Scalable Network Embedding with Approximate Equitable Partitions

Giuseppe Squillace, Mirco Tribastone, Max Tschaikowski, and Andrea Vandin

***Abstract*—Network embedding is a fundamental technique to project a network into a lower-dimensional space while preserving similarities among nodes. Traditional network embeddings primarily capture node proximity, making them effective for community detection but insufficient for identifying roles, i.e., patterns of interaction beyond local neighborhoods. To address this limitation, we introduce a simple and efficient embedding technique based on approximate variants of equitable partitions. Our approach, called ε-BE, introduces a user-tunable tolerance parameter relaxing the otherwise strict condition for exact equitable partitions that can be hardly found in real-world networks. We exploit a relationship between equitable partitions and equivalence relations for Markov chains and ordinary differential equations to develop a partition refinement algorithm for computing an approximate equitable partition in polynomial time. We extend this framework to weighted and directed networks, ensuring applicability to a more general class of graphs and filling a gap in the literature where few approaches are present. We compare our method against state-of-the-art embedding techniques on synthetic and real-world networks. We report comparable—when not superior—performance for visualization, classification, clustering, and regression tasks with smaller running times, enabling the embedding of large-scale networks that could not be efficiently handled by most of the competing techniques. These results and the capability to handle weighted and directed networks make our approach a compelling alternative for structural network embedding.**

***Index Terms*—Equitable partitions, network embedding, backward equivalence, structural equivalence**

I. Introduction

NETWORK analysis is crucial in various fields, including, e.g., social networks [1], bioinformatics [2], and recommendation systems [3]. The increasing availability of data leads to large networks with complex relationships characterized by edges with specific weights and directionality. One of the main challenges is to provide methods to effectively interpret and analyze these networks in a reasonable amount of time. Network embedding tackles this problem by extracting properties of a network and encoding them in a lower-dimensional space [4]. The main objective is to speed up the analysis while preserving the similarity of nodes in the embedded space [5].

Traditionally, the notion of similarity between nodes is based on network proximity. This approach is well-suited for community detection but is not effective in capturing the *role* of a node in the network [6], i.e., patterns of interaction beyond simple local proximity. The notion of role, also known as position, is well studied in the social science literature (e.g., [7]). Roles partition nodes into classes according to some criteria such as structural [8], automorphic [9], and regular equivalence [10]. Roughly speaking, in structural equivalence, two nodes have the same role if they interact with the same set of nodes. In automorphic equivalence, two nodes are equivalent if we can define an automorphism between them. Finally, two nodes are regular equivalent if they interact with the same variety of role classes (cf. Figure 1).

Structural equivalence is strict because it requires that related nodes interact with *the same* set of nodes; on the other hand, automorphic equivalence suffers from well-known difficulties in its computation. Here, we propose a structural embedding method based on equitable partitions [12], also known as exact regular coloration [13] and graph divisor [14]. This notion was originally proposed for undirected networks and states that nodes with the same role have the same number of neighbors for each role. It is a relaxation of automorphic equivalence [11], which can be computed efficiently in polynomial time in the size of the network [13]. It is stricter than regular equivalence because regular equivalence does not count the number of neighbors with the same role [8]. However, since it requires exact matching of neighbor counts, an equitable partition can still be too restrictive for real networks.

We develop a novel approach to compute a network embedding inspired by an approximate variant of equitable partitions that accepts differences in the number of neighbors by means of a parameter ε that, informally, determines the tolerance level for identifying similar nodes. We anticipate the results of our method on a simple undirected network depicted in Figures 2 and 3. Figure 2 shows the equitable partition where node 2 is different from nodes 3 and 4 because it has a different degree. Our approximate variant, with $\varepsilon = 1$ accepts differences up to one node: the result is shown in Figure 3, where nodes 2,

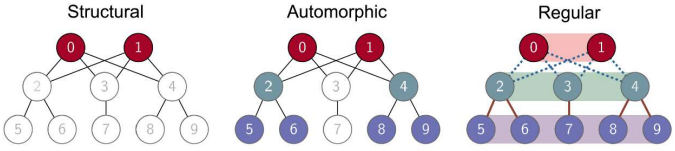


Fig. 1: Role equivalences (example from [11]). Nodes with the same color indicate the same role; white nodes have pairwise distinct roles.

Giuseppe Squillace and Mirco Tribastone are with IMT Lucca, Italy (e-mail: {giuseppe.squillace, mirco.tribastone}@imtlucca.it).

Max Tschaikowski is with Aalborg University, Denmark (e-mail: tschaikowski@cs.aau.dk

Andrea Vandin is with Sant'Anna Pisa, Italy (e-mail: andrea.vandin@santannapisa.it)

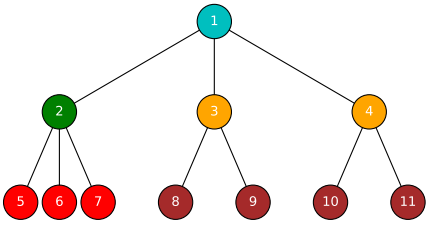


Fig. 2: Running example with equitable partitions.

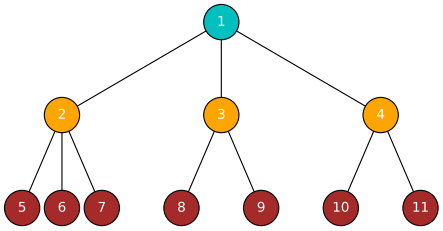


Fig. 3: Approximate variant of equitable partitions computed with ε-BE on the running example from Figure 2 with $\varepsilon = 1$.

