Eduardo Althoff *et al.*
Applied Network Science (2023) 8:82
https://doi.org/10.1007/s41109-023-00606-y

Applied Network Science

RESEARCH Open Access

Coarsening effects on *k*-partite network classification

Paulo Eduardo Althoff[1], Alan Demétrius Baria Valejo[2] and Thiago de Paulo Faleiros[1*]

*Correspondence: thiagodepaulo@unb.br

[1] Department of Computer Science, University of Brasilia, Brasília 70910-900, Brazil
[2] Department of Computing, Federal University of São Carlos, Rod. Washington Luís, km 235 - SP-310, São Carlos 13565-905, Brazil

Abstract

The growing data size poses challenges for storage and computational processing time in semi-supervised models, making their practical application difficult; researchers have explored the use of reduced network versions as a potential solution. Real-world networks contain diverse types of vertices and edges, leading to using *k*-partite network representation. However, the existing methods primarily reduce uni-partite networks with a single type of vertex and edge. We develop a new coarsening method applicable to the *k*-partite networks that maintain classification performance. The empirical analysis of hundreds of thousands of synthetically generated networks demonstrates the promise of coarsening techniques in solving large networks' storage and processing problems. The findings indicate that the proposed coarsening algorithm achieved significant improvements in storage efficiency and classification runtime, even with modest reductions in the number of vertices, leading to over one-third savings in storage and twice faster classifications; furthermore, the classification performance metrics exhibited low variation on average.

Keywords: Network semi-supervised learning, Network coarsening, Heterogeneous network

Introduction

Semi-supervised learning emerged as a way of dealing with many previously labeled data to guide a supervisor's response. These methods use labeled data combined with unlabeled data to perform the learning. Algorithms of this type consider relationships between unlabeled and labeled data to compensate for the lack of labels (van Engelen and Hoos 2020). Representing data as networks can enhance semi-supervised technique by allowing for the extraction of relationships from the topological characteristics of labeled and unlabeled data.

Although the semi-supervised approach has brought a solution to reduce the need for human intervention, a problem is still present. As data grows, training a semi-supervised model's storage cost and computational processing time can become quite large, making it impractical to use in some applications (Walshaw 2004). Recently, a technique widely studied to overcome these limitations is the use of reduced versions of networks in place of the original ones (Liu et al. 2018). Therefore, this technique reduces the storage requirements and improves the performance of algorithms compared to using a

Springer Open

© The Author(s) 2023. **Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

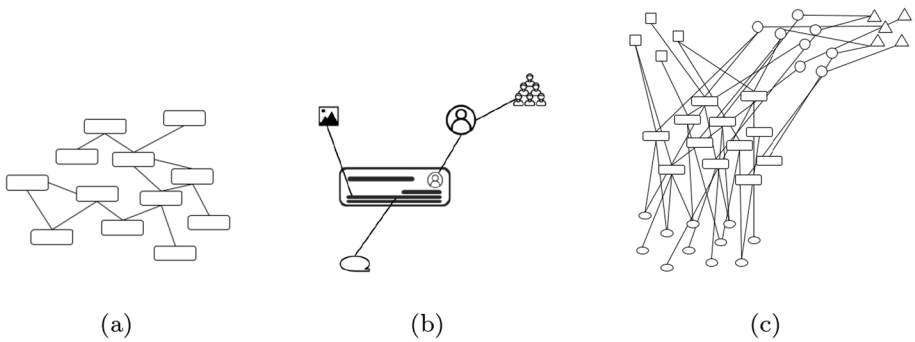

Fig. 1 Example of heterogeneous data representation. Figure **a** showcases interactions among social network posts with a uni-partite scheme. The schema in figure **b** illustrates information from the posts of the words and pictures contained in it, the users who made them, and, at a second level, the groups to which these users belong, forming a heterogeneous *k*-partite network in figure **c**

full-sized graph. A category within these techniques is the *coarsening* algorithms, which join groups of similar vertices, often reducing redundant information. This technique is well established in visualization and graph partitioning and has recently been proven valid for classification problems in homogeneous networks (Liang et al. 2020).

However, most of these methods primarily focus on analyzing networks with only one type of vertex and edge, known as uni-partite networks. Real-world information networks are often heterogeneous, consisting of various types of vertices and edges. One widely-used approach for heterogeneous network representation involves dividing the diverse data types into disjoint subsets or partitions. Edges are then used to depict the relationships between these different types, giving rise to *k*-partite networks. This type of representation offers greater flexibility and expressiveness, enabling the modeling of various relationship types between objects, as illustrated in Fig. 1.

Traditional compression strategies overlook the distinctions between vertex types, even though layers within a *k*-partite network generally correspond to diverse entity types necessitating separate treatment. To illustrate, consider a document collection scenario modeled as a document-work bipartite network. Combining word vertices with document vertices (i.e., matching vertices across different layers) would need more significance in this and most application contexts. Furthermore, as the word count is substantially higher, simplifying the word layer might effectively curtail the asymptotic convergence of resource-intensive algorithms. Additionally, amalgamating vertices of diverse types (such as words, authors, documents, users, and locations) would introduce a novel hybrid entity not present in the original network, thereby altering its inherent *k*-partite structure.

In semi-supervised learning, the model utilizes the labeled data to make predictions and the unlabeled data to improve the overall accuracy and robustness of the model. By leveraging the rich information in the heterogeneous *k*-partite network, the model can learn more robust and generalizable data representations, improving its performance on downstream tasks such as classification and prediction. Label Propagation is a network-based semi-supervised learning algorithm propagating labels or information through a graph. In a heterogeneous graph, different types of vertices and edges exist, and these vertices can represent various entities, attributes, or relationships. Heterogeneous

networks are often used to model complex and diverse data, such as social networks, recommendation systems, and knowledge graphs. In the context of heterogeneous networks, the Label Propagation algorithm aims to infer missing labels for vertices based on the information in the network. It assumes that vertices connected or sharing certain relationships will likely have similar labels. The basic idea is to iteratively update the vertices' labels based on their neighboring vertices' labels, expecting the labels to become more coherent over iterations. In this study, while we examine a k-partite network as our input, we focus solely on classifying vertices from one specific partition, the target partition $\mathcal{V}_t$. The remaining partitions are employed in the propagation process and help the label spread.

As the interest in techniques for heterogeneous networks increases, as seen in studies such as Zhou et al. (2020), Liu et al. (2018), Wu et al. (2021), research on coarsening methods has also gained traction, particularly for bipartite networks (Valejo et al. 2017a, b, 2018, 2020a, 2021). However, methods designed explicitly for heterogeneous networks have yet to be extensively explored. In this work, we want to determine how well the coarsened heterogeneous networks can accurately classify vertices based on their relationship to other vertices in a semi-supervised context. Coarsening can lead to a loss of information and a decrease in classification accuracy. Therefore, evaluating the trade-off between computational efficiency and classification accuracy is essential when coarsening in heterogeneous network classification tasks.

In this context, this work presents the following contributions:

1 The development of a new coarsening method applicable to the k-partite network. Our proposed method uses a technique that orders partitions and selects paths in the schema to improve the coarsening performance.
2 An analysis of the impacts of *coarsening* on k-partite network classification metrics identified the existence of a *threshold* that can be used to solve problems in real networks. These analyses can identify appropriate network reduction levels in each case, depending on time/memory constraints and the acceptability of losses in the classification performance.

An empirical analysis of hundreds of thousands of synthetically generated networks shows that coarsening techniques are promising for solving large networks' storage and processing problems.

The remainder of the paper is organized as follows: "Background" section introduces the notation and describes the multilevel method and coarsening algorithm previously proposed in bipartite networks. The proposed coarsening strategy for k-partite networks is described in "Coarsening algorithm for k-partite network" section. "Experimental results" section presents the results performed on the synthetically generated networks and an experiment conducted using a real network. To accommodate computational resource limitations, the experiments with synthetical networks were split into two groups: the first group, referred to as "Experiment 1" in Section 4.3.1, focused on small networks with a maximum of 15, 000 vertices, while the second group, "Experiment 2" in Section 4.3.2, involved larger networks with a maximum of 100, 000 vertices. "Concluding remarks" section summarizes our findings and discusses future work.

Background

A network (or graph) $G = (V, E)$ is said to be k-partite if V is composed of k disjoint sets $V = \mathcal{V}_1 \cup \mathcal{V}_2 \cup ... \cup \mathcal{V}_k$, where $\mathcal{V}_i$ and $\mathcal{V}_j$ $(1 \leq i, j \leq k)$ are set of vertices and $E \subseteq \bigcup_{i \neq j} \mathcal{V}_i \times \mathcal{V}_j$, where each edges are between vertices of different sets, *i.e.,* for every edge $e = (a, b)$, $a \in \mathcal{V}_i$ and $b \in \mathcal{V}_j$, for $i \neq j$. An edge (a, b) may have an associated weight, denoted as $\omega(a, b)$ with $\omega : E \rightarrow \mathbb{R}^*$; a vertex a may have an associated weight denoted as $\sigma(a)$ with $\sigma : V \rightarrow \mathbb{R}^*$. We assume that a heterogeneous network is a type of k-partite network in which vertices of the same type form a partition, and vertices of different types are connected. We also assume that the edges connecting different types of nodes are undirected and that the relationship between the nodes is symmetric.

