformatics 21 (suppl_2) (2005) ii59–ii65.
[11] J. Ye, L. Akoglu, Robust semi-supervised classification for multi-relational graphs, 2015, arXiv preprint arXiv:1510.06024.
[12] S. Mostafavi, D. Ray, D. Warde-Farley, C. Grouios, Q. Morris, GeneMANIA: a real-time multiple association network integration algorithm for predicting gene function, Genome Biol. 9 (1) (2008) 1–15.
[13] T. Kato, H. Kashima, M. Sugiyama, Robust label propagation on multiple networks, IEEE Trans. Neural Netw. 20 (1) (2008) 35–44.
[14] B. Wang, A.M. Mezlini, F. Demir, M. Fiume, Z. Tu, M. Brudno, B. Haibe-Kains, A. Goldenberg, Similarity network fusion for aggregating data types on a genomic scale, Nature Methods 11 (3) (2014) 333–337.
[15] T. Ma, A. Zhang, Affinity network fusion and semi-supervised learning for cancer patient clustering, Methods 145 (2018) 16–24.
[16] Z. Kang, G. Shi, S. Huang, W. Chen, X. Pu, J.T. Zhou, Z. Xu, Multi-graph fusion for multi-view spectral clustering, Knowl.-Based Syst. 189 (2020) 105102.
[17] S. Park, M.J. Kim, K. Park, H. Shin, Mutual domain adaptation, Pattern Recognit. 145 (2024) 109919.
[18] Z.N. Kesimoglu, S. Bozdag, SUPREME: multiomics data integration using graph convolutional networks, NAR Genom. Bioinform. 5 (2) (2023) lqad063.
[19] T. Zhang, C. Hou, R. Jiang, X. Zhang, C. Zhou, K. Tang, H. Lv, Label informed contrastive pretraining for node importance estimation on knowledge graphs, IEEE Trans. Neural Netw. Learn. Syst. (2024).
[20] J. Gilmer, S.S. Schoenholz, P.F. Riley, O. Vinyals, G.E. Dahl, Neural message passing for quantum chemistry, in: International Conference on Machine Learning, PMLR, 2017, pp. 1263–1272.
[21] L. Gong, Q. Cheng, Exploiting edge features for graph neural networks, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 9211–9219.
[22] P. Yu, C. Fu, Y. Yu, C. Huang, Z. Zhao, J. Dong, Multiplex heterogeneous graph convolutional network, in: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2022, pp. 2377–2387.
[23] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, P.S. Yu, Heterogeneous graph attention network, in: The World Wide Web Conference, 2019, pp. 2022–2032.
[24] J. Melton, S. Krishnan, muxGNN: Multiplex graph neural network for heterogeneous graphs, IEEE Trans. Pattern Anal. Mach. Intell. (2023).
[25] Z.N. Kesimoglu, S. Bozdag, Fusing multiplex heterogeneous networks using graph attention-aware fusion networks, Sci. Rep. 14 (1) (2024) 29119.
[26] H. Shin, K. Tsuda, Prediction of Protein Function from Networks, MIT Press, 2006, pp. 343–356.
[27] X. Zhu, Z. Ghahramani, J.D. Lafferty, Semi-supervised learning using gaussian fields and harmonic functions, in: Proceedings of the 20th International Conference on Machine Learning (ICML-03), 2003, pp. 912–919.
[28] X. Zhu, Semi-Supervised Learning with Graphs, Carnegie Mellon University, 2005.
[29] Y. Bengio, O. Delalleau, N. Le Roux, Label Propagation and Quadratic Criterion, MIT Press, 2006, pp. 183–206.
[30] M. Belkin, I. Matveeva, P. Niyogi, Regularization and semi-supervised learning on large graphs, in: Learning Theory: 17th Annual Conference on Learning Theory, COLT 2004, Banff, Canada, July 1-4, 2004. Proceedings 17, Springer, 2004, pp. 624–638.
[31] O. Chapelle, J. Weston, B. Schölkopf, Cluster kernels for semi-supervised learning, Adv. Neural Inf. Process. Syst. 15 (2002).
[32] X. Zhu, J. Lafferty, Z. Ghahramani, Semi-Supervised Learning: From Gaussian Fields to Gaussian Processes, School of Computer Science, Carnegie Mellon University, 2003.
[33] F.R. Chung, F.C. Graham, Spectral Graph Theory, vol. 92, American Mathematical Soc., 1997.
[34] K.B. Petersen, M.S. Pedersen, et al., The matrix cookbook, Tech. Univ. Den. 7 (15) (2008) 510.
[35] D.A. Spielman, S.-H. Teng, Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems, in: Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing, 2004, pp. 81–90.
[36] C.E. Rasmussen, C.K. Williams, et al., Gaussian Processes for Machine Learning, vol. 1, Springer, 2006.
[37] Y. Dou, Z. Liu, L. Sun, Y. Deng, H. Peng, P.S. Yu, Enhancing graph neural network-based fraud detectors against camouflaged fraudsters, in: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, 2020, pp. 315–324.
[38] S. Boyd, S.P. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, 2004.
[39] A.P. Dempster, N.M. Laird, D.B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, J. R. Stat. Soc. Ser. B Stat. Methodol. 39 (1) (1977) 1–22.
[40] C.J. Wu, On the convergence properties of the EM algorithm, Ann. Statist. (1983) 95–103.

Taehwan Yun received M.S. degree from Ajou University in 2024 and is currently pursuing his Ph.D. degree at Graduate School of Artificial Intelligence, Ajou University, South Korea. His research interest is on graph-based machine learning, especially multi-modal learning on graphs, and algorithms for heterogeneous graphs.

Myun Jun Kim received the Ph.D. degree in Artificial Intelligence from Ajou University in 2021. His research interest is on machine learning, especially semi-supervised learning, algorithms and applications for networks with multi-layered structure.

Hyunjung Shin received Ph.D. degree in Data Mining from Seoul National University, and further majored in Machine Learning during her Post-Doc at Max Planck Institute (MPI) Tübingen in Germany. Since 2006, she joined Ajou University as a faculty member of the Department of Industrial Engineering. Currently, she provides academic services as a member of board of directors in Business Intelligence & Data Mining Society, Korean Institute of Information Scientists and Engineers (KIISE), Artificial Intelligence Society at KIISE, Korean Institute of Industrial Engineers, and Korean Society for Bioinformatics and Systems Biology. Also, she has been the vice chairman for Billing Software Inspection and Review Committee of Health Insurance Review and Assessment Service. Theory interest of her is focused on Machine Learning algorithms, particularly in Kernel and Semi-Supervised Learning methods. Her research activities range across diverse areas including network analytics, biomedical informatics, hospital fraud detection, etc.