Nodes	Embedding
{1}	[0,1,2,0,0]
{2}	[1,0,0,3,0]
{3,4}	[1,0,0,0,2]
{5,...,7}	[0,1,0,0,0]
{8,...,11}	[0,0,1,0,0]

Nodes	Embedding
{1}	[0,3,0]
{2}	[1,0,3]
{3,4}	[1,0,2]
{5,...,11}	[0,1,0]

TABLE I: Node representation in the embedding space associated with the equitable partition of Figure 2 (left) and Figure 3 (right). The columns correspond to the blocks of the partitions sorted by their length.

3, and 4 (as well as their neighbors in the bottom layer) are identified to be approximately equivalent, thus capturing their intuitive structural similarity in the network.

Our starting point is the observation that, for undirected networks, the equitable partition definition corresponds to an equivalence relation known as backward equivalence (BE) [15]. BE extends exact lumpability [16], an aggregation method for Markov chains based on properties of its generator matrix, to arbitrary adjacency matrices [17] and nonlinear models [18]–[20]. Our approximate method, which we call ε-BE, relaxes the equivalence criterion by means of the tolerance parameter ε.

We extended the applicability of our approach to more expressive networks that present weights and directionality in their connections. In this sense, we are proposing a new method that indirectly extends the concept of equitable partitions both for directed and weighted networks. More in detail, for weighted networks, we note that the definition can easily be extended considering the tolerance ε as the difference between the sum of the weights from equivalent classes. On the other hand, for directed networks, we compute two different partitions, one accounting for the out-degree and the other for the in-degree.

A crucial advantage of BE, inherited from results available for Markov chains, is the availability of an efficient algorithm for computing it that runs in $O(m \log n)$ time [21], where m is the number of edges and n is the number of nodes. The algorithm is based on partition refinement [22]: given an initial partition of nodes, where each block represents a candidate role, it iteratively splits blocks until the BE criterion is met. In this paper, we develop an analogous algorithm for ε-BE, still based on [21]. Consequently, exploiting the partition refinement nature of the algorithm, we propose an iterative algorithm called Iterative ε-BE (Iε-BE) that, as an inner step, computes an ε-BE partition with increasing values of ε, keeping track of the positions discovered at previous iterations. As will be explained later in the paper, this avoids the degenerate cases where the algorithm may aggregate too little or too much when using a single-shot strategy with small or large enough, respectively, values of ε.

The resulting partition is employed to build the network embedding E. We are interested in producing a network embedding $E \in \mathbb{R}^{n \times d}$ where n is the number of nodes, and d is the size of the embedding. This can be built by summing up separately the edge weights toward equivalent classes for each node. Table I shows the embeddings respectively for the two partitions of Figures 2 and 3. In particular, nodes with the same role exhibit similar vectors in the network embedding and the ε-BE condition can be checked by computing the difference of the embedding vectors. For example, consider the nodes 2 and 3 of Table I (right). The difference of their embedding vectors is [0,0,1]. Each entry of this vector is $\leq \varepsilon$, and, for this reason, they present the same color in Figure 3.

To compare Iε-BE against state of the art, we consider the best-performing methods to network embedding covering all three aforementioned categories presented in the recent review [4]. We compared the network embeddings across various machine-learning tasks such as classification, clustering, visualization, and regression on real-world networks.

Overall, we show that Iε-BE is able to provide competitive results, outperforming the other methods, with runtimes at least one order of magnitude faster. For larger networks, in particular, Iε-BE provided node embeddings in a few minutes while most of the other methods timed out after 3 hours. Notably, Iε-BE is the only method among the few able to compute efficiently and effectively network embeddings for weighted and directed networks. This promotes ε-BE as a valid alternative for the structural network embedding.

A. *Related works*

We begin by recalling that the technique we propose focuses on role-based network embedding, which differs fundamentally from traditional proximity-based network embedding methods [23]–[27]. While proximity-based techniques aim to embed nodes that are close within the network, the role network embedding targets nodes that play similar structural roles, regardless of their actual distance in the graph. Recently, some works [28]–[31] address the classification task in both heterophilic and homophilic networks using GNNs. In homophilic networks, nodes with similar labels are more likely to be connected, a setting that aligns naturally with

community detection and proximity-based models. In contrast, heterophilic networks are characterized by connections between nodes with different labels. While heterophily is relevant in classification tasks, it is not directly aligned with the notion of structural roles. Role detection focuses on identifying nodes that share similar patterns of interaction with other types of nodes, regardless of whether they are similar or not. Therefore, a GNN that performs well in heterophilic settings does not necessarily capture role-based similarities. Some partition-based frameworks [32], [33] have been proposed, but these are not role-oriented embedding methods. They serve either as initialization schemes or distributed computation frameworks for existing community-preserving algorithms.

More broadly, role-based embedding remains a more complex and challenging direction. One key reason is that the concept of role is deeply connected to graph automorphism, a classic problem in theoretical computer science known for its computational hardness [11].

Methods for role-based structural network embeddings can be categorized into matrix factorization, random walk, and deep learning methods.

Undirected networks: Matrix factorization methods capture structural properties by decomposing the adjacency matrix or other related matrices into lower-dimensional matrices. The most closely related approach to ours is RIDεRs [34]. It is also based on a notion of ε-equitable refinement. However, the embedding method is substantially different. Indeed, instead of the iterative scheme proposed in this paper, RIDεRs computes an embedding E_ε for each value of ε ranging from 0 to the average degree of the network d_{avg}. Since the overall dimension of the embedding can be large, a feature pruning technique is employed to reduce this dimension. Subsequently, RIDεRs aggregates the pruned embedding into a single matrix $E = [E_0|...|E_{d_{avg}}]$ and employs negative matrix factorization (NMF [35]) on E to generate the final embedding.