The degree of a vertex $a \in \mathcal{V}_i$, denoted κ_a, is defined as the total weight of its adjacent edges, i.e. $\kappa_a = \sum_{b \in V} w(a, b)$. The h-hop neighborhood of a, denoted $\Gamma_h(a)$, is formally defined as the vertices in set $\Gamma_h(a) = \{b \mid$ there is a path with the minimum number of edges of length h between a and $b\}$. Thus, the 1-hop neighborhood of a, $\Gamma_1(a)$, is the set of vertices adjacent to a; the 2-hop neighborhood, $\Gamma_2(a)$, is the set of vertices 2-hops away from a, and so forth.

In k-partite network context, the network schema refers to the topological structure linking the k partitions. Formally, a network schema of a k-partite network G can be represented by the network $S(G) = (V_S, E_S)$, where V_S is the set of k vertices related to each partition and E_S is the set of edges. For every edge $(k_i, k_j) \in E_S$, there is at least one edge $(a, b) \in E$ such that a vertex a belongs to a partition $\mathcal{V}_i$, and a vertex b belongs to a partition $\mathcal{V}_j$. A metapath is a sequence of edges that connects vertices from different partitions. Formaly, a metapath P in a network schema $S(G)$ is defined as a path in the form of $\mathcal{V}_1, E_{1,2}, \mathcal{V}_2, \ldots, \mathcal{V}_l, E_{l,l+1}, \mathcal{V}_{l+1}$, where $E_{i,j}$ denotes the set of edges of type $(k_i, k_j) \in E_S$.

Our proposed technique uses a label propagation scheme that spreads labeled vertices from a specific partition, called target partition $\mathcal{V}_t$, to all other vertices within the network. Let $\mathcal{V}^L \subset \mathcal{V}_t$ be the set of labeled vertices in the target partition, and $\mathcal{V}^U \subset \mathcal{V}_t$ be the set of unlabeled vertices. Notably, the labeled and unlabeled vertex sets form the target partition vertex set, i.e. $\mathcal{V}^L \cup \mathcal{V}^U = \mathcal{V}_t$. Each vertex in $\mathcal{V}^L$ is associated with a label from a set $C = \{c_1, c_2, \ldots, c_m\}$ with m classes. The matrix $\mathcal{Y} \in \mathbb{R}^{|\mathcal{V}_t|, m}$ represents the weight of labels for the corresponding vertices in $\mathcal{V}_t$. To simplify the notation, we denote $\mathcal{Y}_{a,i}$ and $\mathcal{Y}_{\mathcal{V}_t}$ respectively as the weight of label c_i to a vertex a and the labels assigned to a subset of vertices in $\mathcal{V}_t$. The transductive learning algorithm inputs a labeled training set of vertices $\mathcal{V}^L$ and unlabeled test vertices $\mathcal{V}^U$. It outputs a transductive learner F that assigns a label $c_i \in C$ to each vertex a in $\mathcal{V}^U$, *i.e.,* $F(a) = \arg\max_i \mathcal{Y}_{a,i}$.

Network summarization and coarsening

Network coarsening and network summarization aim to reduce network complexity. Coarsening involves reducing the size of the network by collapsing vertices or edges to form a simplified version of the original network. The coarsened network retains the essential structural characteristics of the original but with fewer vertices and edges. Meanwhile, summarization focuses on creating a concise representation that retains important network features, enabling faster processing, visualization, or analysis while preserving significant patterns or properties (Liu et al. 2018).

Network summarization techniques involve selecting, transforming, or aggregating a given network and producing a network summary as the output. Selection methods, for instance, identify less important vertices or edges, possibly considered outliers, noise, or irrelevant regarding given criteria, and remove them before executing a mining task (Liu et al. 2018). Such strategies "clean" the network rather than contracting it, although reducing the network is possibly a side effect. Network transformation refers to the techniques that project a network to a simple and summarized structure. Network transformation is also related to embedding a network into a lower-dimensional representation while preserving the original network's topology (Blasi et al. 2022).

Network summary based on aggregation involves grouping vertices and edges in a network to create a more compact representation. It is closely related to the coarsening concept. The network coarsening can be considered a network summarization technique based on aggregation. However, many network aggregation algorithms are not feasible as coarsening strategies, as they do not employ a hierarchy of increasingly compressed models, which is a fundamental feature of a multilevel method. Therefore, network coarsening falls within the broader category of graph summary aggregation.

In the context of the k-partite network, summarization techniques, including those that consolidate super-vertices and super-edges, often overlook the distinctive structure of k-partite graphs. Network summarization algorithms generally aim to minimize some approximation or reconstruction error metric (LeFevre and Terzi 2010; Riondato et al. 2014; Lagraa et al. 2014). For instance, the normalized reconstructed error can be defined as $\frac{1}{|V|^2}\sum_{i\in V}|\hat{A}[i,j]-A[i,j]|$, where A is the original adjacency matrix of the network and $\hat{A}$ is the real-valued approximate adjacency matrix, each entry of which intuitively represents the probability of the corresponding edge existing in the original graph given the summary. These metrics fail to keep the k-partite network's structure, leading to diminished expressiveness and the inevitable loss of information within the partitions.

Our interest lies specifically in network coarsening processes in the context of the k-partite network applied to increase the scalability of transductive classification tasks. Our research indicates that we have not encountered prior works directly related to our context.

Multilevel method in bipartite network

The multilevel approach enables the implementation of complex algorithms on large-scale networks by shrinking the network size. For instance, take a problem defined on a bipartite network $G^0(V^0=\mathcal{V}_1\cup\mathcal{V}_2,E^0)$, where running the target algorithm is infeasible.

The coarsening phase creates a hierarchy of simplified networks G^l, where $l\in[1,...,L-1]$ and L are the desired coarsening levels. This process yields intermediate representations of the networks with varying levels of detail. The coarsening process involves two algorithms: *matching* and *contraction*. Matching determines which vertices will be combined, and contraction builds the reduced representation after defining the matching. A matching, represented as $M=\{\mathcal{V}_i\}_{i=1}^r$, is a division of vertex set V^0 into r non-empty disjoint subsets. There are restrictions on how vertices can be matched. For example, if $|\mathcal{V}_i|=2$ for all i, then vertices must be paired. A vertex $a\in\mathcal{V}_i$ is considered

matched, while a vertex that does not belong to any $\mathcal{V}_i$ in M is referred to as *unmatched* or a *singleton*.

The reduction factor of the network, denoted as ρ, is determined by an additional input parameter $\rho \in [0, 1] \subset \mathbb{R}$. This parameter defines the size of the matching M, represented as ζ, where $\zeta = \sum_{i=1}^{r} |\mathcal{V}_i|$. If vertices are matched pairwise, then $\sum_{i=1}^{r} |\mathcal{V}_i| = \lceil \rho |V| \rceil$, *i.e.,* the number of vertices to be matched is given by the reduction factor multiplied by the network size.

The Contraction Algorithm creates a simpler network by combining a group of matched vertices $a_1, \ldots, a_n$ into a single entity known as the *super-vertex* $s\mathcal{V}_i$. The vertices $a_1, \ldots, a_n$ in V^l that form the super-vertex $s\mathcal{V}_i$ in V^{l+1} are referred to as the precursor vertices of $s\mathcal{V}_i$. The successor network G^{l+1} inherits the non-matched vertices from its predecessor network G^l. To ensure that G^{l+1} serves as an accurate representation of its predecessor network, the weight of a super-vertex $s\mathcal{V}_i = u_1, \ldots, u_n \in V^{l+1}$ is calculated as the sum of the weights of its precursor vertices. Additionally, the edges connecting to vertices $u_1, \ldots, u_n \in V^l$ are collapsed to form the *super-edges* that attach to $s\mathcal{V}_i$.

Coarsening algorithm for bipartite networks

Progressively reducing the network size to obtain coarser network representations is part of the multilevel technique, usually applied in network optimization problems. A multilevel optimization meta-heuristic combines various heuristics to guide, modify, and refine a solution obtained from a target algorithm or subordinate heuristics, such as local or global search, over multiple iterations. This technique operates in three phases: coarsening, solution finding, and uncoarsening. The network size is progressively reduced during the coarsening phase to obtain coarser network representations. In the solution-finding phase, the starting solution is obtained by applying the target algorithm to the coarsest representation. In the uncoarsening phase, the starting solution is projected back to the intermediate networks and refined successively until the final solution is obtained. Figure 2 illustrates such a process, considering an initial network G^0 (in which the original problem instance is defined), where G^L denotes the coarsest network obtained after L coarsening steps (levels).