RolX [36] factorizes the network into a role membership matrix and a role-feature matrix. The matrix factorization is optimized by minimizing the reconstruction error. GLRD extends RolX by adding further constraints in the previous optimization problem [37]. GraphWave is based on a physics interpretation [38]. It considers the application of the heat diffusion process on the network. Then it computes the graph diffusion kernel and gets the embedding using the characteristic function. xNetMF computes a similarity matrix between nodes, and then it factorizes this matrix [39]. SEGK uses graph kernels to compute node structural similarities [40]. The standard one is the shortest path kernel, which uses the Nystrom method [41] for factorization. SPaE provides a structural embedding via the Laplacian eigenmaps method [42]. Approaches based on random walks explore the networks with paths of a given length. According to the taxonomy in [4], network embeddings are usually learned using methods from natural language processing. One popular technique is struc2vec [43]. It generates a multi-layer weighted context graph, and then it employs SkipGram [44] to learn the embedding. Role2vec assigns to each node a role and then builds a random walk where the generated sequence is replaced with role indicators [45]. NODE2BITS employs temporal random walks where the edges are sampled with non-decreasing timestamps [46]. The last category learns the embedding through deep learning techniques such as neural networks and graph neural networks. Among these, we mention DRNE [47]. It produces a network embedding based on the regular equivalence definition. The method is based on training a long-short-term memory (LSTM) neural network. GAS employs GNNs exploiting the Weisfeiler-Lehman test [48]. Other works in this direction are GraLSP [49], which incorporates random walks in the GNN, and GCC [50], which uses a pre-trained Graph Neural Network model.

Recently, in [51], the authors presented an optimization-based method to compute approximate equitable partitions. The emphasis is on partition computation, with only limited attention to embeddings. In contrast, our approach develops role-oriented embeddings and provides a broader experimental evaluation across downstream tasks and scalability analysis. Complementary to this line, our recent study [52] focused on approximate regular equivalence rather than equitable partitions, relying on the ε-BDE variant originally developed for dynamical systems. That line of work concentrated exclusively on partition computation, without deriving node embeddings.

Weighted and directed networks: While in network embedding based on proximity, there are several methods for weighted and directed networks [53]–[55], in the role network embedding, few approaches were proposed. Among the previously considered methods, DRNE is also suitable for unweighted directed networks, while Graphwave can handle weighted undirected networks. Additionally, RIDεRs formally is suitable for weighted and directed networks, but the actual working implementation does not support them.

To the best of our knowledge, the only method in literature with an accessible implementation and the ability to handle weighted and directed networks is EMBER [46]. EMBER is a matrix factorization method that extends xNetMF and was originally developed for mining professional roles in email networks. It builds the feature matrices for outgoing and incoming edges that compute the product of all edge weights in the k-step shortest outgoing/incoming paths. Then, similarly to our approach, these matrices are concatenated in a single embedding matrix.

Relationship with [56]: This work builds upon our previous conference paper [56], extending the theoretical framework of Iε-BE to support both directed and weighted networks. In addition to the theoretical extension, we have significantly strengthened the experimental evaluation. Specifically, compared to the conference version, we broadened the experimental evaluation by adding a new visualization study on a real-world network, a clustering task, a sensitivity analysis, and an extended scalability study on networks with up to 1 million nodes. For the directed and weighted setting, we incorporate all the tasks already present in the conference version, including classification experiments on synthetic networks as well as a detailed scalability analysis.

II. Background

We consider a graph with a set of nodes V denoted by an adjacency matrix $A = (a_{ij}) \in \mathbb{R}^{n\times n}$. An entry with a

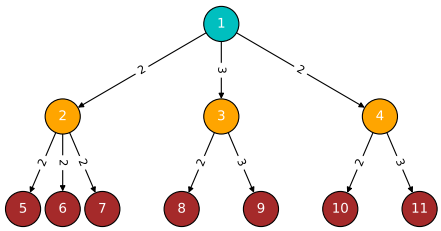


Fig. 4: 1-BE partition computed on the adjacency matrix A of the directed, weighted version of the network presented in the introduction. The partition obtained from A^T is equivalent.

value > 0 indicates the presence of an edge from node i to node j with weight a_{ij}, while $a_{ij} = 0$ signifies the absence of an edge. The graph is unweighted when the adjacency matrix A assumes only values in the set $\{0, 1\}$, otherwise, it is weighted. Furthermore, the graph is undirected if A is a symmetric matrix, meaning that an edge from i to j implies an edge from j to i. In contrast, when this condition is not satisfied, the graph is directed.

Network embedding consists in finding a matrix $E \in \mathbb{R}^{n\times d}$ where $d < n$. We are interested in an embedding that preserves the similarities between nodes with similar roles in the network.

We present the definition of BE. Originally developed for hypergraphs [15], here we provide a simplified version for graphs. Informally, a BE is a partition of the nodes such that nodes belonging to the same block present the same cumulative weighted in-degrees from equivalence classes.