It is important to note that the coarsening process is a crucial aspect of the multilevel method, as it is a problem-agnostic step, in contrast to the other phases (Valejo et al. 2020b). Therefore, many algorithms have been developed, and some strategies designed for handling bipartite networks have gained widespread recognition.

The first method, OPM_{hem} (Valejo et al. 2017a, b), decomposes the bipartite network into two separate unipartite networks, but this may result in a loss of information. In Valejo et al. (2018), the authors introduced two coarsening algorithms, RGMb and GMb, which directly use the bipartite network to select a set of vertices in pairs. Later, the authors in Valejo et al. (2020a) proposed a coarsening method based on label propagation but did not indicate stability and convergence guarantee. The most recent method, the CLPb algorithm (Valejo et al. 2021), employs a semi-synchronous strategy through cross-propagation, providing a time-efficient implementation and effectively reducing the oscillation issue. The empirical analysis showed that the CLPb algorithm is more accurate and efficient than previous methods.

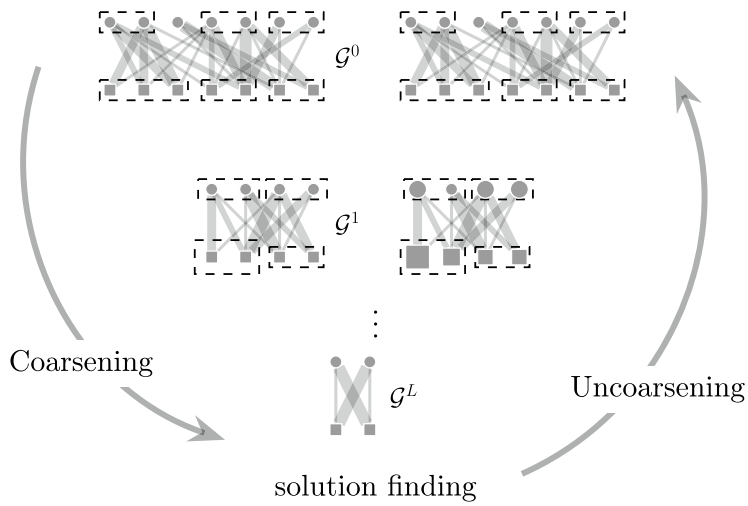

Fig. 2 Overview of the general-purpose multilevel optimization for the bipartite network. The original network is reduced, level-by-level, until a desired network size is reached. The number of coarsening levels L is an input parameter, and the dotted rectangles denote matched vertices

Coarsening via semi-synchronous label propagation for bipartite networks: CLPb

In this section, we elucidate the **c**oarsening strategy via semi-synchronous **l**abel **p**ropagation for **b**ipartite networks (CLPb) as introduced in Valejo *et al.*'s work (Valejo et al. 2021). Unlike our proposed approach, the CLPb algorithm was originally designed for the unsupervised scenario.

Here, we will introduce label definitions that deviate from the notation presented in this article to enhance the comprehension of the CLPb algorithm. The labels are represented as a tuple $\mathcal{L}_a(c, \beta)$, where c signifies the current label, and $\beta \in [0, 1] \subset \mathbb{R}^+$ represents its associated score. Initially, every vertex $a \in V$ is assigned a starting label $\mathcal{L}_a = \left(a, 1.0/\sqrt{\kappa(a)}\right)$, where the initial $\mathcal{L}_a$ is identified by its "id" (or "name") with a maximum score of $\beta = 1.0$.

In each step, a new label is propagated to a receiving vertex a by selecting the label with the highest β from the collective labels of its neighboring vertices, denoted as $\mathcal{L}_a = \bigcup \mathcal{L}_b$, $\forall b \in \Gamma_1(a)$. This propagation process adheres to the subsequent filtering rules:

1. Equal labels $\mathcal{L}^{eq} \subseteq \mathcal{L}_a$ are merged and the new $\beta^{'}$ is composed by the sum of its belonging scores: $\beta^{'} = \sum_{(l,\beta)\in\mathcal{L}^{eq}} \beta$,
2. The belonging scores of the remaining labels are normalized, i.e.: $\mathcal{L}_a = \{(l_1, \frac{\beta_1}{\beta^{sum}}), (l_2, \frac{\beta_2}{\beta^{sum}}), \ldots, (b_\gamma, \frac{\beta_\gamma}{\beta^{sum}})\}$, where $\beta^{sum} = \sum_{i=1}^{\gamma} \beta_i$ and γ is the number of remaining labels.
3. The label with the largest β is selected: $\mathcal{L}_a^{'} = \underset{(l,\beta)\in\mathcal{L}_a}{\arg\max}\, \mathcal{L}_a$.
4. The size of the coarsest network is naturally controlled by the user, i.e. require defining a number of reduction levels, a reduction rate or any other parameter to fit a desired network size. Here, each layer's minimum number of labels η is a user-defined parameter. A vertex $a \in \mathcal{V}_i$, with $i \in \{1, 2\}$ define a bipartite layer, is only allowed to

update its label if, and only if, the number of labels in the layer $|\mathcal{L}^i|$ remains equal to or greater than η^i, *i.e.:* $|\mathcal{L}^i| \leq \eta^i$.

5. At last, a classical issue in the multilevel context is that super-vertices tend to be highly unbalanced at each level (Valejo et al. 2020b). Therefore, it is common to constrain the size of the super-vertices from an *upper-bound* $\mu \in [0, 1] \subset \mathbb{R}^+$, which limits the maximum size of a group of labels in each layer: $\mathcal{S}^i = \frac{1.0+(\mu*(\eta^i-1))*|\mathcal{V}_i|}{\eta^i}$, wherein $\mu = 1.0$ and $\mu = 0$ imply highly imbalanced and balanced groups of vertices, respectively. Therefore, a vertices a with weight $\sigma(a)$ can update its current label l to a new label l' if, and only if $\sigma(a) + \sigma(l') \leq S^i$ and $\sigma(l') = \sum_{b \in l'} \sigma(b)$.

In Fig. 3, we can observe a single step of CLPb in a bipartite network utilizing the previously defined strategy. The propagation process repeats $\mathcal{T}$ times, a parameter set by the user, until convergence is reached or until label changes cease to occur.

Following the convergence of cross-propagation, the algorithm collapses each cluster of corresponding vertices (i.e., vertices with the same label) into a unified "super-vertex." The edges connected to these matched vertices are collapsed into what is called "super-edges." This process is visually depicted in Fig. 4.

The iterative CLPb coarsening algorithm transforms the original network $\mathcal{G}^0$ into a hierarchy of smaller networks denoted as $\mathcal{G}^1, \mathcal{G}^2, \ldots, \mathcal{G}^L, \ldots$, where $\mathcal{G}^L$ represents an arbitrary level. Users can control the maximum levels and the reduction factor ρ for each layer instead of specifying the desired number of vertices in the coarsest network.

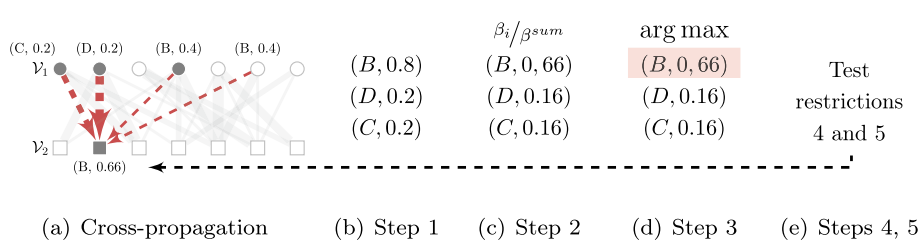

Fig. 3 One step of the CLPb algorithm in a bipartite network. In **a**, the process is performed from the top layer, considering the propagators vertex $\in \mathcal{V}_1$, to the bottom layer, considering the receiver nodes $\in \mathcal{V}_2$. At first, represented in (**b**), equal labels are merged. In **c**, second step, remaining labels are normalized. In third step, the label B is selected, as showed in (**d**). In **e**, the restriction 4 and 5 are tested. Finally, label B is propagated to the node in the bottom layer, as illustrated by the black dashed line

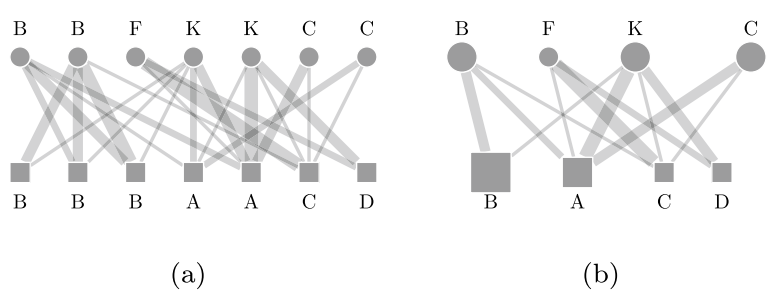

Fig. 4 Contraction process. In **a**, a group of vertices are matched using CLPb algorithm; in **b**, the original network is coarsened, i.e., vertices that share labels are collapsed into super-vertices and edges incident to matched vertices are collapsed into super-edges

The computational complexity of the Label Propagation is nearly linear for the number of edges, i.e., it requires $\mathcal{O}(|V| + |\mathcal{E}|)$ operations at each iteration. Assuming a constant number of $\mathcal{T}$ iterations, the overall complexity becomes $\mathcal{O}(\mathcal{T}(|V| + |\mathcal{E}|))$. For the contraction process (as depicted in Fig. 4), it first iterates through all matched vertices in network G^L to generate super-vertices for G^{L+1}. Subsequently, each edge in G^L is chosen to create super-edges in G^{L+1}, which also incurs a complexity of $\mathcal{O}(|V| + |\mathcal{E}|)$. These complexities are well-documented in the literature, with more extensive discussions available in Valejo et al. (2020b) and Raghavan et al. (2007). Taking these considerations into account, the overall computational complexity of CLPb at each level is $\mathcal{O}(\mathcal{T}(|V| + |\mathcal{E}|)) + \mathcal{O}(|V| + |\mathcal{E}|)$.