Definition 1 (Backward equivalence, simplified to graphs from [15])**.** *For an adjacency matrix $A \in \mathbb{R}^{n\times n}$, a partition of the nodes $\mathcal{H}$ is called a backward equivalence (BE) when*

$$\sum_{k\in B'} a_{ki} = \sum_{k\in B'} a_{kj}$$

for all blocks $B, B' \in \mathcal{H}$ and $i, j \in B$.

For undirected networks, the definition also assures that nodes in the same block present the same cumulative weighted out-degrees toward equivalence classes. Indeed, when simplified to unweighted undirected graphs, BE is equivalent to the equitable partition definition presented in [34], as is apparent from comparing the previous and the forthcoming definitions.

Definition 2 (Equitable partition, adapted from [34])**.** *For a network with nodes V, and edges E, a partition $\mathcal{H}$ of V is an equitable partition if and only if*

$$deg(v_i, B') = deg(v_j, B')$$

for all blocks $B, B' \in \mathcal{H}$, and $v_i, v_j \in B$, where

$$deg(v_l, B'') = |\{v_k | (v_l, v_k) \in E \text{ and } v_k \in B''\}|$$

The colors in Figure 2 identify the different blocks of a BE/equitable partition. We now relax the notion of BE to account for tolerances in node degrees. Here, we consider the tolerance ε as a measure to accept approximate and not only exact equitable partitions.

Definition 3 (ε-BE)**.** *For an adjacency matrix $A \in \mathbb{R}^{n\times n}$, a partition of the nodes $\mathcal{H}$ is called ε-BE partition when*

$$\left| \sum_{k\in B'} a_{ki} - \sum_{k\in B'} a_{kj} \right| \leq \varepsilon$$

for all blocks $B, B' \in \mathcal{H}$ and $i, j \in B$.

We can see that ε-BE corresponds to BE for $\varepsilon = 0$. However, by increasing ε, two nodes can be related even if they have similar, but not exact, degrees from (approximately) equivalent blocks. The parameter ε tunes the level of similarity of the nodes in the same block. For instance, setting $\varepsilon = 1$ allows larger identifications of similar states in Figure 3 as opposed to the equitable partition in Figure 2.

This definition is inspired by a technique to reduce dynamical systems called ε-BDE [57]. The main difference between the two approaches lies in the use of the tolerance ε. In ε-BDE, the cumulative difference in the sum of the weights towards all equivalent blocks should be less than ε. Instead, ε-BE is less strict as it requires the difference from each block to be smaller than ε. With this change, it turns out that, for undirected networks, ε-BE corresponds to the notion of approximate equitable partition from [34].

We extend ε-BE to account for the weights and directionality of the edges, providing an extension of the definition of equitable partition for directed and weighted networks. In Figure 4, we show the partition computed on a weighted directed version of the running example presented in the introduction. In this case, the method assures that the difference in the sums of the weights from equivalent classes is less or equal than ε for each node. Additionally, due to the asymmetric nature of the adjacency matrix, we compute a partition with respect to A^T and a partition with respect to A. The former checks the condition on the outgoing edges, while the latter on the incoming edges. For our running example, the partition on A^T and A is equivalent and corresponds to the one depicted in Figure 4. We show in the experimental section that this is not always the case and that, in general, the partitions on A^T and A are different.

We build a node embedding using the partitions computed by ε-BE as follows.

Definition 4 (ε-BE embedding)**.** *Let us consider the ε-BE partitions $\mathcal{H} = \{B_1, ..., B_h\}$ with $h < n$ and $\mathcal{H}_T = \{B'_1, ..., B'_c\}$ with $c < n$ computed respectively on adjacency matrices A^T and A. We define the network embedding matrix $E \in \mathbb{R}^{n\times d}$ as follow:*

$$E = [E^+ | E^-]$$

where :

$$E^+_{ik} = \sum_{j\in B_k} a_{ij} \qquad E^-_{iz} = \sum_{j\in B'_z} a_{ji}$$

for all i = {1,...,n}, k = {1,...,h}, z = {1,...,c} and d = h+c. For undirected graphs where A is symmetric $E^+ = E^-$, and the embedding matrix reduces to $E = E^+$.

The matrix E collects the encoding of each node in the lower dimensional space $\mathbb{R}^d$. It is composed of the two

$$E^+ = \begin{bmatrix} 0 & 7 & 0 \\ 0 & 0 & 6 \\ 0 & 0 & 5 \\ 0 & 0 & 5 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad E^- = \begin{bmatrix} 0 & 0 & 0 \\ 2 & 0 & 0 \\ 3 & 0 & 0 \\ 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 2 & 0 \\ 0 & 2 & 0 \\ 0 & 2 & 0 \\ 0 & 3 & 0 \\ 0 & 2 & 0 \\ 0 & 3 & 0 \end{bmatrix}, \quad E = \begin{bmatrix} 0 & 7 & 0 & 0 & 0 & 0 \\ 0 & 0 & 6 & 2 & 0 & 0 \\ 0 & 0 & 5 & 3 & 0 & 0 \\ 0 & 0 & 5 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \end{bmatrix}$$

Fig. 5: Network embedding for the weighted directed network of Figure 4. The figure shows the two embedding matrices E^+ and E^- computed on the outgoing and incoming edges alongside the resulting network embedding E. The columns correspond to the blocks of the partitions sorted by their length.

matrices E^+ and E^-. In the former, the value E^+_{ik} corresponds to the sum of the weights of the outgoing edges from node i to nodes in the equivalence class B_k. In the latter, the value E^-_{ik} corresponds to the sum of the weights of the incoming edges from nodes in the equivalence class B'_z to node i. A distinctive feature of our approach is that each component of the embedding vector carries a well-defined semantic interpretation, ensuring full explainability. Furthermore, the algorithm is entirely deterministic, as it does not rely on any form of randomization. This guarantees that the resulting embeddings are inherently interpretable and reproducible.

In Figure 5, we show the embedding matrix E computed for the network in Figure 4 as a result of the concatenation of the two matrices E^+ and E^-. Consistent with the example introduced earlier, nodes that play similar roles within the network are assigned similar embedding vectors, highlighting the ability of our method to capture and preserve role-based similarity.

The presence of zero entries in the embedding matrix reflects the connections of nodes across equivalent blocks, and is a typical feature of our method. Interestingly, this inherent sparsity could be exploited to further improve the efficiency in downstream tasks.

In undirected networks, the matrices E^+ and E^- are equivalent, due to the symmetry of A. In this case, $E = E^+$ as can be seen in the embedding presented in the introduction and reported in Table I.

III. Computation of Iterative ε-BE

A. Partition-refinement Algorithm

We present an efficient algorithm to find an ε-BE partition based on the Markov chain lumping algorithm from [21]. We report the pseudocode in Algorithm 1. The algorithm starts with an initial partition and refines it, splitting the blocks, until it gets a partition where the ε-BE condition is satisfied. In general, the initial partition can be used to avoid the aggregation of undesired nodes; indeed, nodes in different blocks of the initial partition cannot be aggregated. Throughout this paper, initial partitions always consist of a single block containing all nodes. We will see how the notion of initial partition is instrumental to obtaining an iterative version of the algorithm in Section III-B.

Algorithm 1 ε-BE

Require: Adjacency matrix A, initial partition $\mathcal{H}_{in}$, tolerance $\varepsilon \geq 0$.

$U = \mathcal{H}_{in}$; $B_T = \{\}$;
while $U \neq \emptyset$ **do**
  $w^{B'}[s] = 0$ for each node s
  $B' =$ remove a block from U.
  $S_T = \{\}$
  for $s' \in B'$ **do**
    for s connected to s' **do**
      if $w^{B'}[s] == 0$ **then**
        $S_T = S_T \cup \{s\}$ $w^{B'}[s] = A[s][s']$
      else
        $w^{B'}[s] = w^{B'}[s] + A[s][s']$
      end if
    end for
  end for
  for $s \in S_T$ **do**
    $B =$ the block that contains s
    if B not in B_T **then**
      $B_T = B_T \cup \{B\}$
    end if
  end for
  while $B_T \neq \emptyset$ **do**
    $B =$ remove a block from B_T
    sort B according to the values $w^{B'}[s]$
    partition B in $B_1, B_2, ..., B_l$ respect to tolerance ε
    if $l > 1$ **then**
      add $B_1, B_2, ..., B_l$ to U
    end if
  end while
  for $s \in S_T$ **do**
    $w^{B'}[s] = 0$
  end for
end while

The algorithm iteratively scans a list of *splitters*, i.e., partition blocks. The list is initialized with the initial partition. It removes each splitter from the list and checks whether all blocks satisfy the condition for the splitter. The blocks not satisfying the condition are refined such that the ε-BE condition is satisfied for the splitter; the newly identified blocks are added to the list of splitters. The computation terminates when the list of splitters is empty.

For the sake of simplicity, we present our approach applied to the undirected, unweighted network presented in the introduction. In general, the algorithm does not require any modifications, both for weighted and directed networks, although for the latter, it has to compute two partitions, one for A^T and another for A. More specifically, the original algorithm [21] checks the condition on the outgoing edges, while BE accounts for incoming edges. Although these are equivalent in the undirected case, for directed networks, running Algorithm 1 on A corresponds to computing BE on A^T, whereas running it on A^T corresponds to computing BE on A.

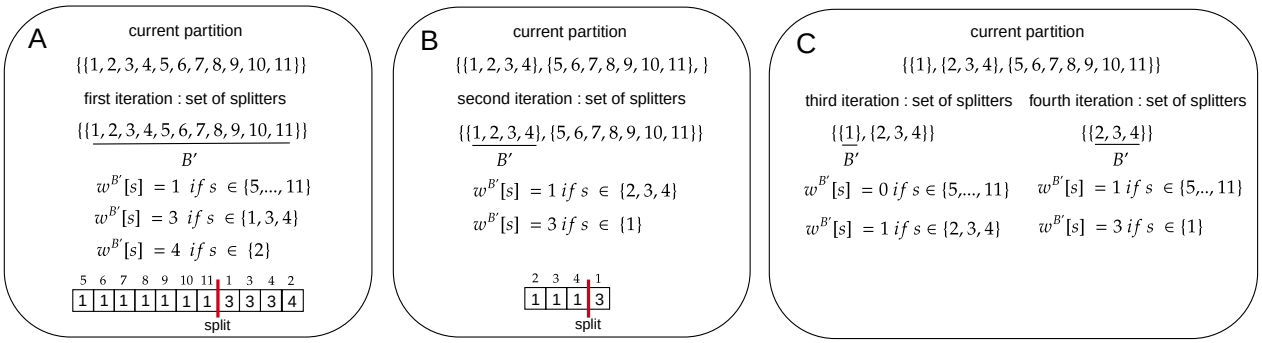


Fig. 6: Execution of Algorithm 1. We indicate the current splitter by underlining it. The weight $w^{B'}$ denotes the number of edges connected to elements of the splitter B'. The arrays at the bottom show the splitting phase of a certain block B with respect to the current splitter. The red line indicates where the block is split because it does not respect the tolerance ε. The computation of weights $w^{B'}[s]$ for certain splitters is omitted.