Coarsening algorithm for *k*-partite network

This section presents the proposed coarsening algorithm for reducing k-partite networks to facilitate subsequent classification tasks. The algorithm utilizes labeled vertices from the target partition $\mathcal{V}_t$ to guide the reduction process. The k-partite network is first decomposed into a series of bipartite networks, with pairs of partitions selected from the original network. Next, an adaptation of the CLPb coarsening algorithm is applied to these pairs of partitions, with one partition acting as the propagator partition, $\mathcal{V}_p$, and the other as the receptor partition, $\mathcal{V}_r$. The coarsening process is executed semi-synchronously, and vertices from $\mathcal{V}_r$ with the same labels are grouped into super-vertices. An illustration of the coarsening process in a k-partite network is presented in Fig. 5. As methods applied to networks have complexity generally associated with the number of vertices and edges, this coarsening procedure intends to reduce the overall training time for transductive learning.

Once label propagation has been established as a matching approach for each bipartition, the next step is determining the strategy for selecting the pairs of partitions and the order in which CLPb will be applied. Considering all possible partition pairs can lead to numerous procedures in networks with high connectivity schemas, resulting in a quadratic complexity in the number of vertices (Zhu et al. 2016). Additionally, the presence of cycles in the network schema would result in the repetition of information. To overcome these challenges, we aimed to identify, for each

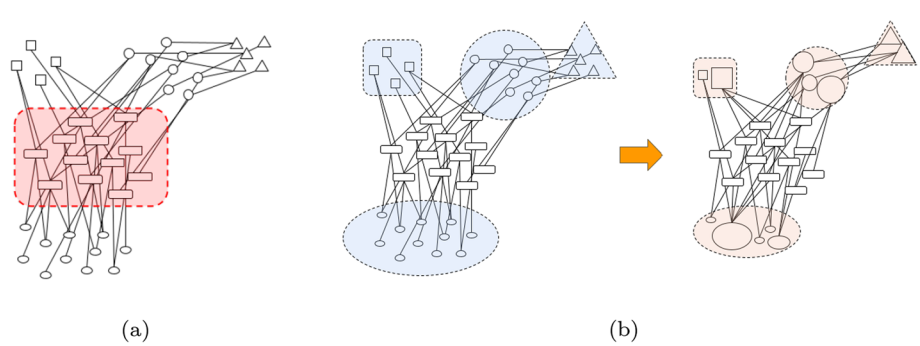

Fig. 5 Example of coarsening realized by the proposed algorithm in a heterogeneous network. The target partition is highlighted in red in figure **a**. The other partitions highlighted in blue in figure **b** are reduced by the coarsening process

partition, the most suitable neighboring partition to act as a pair in the bipartite coarsening procedure. We called this neighbor partition "guide partition."

One approach to limit the number of partition pairs used is to locate paths in the schema of the k-partite network (Luo et al. 2014). As only the target partition has label information at the beginning, a logical approach for the coarsening procedure is to propagate the information from the target partition to the others utilizing the label information during the matching phase. Shorter paths are more likely to indicate a strong relationship between vertices (Gupta et al. 2017), and thus, the shortest metapath between the two partitions was selected.

The goal is to perform coarsening on all non-target partitions following a metapath. The procedure is performed synchronously, one partition pair at a time, and the partition pairs are selected radially starting from the target partition. First, the guide partitions at a 1-hop distance from the target are coarsened, followed by those at a 2-hop distance, and so on. For each guide partition being reduced, the pair used is the neighbor partition within the metapath that goes from the partition being reduced to the direction of the target partition. This chosen order not only propagates label information initially present in the target partition but also ensures that the selected neighbor partition has already undergone a reduction in the previous coarsening process (except in the first iteration), as illustrated in Fig. 6.

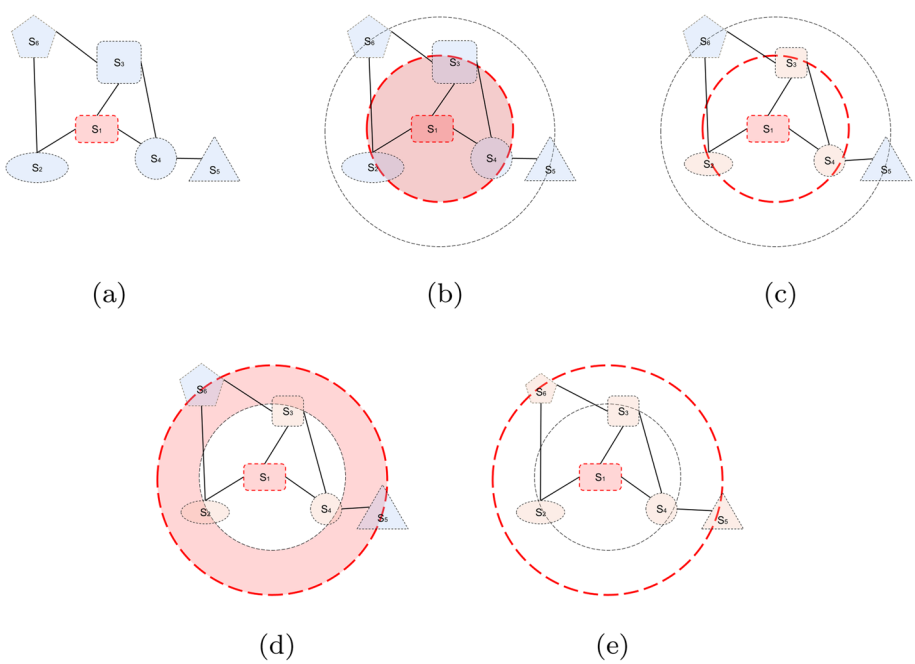

Fig. 6 The coarsening process is executed radially, starting from the target partition $S_t = S_1$. As seen in Fig. 6 (**a**), the first step is to perform coarsening on the immediate neighbors of S_1 (from **b** to **c**), who are only influenced by the target partition. Subsequently, coarsening is carried out on partitions that are farther away from S_t (from **d** to **e**), with these partitions being influenced only by their guide partitions, which carry structural information from S_a

Algorithm description

The procedure of coarsening for k-partite network is described by Algorithm 1, taking as input a k-partite network $G = (V, E)$, a schema $S(G)$ of the k-partite network, and a vertex $\mathcal{S}^t$ in schema $S(G)$ correspondent to the target partition $\mathcal{V}_t$. The output is a coarsened version of k-partite network G.

Algorithm 1 *Coarsening* procedure for non-target partitions

input : $G = (V, E)$, $S(G)$, S^t
output : coarsed graph G^c
1 **begin**
2 $bfsList \leftarrow [\mathcal{S}^t]$
3 $added \leftarrow [\mathcal{S}^t]$
4 $guide \leftarrow \{\}$
5 $G^c \leftarrow G$
6 **while** $bfsList.length \neq 0$ **do**
7 $\mathcal{S}_i \leftarrow pull(bfsList)$
8 **forall** $\mathcal{S}_k \in \Gamma_1(\mathcal{S}_i)$ **do**
9 **if** $\mathcal{S}_k \notin added$ **then**
10 $added.push(\mathcal{S}_k)$
11 $bfsList.push(\mathcal{S}_k)$
12 $guide[\mathcal{S}_k] \leftarrow \mathcal{S}_i$
13 **if** $\mathcal{S}_i \neq \mathcal{S}^t$ **then**
14 $\mathcal{S}_j \leftarrow guide[\mathcal{S}_i]$
15 $G_i \leftarrow (\mathcal{V}_j, \mathcal{V}_i, E_i) \leftarrow getBipartiteSubgraph(G^c, \mathcal{S}_j, \mathcal{S}_i)$
16 $G_i^c \leftarrow ({\mathcal{V}_j}^c, \mathcal{V}_i^c, E_i^c) \leftarrow coarseningBiPartite(G_i)$
17 $G^c \leftarrow ((V/\mathcal{V}_i) \cup \mathcal{V}_i^c, (E - E_i) \cup E_i^c)$
18 return G^c

Let P_i be the shortest metapath starting from the labeled target partition S^t and ending at non-target partition S_i. If multiple shortest metapaths exist, the first one found is chosen. For each S_i, a 1-hop distance partition $S_j \in P_i$, $\Gamma_1(\mathcal{S}_i)$, is selected as a "guide partition." Fig. 7 illustrates this process.