For an illustrative explanation, Figure 6 reports the computation of ε-BE on our running example with $\varepsilon = 1$. It is divided into three phases (panels A, B, and C in Figure 6). In panel A, we show the set of splitters U as the blocks in the initial partition (line 1). The procedure removes a block from U and initializes S_T, the set of nodes directed to any node in the current splitter (lines 3-5). In Figure 6 we indicate the current splitter with B'. Then, the algorithm computes the $w^{B'}[s]$ as the sum of the weights from the node s to a node in the current splitter B' (lines 6-14). In panel A, $w^{B'}[s]$ is computed for the first block. The nodes $\{5, ..., 11\}$ are connected to one node belonging to B', $\{1, 3, 4\}$ to three nodes, and $\{2\}$ to four nodes.

The blocks containing at least a node directed to the current splitter are retrieved and stored in B_T (lines 15-20). For each block B in B_T, the algorithm sorts $w^{B'}[s]$ with $s \in B$ and splits them into blocks (lines 21-28). Here, differently than [21], we allow a tolerance ε in the aggregation of the nodes.

Figure 6 shows the splitting process in the bottom part of panel A. The nodes s belonging to B are sorted by $w^{B'}[s]$. If the difference between the maximum and minimum nodes is greater than ε, it splits the block. A red line indicates the split. For instance, when node $\{1\}$ is considered, the difference is greater than $\varepsilon = 1$ and it splits the block.

After the splitting process, in panel B of Figure 6, the current partition is composed of two blocks. The first block $\{1, ..., 4\}$, where the difference is 0, and the second block $\{5, ..., 11\}$, where the difference is 1. On line 25, we evaluate whether the number of new blocks exceeds 1. If this condition holds true, the new blocks are incorporated into U as additional splitters. When l equals 1, B is not split into any new blocks, so we avoid adding itself to U.

The algorithm considers a new splitter B' and tries to split the first block. It considers the sorted $w^{B'}$ and it finds that node 1 can be separated. At this point, the set of splitters and the partition will be updated. In phase C, we consider the remaining splitters. No actual refinements are performed, hence the procedure terminates. Therefore, the output is a partition with three blocks as pictured in Figure 3.

Algorithm 2 Iterative ε-BE.

Require: Adjacency matrix A, an initial tolerance ε_0, step size $\delta \geq 0$, maximum tolerance $\Delta \geq 0$.

1: $\varepsilon = \varepsilon_0$
2: $\mathcal{H}_{in} = \{\{1, ..., N\}\}$
3: **while** $\varepsilon \leq \Delta$ **do**
4: $\quad \mathcal{H}_\varepsilon = \varepsilon$-BE$(A, \mathcal{H}_{in}, \varepsilon)$
5: $\quad \mathcal{H}_{in}$ = joinSingletons$(\mathcal{H}_\varepsilon)$
6: $\quad \varepsilon = \varepsilon + \delta$
7: **end while**
8: **return** $\mathcal{H}_\varepsilon$

B. Iterative ε-BE

One of the crucial aspects of Algorithm 1 is the choice of the tolerance ε: increasing this value leads to partitions with few, large blocks, but it may collapse the network too aggressively. For instance, running it with $\varepsilon = 3$ on the running example in Figure 2 results in the trivial partition composed of a single block. For an effective network embedding, we propose an iterative strategy called Iterative ε-BE (Iε-BE). The pseudocode is shown in Algorithm 2.

It requires the following parameters: an initial tolerance, denoted by ε_0, a step size δ, which determines the increment of ε at each step, and Δ, the maximum value of ε to be considered. In line 2, the $\mathcal{H}_{in}$ is initialized with the trivial partition composed of a single block. Although the initial partition can be initialized to exploit insights of the problem, we decide to start with the trivial one in order to provide a fair analysis against the competitor in the experimental analysis. The main loop iterates in lines 3-7 until it reaches the maximum threshold Δ. In line 4, Algorithm 1 is called to find the partition up to a certain threshold ε. The so-computed partition is stored in $\mathcal{H}_\varepsilon$ (line 4). In line 5, all the singleton blocks in the current partition $\mathcal{H}_\varepsilon$ are joined into one block for the next iteration. The intuition is to attempt node aggregation for smaller values of ε first. If that fails, i.e., nodes are eventually outputted as singleton blocks, the merging of such nodes is used to attempt aggregation for the

This article has been accepted for publication in IEEE Transactions on Knowledge and Data Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2026.3682970

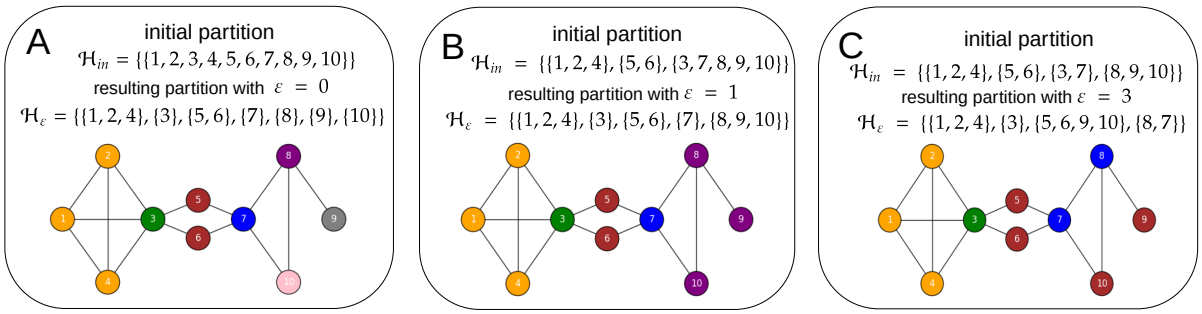


Fig. 7: Execution of the iterative Iε-BE on a network composed of a complete clique on the left, and another on the right with missing edges. We show the first and the second iteration with ε equal to 0 and 1 in panels A and B. For each panel, we indicate the initial partition and the resulting one after the application of ε-BE. The resulting partition of a phase will be employed as the initial partition in the following phase after merging the singleton blocks. The initial partition $\mathcal{H}_{in}$ in panel B corresponds to $\mathcal{H}_{\varepsilon}$ in panel A, where the singleton blocks are merged into a single block. We depict the network with the different colors assigned by the resulting partition. Panel C represents the result of the application of Algorithm 1 without the iterative scheme.

larger ε values in the next iterations. Finally, in line 6, the algorithm increments the value of ε by δ.

Figure 7 provides an illustrative example. It shows a network with two cliques connected by edges; the one on the left side is a complete clique (nodes 1, 2, 4, 3), while the one on the right has some missing edges (nodes 7, 8, 9, 10). Intuitively, one expects to group the elements of the complete clique with $\varepsilon = 0$, while for the incomplete one, we expect to find an association of the cliques' nodes, considering a tolerance greater than 0. We employ the Algorithm 2 in the example in Figure 7. We set up $\varepsilon_0 = 0$, $\delta = 1$ and $\Delta = 1$. This corresponds to two iterations, with ε equal to 0 and 1, shown in panels A and B of Figure 7, respectively.

Panel A shows the initial partition $\mathcal{H}_{in}$ and the coloration for $\varepsilon = 0$. As expected, the nodes in the left clique present the same role. In panel B the initial partition $\mathcal{H}_{in}$ presents the merged singleton blocks $\{3\}$, $\{7\}$, $\{8\}$, $\{9\}$, $\{10\}$ of $\mathcal{H}_{\varepsilon}$ at the previous iteration of panel A.

In the subsequent iteration, the algorithm computes the partition with $\varepsilon = 1$, considering as a new initial partition the one computed in the previous iteration. The results are depicted in panel B, where, due to the tolerance parameter ε, nodes within the incomplete clique are denoted with the same color. This example demonstrates how the algorithm can collect similar groups of nodes while gradually increasing the tolerance ε.

To show the advantages brought by the iterative scheme, panel C shows the output of a *one-shot* application of Algorithm 1 directly with $\varepsilon = 1$. This comparison shows that the iterative algorithm is able to better assign the roles to nodes in the two cliques, while the one-shot application identifies nodes 5, 6, 9, and 10, in contrast with their intuitively different roles in the network.

This approach proves particularly effective in real-world networks, which often exhibit skewed degree distributions characterized by a small number of high-degree nodes and a large number of low-degree ones. In such scenarios, identifying meaningful aggregations among the most important nodes with high degrees typically requires setting a large value of ε. However, doing so tends to cause indiscriminate aggregation among the numerous low-degree nodes located in the distribution's plateau.

The proposed iterative scheme addresses this limitation by progressively increasing the tolerance ε. In [52], we demonstrated the effectiveness of a similar iterative strategy for computing approximate regular equivalence on real networks with skewed degree distributions, using ε-BDE, a variant originally developed for dynamical systems [57].

C. Correctness

We now provide an informal proof for the correctness of the ε-BE and Iε-BE algorithms. Recall that Algorithm 1 generalizes Valmari's partition-refinement procedure [21] by allowing a tolerance $\varepsilon \geq 0$.

Theorem 1. *Let $\mathcal{H}$ be a partition of the node set V. For a block $B' \in \mathcal{H}$ and a node $s \in V$, define*

$$w^{B'}[s] \;=\; \sum_{t \in B'} a_{s,t}.$$

We say that $\mathcal{H}$ is an ε-BE partition *if for all blocks $B, B' \in \mathcal{H}$ and $i, j \in B$,*

$$w^{B'}[i] - w^{B'}[j] \;\leq\; \varepsilon.$$

Algorithm 1 terminates and returns a ε-BE refinement of the initial partition $\mathcal{H}_{in}$.

Proof. The algorithm processes a queue of splitters U, initially set to $\mathcal{H}_{in}$. Each time a splitter B' is removed from U, the algorithm computes the values $w^{B'}[s]$ for all s and refines any block B where the range of $w^{B'}$ exceeds ε. The resulting sub-blocks are guaranteed to be ε-stable with respect to B', and they are added to U for further checking.

Each refinement strictly increases the number of blocks, and no merges occur. Since at most $n-1$ splits are possible, the algorithm must terminate. At termination, every block has been checked against every splitter, so the partition is ε-BE. □

The correctness arguments extend unchanged to weighted and directed networks, with the only difference that, in the directed case, one must compute the partition for both A and $A^\top$. Once the correctness of Algorithm 1 is established, the correctness of Iε-BE follows directly, since Algorithm 1 is invoked as an inner step and the final partition $\mathcal{H}_\varepsilon$ is produced by it. In summary, Algorithm 1 is a direct extension of Valmari's procedure to $\varepsilon > 0$, and Algorithm 2 considers a sequence of such refinements with increasing tolerances. Together, they guarantee that the output is always a valid ε-BE partition.