The coarsening process is applied to non-target partitions using a breadth-first search order in S (represented in the loop from line 6), starting from S^t. The target partition $\mathcal{V}_t$ is the father in the breadth-first search tree. The goal is to replace in G, at each iteration of this loop, the vertices from $\mathcal{V}_i$ for the super-vertices $\mathcal{V}_i^c$ (the superscript c indicate a coarsening version of the partition $\mathcal{V}_i$) and its associated super-edges (see line 17). These super-vertices and super-edges are obtained from the bipartite coarsening of G, the subgraph composed by the nodes $\mathcal{V}_j \cup \mathcal{V}_i$ and its associated edges. Since the vertices of $\mathcal{V}_i$ are replaced, and we follow a breadth-first search order, when in a later iteration $\mathcal{V}_i$ is a guide-partition, its already compressed version is used on the bi-partite coarsening, making the procedure from line 16 faster than if was done with its initial (non-compressed) version. At the end, the algorithm then

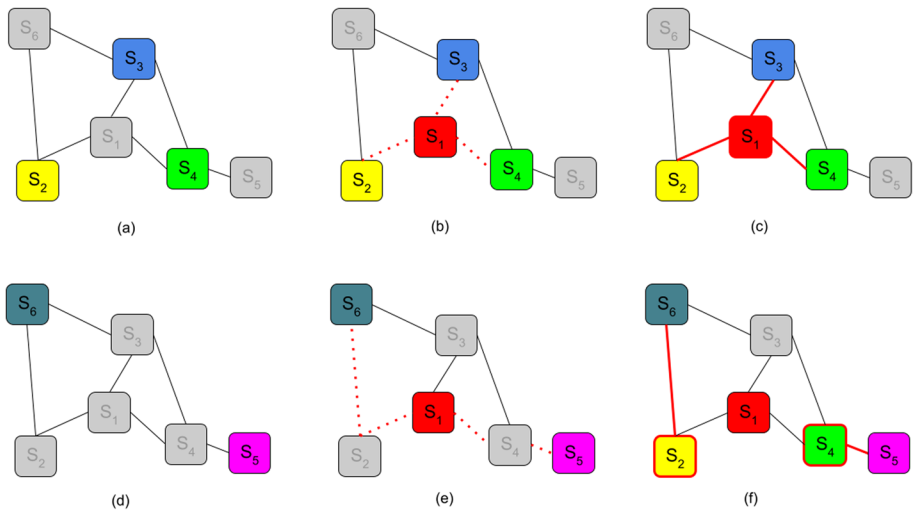

Fig. 7 Suppose $S^t = S_1$ be the target partition. **a** The 1-hop partitions from S_1, described in schema as S_2, S_3, and S_4 are selected; **b** the shortest paths from S_1 to selected partitions are trivial in this case; **c** the guide partitions for S_2, S_3, and S_4 are identified as being S_1 itself; **d** the partitions S_5 and S_6 are selected; **e** the shortest metapaths between S_5 and S_1, and between S_6 and S_1 are found; **f** the guide partition for S_5 is identified as S_4 and the guide partition for S_6 is identified as S_2

returns the coarsened network G^c, obtained by applying coarsening to all non-target partitions of the original k-partite network.

Experimental results

Experimental studies were conducted on synthetic and real datasets using transductive classification to assess the effectiveness of the proposed coarsening algorithm. The primary reduction objectives, namely memory savings and classification runtime, were analyzed as the number of vertices increased. Furthermore, the accuracy of various metrics, such as Accuracy, Precision, Recall, and F-score, were compared for each reduction level relative to the original uncoarsened network. The results of the experiments are presented in subsequent sections, along with insights into the findings.

Synthetic network generation

Although heterogeneous data are ubiquitous, there are few standard datasets for study. Here we use a tool to generate k-partite networks. The tool chosen was HNOC (Valejo et al. 2020c), a synthetic k-partite network generator developed to help analysis of learning methods in networks. The characteristics that led to this choice were the tool's ability to vary the partition size, the number of possible classes, the probability of possible classifications, and the noise and dispersion levels. These features allow for a comprehensive analysis of the classification tasks in the network.

The original purpose of the HNOC tool was for community detection, but its concept of communities can be expanded to classify data in a semi-supervised setting. The tool initially aligns each vertex precisely with its designated community. Each vertex in a community is then connected to all other vertices in the same community and different partitions. Next, edges are selectively removed based on the *dispersion* parameter, which controls community density. Lower dispersion values lead

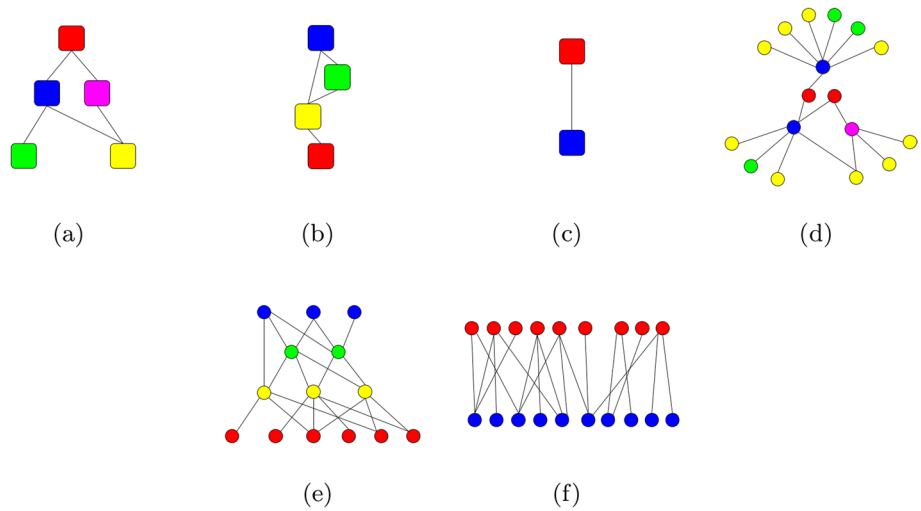

Fig. 8 Distinct topologies networks schemes generated in the experiments: hierarchical star (**a**), hierarchical web (**b**), and bipartite (**c**). The networks examples respectively in figures (**d**) (**e**), and (**f**). The red vertice indicates the partition with the label information

to sparser communities, whereas higher dispersion values result in denser communities. The dispersion parameter regulates intra-community edges. The tool utilizes the *noise* parameter for edges between different communities or inter-community edges. Network noise affects the ability to identify community boundaries and increases overlap, making finding communities within the network more complex. The noise level can significantly decrease classification accuracy and increase complexity. Lower noise levels result in fewer inter-community borders and more easily separable communities. As noise levels increase, inter-community borders become more frequent, making identifying class boundaries harder. Generally, $noise > 0.5$ produces networks with poorly defined and sparse community structures, where inter-community edges outnumber intra-community edges. Optimal noise values range from 0.1 to 0.4, which increases the difficulty of class detection while preserving the overall network structure (Valejo et al. 2020c). It is essential to note that no edge connects vertices within the same partition.

In heterogeneous networks, there are diverse object types and topological structures. Generating various network topologies to simulate real-world scenarios and complex systems is essential. In addition to the dispersion and noise parameters, the type of topological structure of the network varied. Three topologies were selected, namely, the *hierarchical star* (Fig. 8a), the *hierarchical web* (Fig. 8b), and the *bipartite topology* (Fig. 8c). The hierarchical star topology could , for instance, represent user interface devices on computer networks or employees without management roles within a company. In this type of structure, starting from the center of the structure, and moving towards the leaves, each level can be seen as feature vertices of the most central vertex.

The hierarchical web topology is typical of biological networks like food chains. It has a relative order between partitions, with more vertices at lower levels. Laying out the hierarchical topology in a tree-like star, with the most central vertex at

higher levels, it would operate similarly to the hierarchical web, with the distinction that when we divide the partitions into levels based on the number of hop distances within the scheme with respect to the upper partition, the mesh topology allows hops between partitions of non-consecutive level. This distinction has significance in our experiment, as it generates distinct patterns of metapaths that would not arise in star topology.

The last topology is the bipartite structure. This type of network is widely explored in the literature on textual classification problems (Rossi et al. 2012; Faleiros et al. 2016; Redmond and Rozaki 2017). In this case, the vertices of the target partition represent documents, while those of the non-target partition represent words. The edges in the bipartite network example would represent the presence of a word in a document.

Experimental setup

Due to limitations in computational resources, the experiments were divided into two groups. The first group, referred to as *Experiment 1* in Section 4.3.1, comprised small networks with up to 15, 000 vertices, while the second group, *Experiment 2* in Section 4.3.2, consisted of larger networks with up to 100, 000 vertices.

In Experiment 1, a range of parameter configurations was considered to generate synthetic k-partite networks with distinct class structures (Valejo et al. 2020c). The number of vertices ranged from 2000 to 15, 000 in 9 different schemes for each topology (Fig. 8); the number of classes ranged from 4 to 10 at increments of 1; the dispersion ranged from 0.1 to 0.9 at increments of 0.1; and the noise level ranged from 0.1 to 0.9 at increments of 0.1. Ten distinct networks were created for each configuration, resulting in over 130, 000 networks in total. The vertices were evenly distributed among the non-target partitions. To accurately reflect the memory savings associated with the coarsening algorithm, the non-target partitions were set to be five times greater than the target partition.

In Experiment 2, the size of the k-partite networks was increased to up to 100, 000 vertices. However, the variability for some parameters was reduced due to computational limitations. The noise value was fixed at 0.3 to obtain more challenging networks for classification without excessive degradation. Furthermore, five networks with less than 10, 000 vertices and three with more than 60, 000 vertices were created instead of generating ten networks with the same parameter configuration. This decrease in the number of tests resulted in less stable average results and increased the standard deviation.

Results in synthetic *k*-partite networks

The purpose of this study is to assess the effectiveness of the proposed coarsening algorithm in the transductive classification of k-partite networks. The GNetMine algorithm (Ji et al. 2010) was used for the experiments, as it is a widely used reference algorithm in the area of heterogeneous classification and has served as a benchmark for comparison in several studies (Luo et al. 2014; Zhi et al. 2015; Bangcharoensap et al. 2016; Faleiros et al. 2016; Luo et al. 2018; Ding et al. 2019).

All generated networks were classified using GNetMine, and metrics for Precision, Recall, F-score (macro variant), Accuracy, and classification time. The proposed coarsening algorithm was applied with a reduction of 0–80% at increments of 5%. The transductive classification evaluation was performed by varying the number of labeled vertices in

1%, 10%, 20%, and 50% of vertices from the target partition. The results were used as a comparative reference for classification metrics and network storage size.

Experiment 1

The results of the first analysis of the reduction's main objectives (memory savings and classification runtime) are illustrated in Fig. 9. The diagrams show the evolution of memory size and transductive classification runtime when varying the number of vertices. Table 1 reveals the relative economy of these two metrics for each reduction level compared to the original unreduced network. Notably, there is a more significant relative economy in the early stages of coarsening, resulting in satisfactory results with only a 20% decrease in the number of vertices. This observation could be attributed to the initial levels already clustering many vertices with more shared connections.

Memory economies were analyzed concerning the dispersion parameter. As shown in Fig. 10, higher edge densities result in more expensive storage and classification runtime networks. As a result, coarsening is more effective in such cases, leading to more significant gains.

Experiments were conducted to assess the impacts of network reduction on classification F-score. The experiment initially involved varying the dispersion of the networks, and the results are described in Fig. 11. The results showed that low dispersion levels resulted in low memory savings but had minimal impact on the classification metrics.

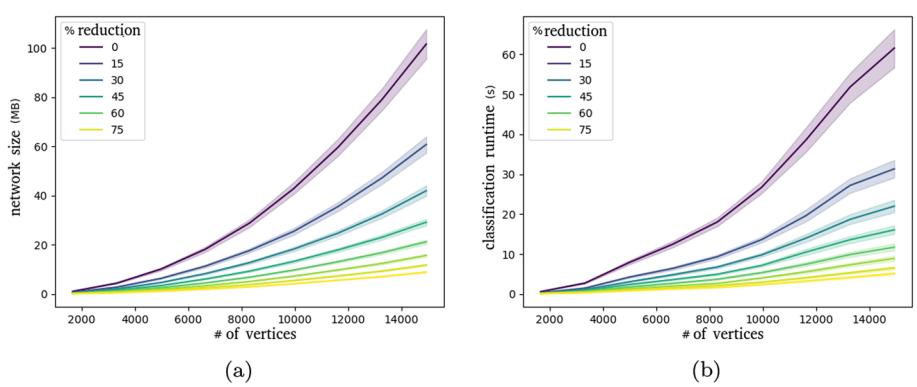

Fig. 9 Variations in the size of the networks in MB (**a**) and the time required to classify them (**b**) with the increase in the number of vertices in the networks

Table 1 Compared to their original forms, reducing networks can lead to savings in both storage size and classification time

	Storage savings			Time savings for classification		
# of vertices Reduction	**2000**	**Mean**	**15, 000**	**2000**	**Mean**	**15, 000**
20%	32.87 ± 0.48%	36.92 ± 0.47%	38.75 ± 0.39%	39.76 ± 0.84%	45.35 ± 0.89%	47.19 ± 0.82%
35%	47.59 ± 0.73%	53.43 ± 0.67%	56.92 ± 0.47%	50.61 ± 0.89%	59.03 ± 0.94%	62.41 ± 0.71%
50%	59.88 ± 0.80%	65.29 ± 0.71%	69.51 ± 0.47%	59.39 ± 0.82%	68.57 ± 0.87%	72.09 ± 0.66%
60%	68.78 ± 0.79%	73.75 ± 0.67%	77.46 ± 0.45%	65.86 ± 0.86%	75.88 ± 0.81%	79.43 ± 0.56%
80%	84.41 ± 0.54%	88.12 ± 0.43%	90.11 ± 0.30%	79.30 ± 0.75%	88.12 ± 0.59%	90.73 ± 0.33%

For both metrics, the values are represented as percentages

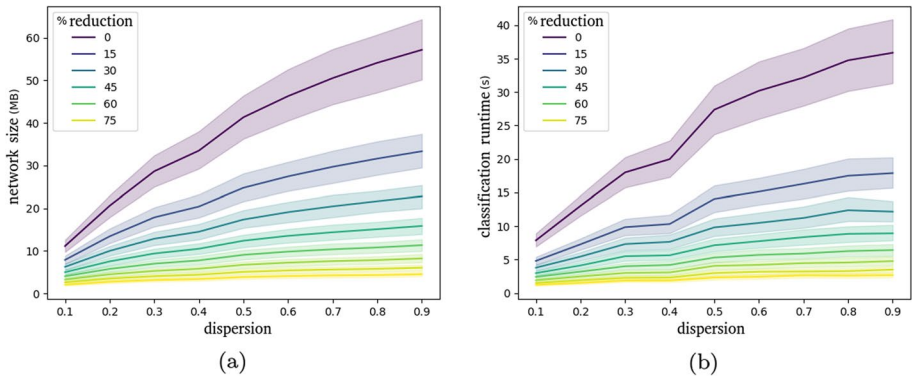

Fig. 10 Variations in the size of networks in MB (**a**) and in the time required to classify them (**b**) with the change in edge density

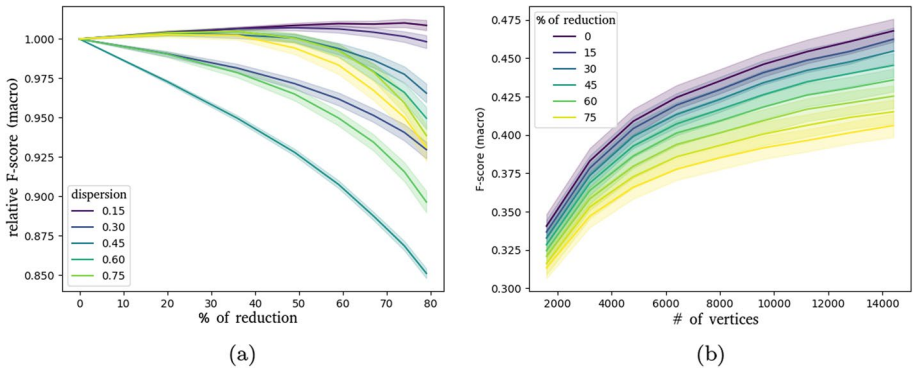

Fig. 11 Variations in F-score (macro) metric (**a**) as the network is reduced for different levels of dispersion. And the same metric as the size of the input network increases (**b**)

Table 2 summarizes the relationship between network reduction and classification metrics. The results show that as the networks grow, there is an increase in the loss caused by coarsening. For a network with 2000 vertices, the difference between reducing by 20% and 80% resulted in a relative F-score loss of 5%. This relative F-score difference is amplified to around 10% in networks with 15, 000 vertices. The results also reveal two essential facts. Firstly, the Precision metric appears relatively stable, indicating that the proposed coarsening algorithm is more suitable for applications that aim to reduce the number of false positives. Secondly, the low relative F-score loss achieved by reducing the dataset by about 20% is noteworthy, especially considering the significant savings in storage and classification runtime.

Table 2 Increased loss with the reduction of networks accentuated by the larger size of the networks (comparison between 2000 and 15, 000 vertices)

# of vertices	Reduction	Accuracy	Precision (macro)	Recall (macro)	F-score (macro)
2000	20%	1.01 ± 0.32%	0.76 ± 0.51%	1.08 ± 0.34%	0.92 ± 0.47%
	80%	6.20 ± 1.12%	2.54 ± 1.05%	6.49 ± 1.15%	6.12 ± 1.42%
15,000	20%	1.50 ± 0.52%	0.57 ± 0.40%	1.51 ± 0.53%	1.69 ± 0.76%
	80%	10.48 ± 1.73%	0.63 ± 0.82%	10.63 ± 1.75%	11.97 ± 2.24%

Experiments 2

We performed a complete analysis ranging from 2 thousand to 100 thousand vertices to determine whether the results would remain consistent for larger k-partite networks. The results are described in Fig. 12 and Table 3.

The data collected in this experiment confirm the effectiveness of the proposed coarsening algorithm for reductions of around 20%, especially in the Precision metric, obtaining an average variation of only $0.55 \pm 0.17\%$ compared to the original graph. The Accuracy, Recall, and F-score metrics variations were $1.72 \pm 0.54\%$, $1.78 \pm 0.55\%$, and $2.36 \pm 0.77\%$, respectively, suggesting some degree of information redundancy in the network's topology. These findings suggest that coarsening is an effective technique for reducing storage and processing resources.

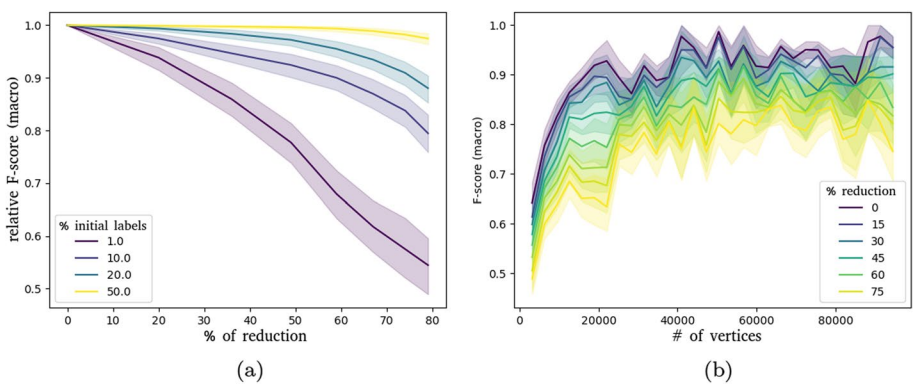

Fig. 12 Variations in F-score (macro) metric (**a**) as the network is reduced, for different percentages of initial labels given. And the same metric as the size of the input network increases from 2 to 100 thousand vertices (**b**)

Table 3 Complete experimental data for the smallest (2000) and largest (100,000) number of vertices

# of vert.	Reduct.	Accuracy	Precision (macro)	Recall (macro)	F-score (macro)
2000	20%	$2.55 \pm 0.40\%$	$0.76 \pm 0.14\%$	$3.10 \pm 0.46\%$	$3.85 \pm 0.62\%$
	35%	$4.07 \pm 0.56\%$	$1.27 \pm 0.18\%$	$4.85 \pm 0.63\%$	$5.73 \pm 0.84\%$
	50%	$6.22 \pm 0.77\%$	$1.81 \pm 0.24\%$	$7.31 \pm 0.86\%$	$8.31 \pm 1.05\%$
	60%	$8.69 \pm 1.00\%$	$2.48 \pm 0.31\%$	$10.09 \pm 1.10\%$	$10.89 \pm 1.23\%$
	80%	$15.86 \pm 1.61\%$	$4.02 \pm 0.40\%$	$18.31 \pm 1.82\%$	$19.32 \pm 1.92\%$
100,000	20%	$3.25 \pm 1.13\%$	$1.08 \pm 0.38\%$	$3.25 \pm 1.13\%$	$3.92 \pm 1.36\%$
	35%	$4.79 \pm 1.16\%$	$1.55 \pm 0.41\%$	$4.90 \pm 1.17\%$	$5.77 \pm 1.45\%$
	50%	$11.25 \pm 1.80\%$	$3.24 \pm 0.51\%$	$11.06 \pm 1.78\%$	$14.24 \pm 2.27\%$
	60%	$12.67 \pm 1.93\%$	$3.59 \pm 0.57\%$	$12.56 \pm 1.93\%$	$15.91 \pm 2.41\%$
	80%	$19.81 \pm 2.58\%$	$5.24 \pm 0.61\%$	$19.65 \pm 2.56\%$	$23.22 \pm 2.97\%$

Table 4 Distribution of vertices utilized in the experiment, sourced from the DBLP dataset

Vertice type	Quantity
Authors ($\mathcal{V}_A$)	14,475
Articles ($\mathcal{V}_P$)	14,376
Conferences ($\mathcal{V}_C$)	20
Terms ($\mathcal{V}_T$)	8920
Total	37,791

Experiments with real data

This section presents the results of the proposed coarsening algorithm on a real-world heterogeneous network. The DBLP dataset[1] was used for the classification task, and the GNetMine algorithm was employed with the same parameter set as in the experiments of Section 4.3. The DBLP dataset contains open bibliographic information from major computer science journals and proceedings. The dataset description is outlined in Table 4.

The problem addressed using the DBLP dataset involves classifying authors into four areas of knowledge. The authors serve as the target partition. The non-target partitions include the articles written by each author, the conferences where the authors have published, and the terms found in these articles. This problem can be mathematically modeled as a tuple $G_{DBLP} = (V, E)$ with partitions $V = \{\mathcal{V}_P, \mathcal{V}_A, \mathcal{V}_C, \mathcal{V}_T\}$ representing articles, authors, conferences, and terms respectively. The schema of G_{DBLP}, $S(G_{DBLP})$, has the partition $\mathcal{V}_A$ serving as the target for the classification task. There is a set of four classes, $C = DataMining, Database, Information\ Retrieval, Machine\ Learning$, which represent research areas. Additionally, a set $\mathcal{V}_A^L \subset \mathcal{V}_A$ of authors have already been labeled with one of the areas in C. The network G_{DBLP} has a low number of edges, totaling 170, 795 edges, and a dispersion of approximately 0.002.

The proposed coarsening algorithm was applied to the real network G_{DBLP}. As shown in Table 5, there was no significant decrease in classification metrics, even with a 67% reduction in the network. However, the coarsening did not effectively reduce the storage size of the network, as observed in Table 6. This result corresponds to the observations made in the synthetic experimental evaluation. Notably, the network G_{DBLP} has a low dispersion level of about 0.002. As previously discussed, low dispersion levels (near zero) result in minimal loss of classification quality (as seen in Fig. 11) and low storage savings (as observed in Fig. 10a).

Concluding remarks

This study aimed to test a proposed algorithm for reducing the size of k-partite networks while maintaining classification performance. The algorithm was applied on synthetic k-partite networks with varying characteristics to evaluate its effectiveness in improving scalability and storage efficiency. Existing techniques for network reduction have primarily been tested on homogeneous networks (Chen et al. 2017; Liang et al. 2020), making this study a significant contribution to the field. The study obtained metrics

[1] DBLP dataset available at https://dblp.org.

Table 5 Metrics of accuracy, precision (macro), recall (macro) and F-score (macro) according to G_{DBLP} reduction

Reduction	Accuracy	Precision (macro)	Recall (macro)	F-score (macro)
20%	2.47%	0.02%	2.59%	1.25%
36%	4.88%	0.11%	4.46%	2.46%
49%	4.92%	1.21%	5.12%	3.75%
59%	4.92%	2.43%	5.02%	3.75%
67%	4.92%	1.91%	5.02%	3.75%

Table 6 Storage and time savings analysis for G_{DBLP}

Reduction	Storage savings (MB)	Time savings for classification (s)
0%	2476.87	21.15
20%	2219.46	14.40
36%	2054.75	11.84
49%	2014.74	11.22
59%	2010.69	11.10
67%	2010.24	11.12

related to resource savings and classification performance and validated the effectiveness of the proposed coarsening algorithm for k-partite networks.

This study has some limitations that should be described. Firstly, using synthetic networks instead of real-world networks may introduce bias. It is worth noting that the HNOC tool generates networks with a high level of assortativity, which may only be present in some real-world networks. Additionally, the number of k-partite network schemes used in the experiments was limited, which is a potential limitation. However, based on the various experiments conducted and different parameter configurations tested, our proposed technique demonstrates the promising potential for application in diverse networks. The entire source code used in the experiments is made available,[2] which enables future works to test other parameters and networks.

According to the findings, the proposed coarsening algorithm effectively achieved considerable savings in storage and classification runtime, even when the reduction levels were modest. For instance, a 20% reduction in the number of vertices resulted in over 1/3 savings in storage and twice faster classifications. Additionally, the classification performance metrics had low average levels of variation.

Acknowledgements

This research was funded by FAPDF (Grant 07/2019) and funded by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001 and The State of São Paulo Research Foundation (FAPESP) under Grant Number: 22/03090-0 and 21/06210-3.

Author contributions

Conceptualization: PEA, ADBV, and TPF provide the main idea of the proposed method. Methodology: the models, methodology, and experiments were designed by PEA and ADBV. Validation: the accuracy of results was checked by TPF. Software: PEA implemented the methods and carried out the experiments. Writing-original draft: the original draft was prepared initially by TPF. Visualization: PEA provided all the figures and conceptually checked by ADBV and TPF. Supervision: the whole project was supervised by ADBV and TPF. Proofread: the paper documents were proofread by ADBV and TPF. All authors read and approved the final manuscript.

[2] https://github.com/pealthoff/CoarseKlass.

Availability of data and materials
The code for the experiments and the generated data sets can be accessed on https://github.com/pealthoff/CoarseKlass.

Declarations

Competing interests
The authors declare that they have no competing interests.

Received: 24 May 2023 Accepted: 21 November 2023
Published online: 05 December 2023

References

Bangcharoensap P, Murata T, Kobayashi H, Shimizu N (2016) Transductive classification on heterogeneous information networks with edge betweenness-based normalization. In: Proceedings of the ninth ACM international conference on web search and data mining

Blasi M, Freudenreich M, Horvath J, Richerby D, Scherp A (2022) Graph summarization with graph neural networks. arXiv:2203.05919

Chen H, Perozzi B, Hu Y, Skiena S (2017) HARP: hierarchical representation learning for networks. CoRR arXiv:abs/1706.07845

Ding P, Shen C, Lai Z, Liang C, Li G, Luo J (2019) Incorporating multisource knowledge to predict drug synergy based on graph co-regularization. J Chem Inf Model. https://doi.org/10.1021/acs.jcim.9b00793

Faleiros T, Rossi R, Lopes A (2016) Optimizing the class information divergence for transductive classification of texts using propagation in bipartite graphs. Pattern Recognit Lett. https://doi.org/10.1016/j.patrec.2016.04.006

Gupta M, Kumar P, Bhasker B (2017) HeteClass: a meta-path based framework for transductive classification of objects in heterogeneous information networks. Expert Syst Appl 68:106–122. https://doi.org/10.1016/j.eswa.2016.10.013

Ji M, Sun Y, Danilevsky M, Han J, Gao J (2010) Graph regularized transductive classification on heterogeneous information networks. In: Balcázar JL, Bonchi F, Gionis A, Sebag M (eds) Machine learning and knowledge discovery in databases. Springer, Berlin, pp 570–586

Lagraa S, Seba H, Khennoufa R, Maya A, Kheddouci H (2014) A distance measure for large graphs based on prime graphs. Pattern Recognit 47(9):2993–3005. https://doi.org/10.1016/j.patcog.2014.03.014

LeFevre K, Terzi E (2010) Grass: graph structure summarization. In: Tenth SIAM international conference on data mining (SDM), pp 454–465

Liang J, Gurukar S, Parthasarathy S (2020) MILE: a multi-level framework for scalable graph embedding. arXiv:1802.09612

Liu Y, Safavi T, Dighe A, Koutra D (2018) Graph summarization methods and applications: a survey. ACM Comput Surv. https://doi.org/10.1145/3186727. arXiv:1612.04883

Luo C, Guan R, Wang Z, Lin C (2014) HetPathMine: a novel transductive classification algorithm on heterogeneous information networks. In: LNCS, vol 8416, pp 210–221. https://doi.org/10.1007/978-3-319-06028-6_18

Luo J, Ding P, Liang C, Chen X (2018) Semi-supervised prediction of human miRNA-disease association based on graph regularization framework in heterogeneous networks. Neurocomputing 294:29–38. https://doi.org/10.1016/j.neucom.2018.03.003

Raghavan N, Albert R, Kumara S (2007) Near linear time algorithm to detect community structures in large-scale networks. Phys Rev E Stat Nonlinear Soft Matter Phys 76:036106

Redmond S, Rozaki E (2017) Using bipartite graphs projected onto two dimensions for text classification. Int J Adv Comput Sci Its Appl. https://doi.org/10.15224/978-1-63248-131-3-19

Riondato M, García-Soriano D, Bonchi F (2014) Graph summarization with quality guarantees. In: 2014 IEEE international conference on data mining, pp. 947–952. https://doi.org/10.1109/ICDM.2014.56

Rossi RG, de Paulo Faleiros T, de Andrade Lopes A, Rezende SO (2012) Inductive model generation for text categorization using a bipartite heterogeneous network. In: 2012 IEEE 12th international conference on data mining, pp 1086–1091. https://doi.org/10.1109/ICDM.2012.130

Valejo A, Lopes AA, Filho GPR, Oliveira MCF, Ferreira V (2017a) One-mode projection-based multilevel approach for community detection in bipartite networks. In: International symposium on information management and big data (SIMBig), track on social network and media analysis and mining (SNMAN), pp 101–108

Valejo A, Ferreira V, Oliveira MCF, Lopes AA (2017b) Community detection in bipartite network: a modified coarsening approach. In: International symposium on information management and big data (SIMBig), track on SNMAN. Communications in computer and information science book series (CCIS, volume 795), pp 123–136

Valejo A, Ferreira de Oliveira MC, Filho GPR, de Andrade Lopes A (2018) Multilevel approach for combinatorial optimization in bipartite network. Knowl Based Syst 151:45–61. https://doi.org/10.1016/j.knosys.2018.03.021

Valejo A, Faleiros T, de Oliveira MCF, de Andrade Lopes A (2020a) A coarsening method for bipartite networks via weight-constrained label propagation. Knowl Based Syst 195:105678. https://doi.org/10.1016/j.knosys.2020.105678

Valejo A, Ferreira V, Fabbri R, Oliveira MCRF, Lopes A (2020b) A critical survey of the multilevel method in complex networks. ACM Comput Surv 53(2):35

Valejo A, Góes F, Romanetto L, Ferreira de Oliveira MC, de Andrade Lopes A (2020c) A benchmarking tool for the generation of bipartite network models with overlapping communities. Knowl Inf Syst 62(4):1641–1669. https://doi.org/10.1007/s10115-019-01411-9

Valejo A, Althoff P, Faleiros T, Chuerubim M, Yan J, Liu W, Zhao L (2021) Coarsening algorithm via semi-synchronous label propagation for bipartite networks. In: Anais da X Brazilian conference on intelligent systems. SBC, Porto Alegre, RS, Brasil. https://sol.sbc.org.br/index.php/bracis/article/view/19047

van Engelen JE, Hoos HH (2020) A survey on semi-supervised learning. Mach Learn 109(2):373–440. https://doi.org/10.1007/s10994-019-05855-6

Walshaw C (2004) Multilevel refinement for combinatorial optimisation problems. Ann Oper Res 131(1):325–372. https://doi.org/10.1023/B:ANOR.0000039525.80601.15

Wu Z, Pan S, Chen F, Long G, Zhang C, Yu PS (2021) A comprehensive survey on graph neural networks. IEEE Trans Neural Netw Learn Syst 32(1):4–24. https://doi.org/10.1109/TNNLS.2020.2978386

Zhi S, Han J, Gu Q (2015) Robust classification of information networks by consistent graph learning. In: Appice A, Rodrigues PP, Santos Costa V, Gama J, Jorge A, Soares C (eds) Machine learning and knowledge discovery in databases. Springer, Cham, pp 752–767

Zhou J, Cui G, Hu S, Zhang Z, Yang C, Liu Z, Wang L, Li C, Sun M (2020) Graph neural networks: a review of methods and applications. AI Open 1:57–81. https://doi.org/10.1016/j.aiopen.2021.01.001. arXiv:1812.08434

Zhu L, Ghasemi-Gol M, Szekely P, Galstyan A, Knoblock CA (2016) Unsupervised entity resolution on multi-type graphs. In: Groth P, Simperl E, Gray A, Sabou M, Krötzsch M, Lecue F, Flöck F, Gil Y (eds) The semantic web—ISWC 2016. Springer, Cham, pp 649–667

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Submit your manuscript to a SpringerOpen journal and benefit from:

- ► Convenient online submission
- ► Rigorous peer review
- ► Open access: articles freely available online
- ► High visibility within the field
- ► Retaining the copyright to your article

Submit your next manuscript at ► springeropen.com