D. Complexity

The original partition-refinement algorithm from [21] runs in $O(m\log(n))$ time, where n is the number of nodes and m is the number of edges. The logarithmic term is achieved by discarding the largest block in each iteration. Unfortunately, for an approximate reduction, as presented here, discarding the largest block does not assure correctness (i.e., the partition does not satisfy the ε-BE condition). Consequently, the complexity of the Algorithm 1 is $O(mn)$. The splitting of the blocks according to the tolerance ε can be implemented in linear time once the nodes are sorted for their weights w, and does affect the overall complexity.

Algorithm 1 is employed as an inner step in Algorithm 2. An important point to consider is how many times lines 3-7 are repeated. In this case, the maximum threshold Δ and the step size δ have to be considered. The inner loop will be repeated $\lceil\Delta/\delta\rceil$ times. Consequently, the overall complexity is $O(mn\lceil\Delta/\delta\rceil)$. Despite this, the parameter Δ is always orders of magnitude smaller than n to avoid trivial results where all the nodes belong to the same block. In the case of directed networks, the algorithm computes two partitions, one for A^T and another for A, adding only a multiplicative constant that does not affect the overall complexity of our approach.

IV. Numerical Experiments

In this section, we present the experimental results achieved by Iε-BE compared with state-of-the-art methods for network embedding. All the experiments and datasets with the implementation of our method are accessible on the GitHub page.[1] We considered the best methods reported in the recent review [4]. For undirected unweighted networks, we chose Graphwave [38], SEGK [40], and RIDεRs [34] as representatives of matrix factorization techniques, struc2vec [43] as a random-walks-based approach, and DRNE [47] and GAS [48] as representatives of deep learning methodologies. For weighted and directed networks, we consider respectively Graphwave and DRNE, while EMBER [46] is the only tool that, similarly to our approach, can handle both. The implementation details for each method were taken from GitHub repositories mentioned in the respective publications or in [4].[2] Following [4], [38], [40], we conducted experiments on synthetic and real networks considering four machine-learning tasks: visualization, classification, clustering, and regression. All experiments were run on a machine equipped with an Intel(R) Xeon(R) CPU E7-4830 with 500 GB RAM. We set up a timeout of 3 hours for all the computations reported in this section.

[1]GitHub link: https://github.com/EmbNet01/EmbNetworks.git

[2]GitHub repositories accessed on 01/01/2025.

Network	*Ref.*	*type*	N	h	c	d	δ	Δ	ε_0
Brazil airport	[43]	UU	131	38	38	38	1	3	0
Europe airport	[43]	UU	399	96	96	96	1	4	0
USA airport	[43]	UU	1190	325	325	325	1	2	0
actor	[4]	UU	7779	201	201	201	2	8	2
film	[4]	UU	27312	177	177	177	6	30	6
Miserables	[58]	WU	77	21	21	21	5	50	0
New Zealand	[58]	WU	1511	160	160	160	25	150	0
Bible	[58]	WU	1773	27	27	27	20	100	20
HS	[58]	WU	866	9	9	9	20	100	20
SITC	[58]	WU	774	23	23	23	30	60	30
Anybeat	[58]	UD	12645	65	59	124	10	50	10
Film Trust	[59]	UD	874	10	9	19	5	30	5
FAA	[58]	UD	1226	16	11	27	3	30	3
Ecoli	[58]	UD	423	8	16	24	1	12	1
Email	[58]	UD	1133	14	14	28	10	100	10
FB	[60]	WD	1899	8	6	14	2E+4	1E+5	2E+4
USA airport	[61]	WD	500	20	20	40	1E+6	1E+7	1E+6
Advogato	[58]	WD	6541	6	7	13	50	250	50
Hall	[58]	WD	217	21	19	40	5	30	15
CS-Hiring	[58]	WD	2037	25	22	47	5	30	10

TABLE II: Parameter settings of the Iε-BE algorithm for the real-world networks employed in the downstream tasks. The third column indicates the type of the network: unweighted undirected (UU), unweighted directed (UD), weighted undirected (WU), and weighted directed (WD). For undirected networks $d = h = c$, while for directed $d = h + c$.

A. Set-up

Most of the methods in network embedding allow the user to select the desired embedding dimension d for the network. However, choosing the optimal dimension is challenging [62]. In this study, we adopt the default embedding dimensions recommended for each method. Although this results in different dimensionalities across methods, the choice is consistent with the methodology used in the original publications [4], [38], [40], where comparisons were also performed using the respective default settings. Specifically, we set d to the following values: 20 (SEGK), 100 (Graphwave), 64 (DRNE), and 128 (struct2vec, GAS, and EMBER). For RiDεRs, where the embedding dimension is determined by NMF, a fixed dimensionality cannot be predefined. Therefore, we use the dimension corresponding to the best model returned by NMF.

Iε-BE yields a discrete embedding based on the partition computed by Algorithm 2. The reduction with the maximum number of blocks corresponds to a partition where every node is a singleton, resulting in an embedding equivalent to the adjacency matrix itself. In this scenario, similar nodes are defined as those that are connected to the exact same set of nodes. Conversely, considering a sufficiently large ε, the partition with the minimum number of blocks groups all nodes into a single block, resulting in an embedding dimension d equal to 1. Here, each entry represents the sum of the elements in the respective columns, and similar nodes are defined as those with comparable degrees. These represent two extreme reductions of the embedding space. Given that the primary objective of network embedding is to offer an encoding where d is significantly smaller than N, we set the parameters to achieve a reduction of at least 70% of the

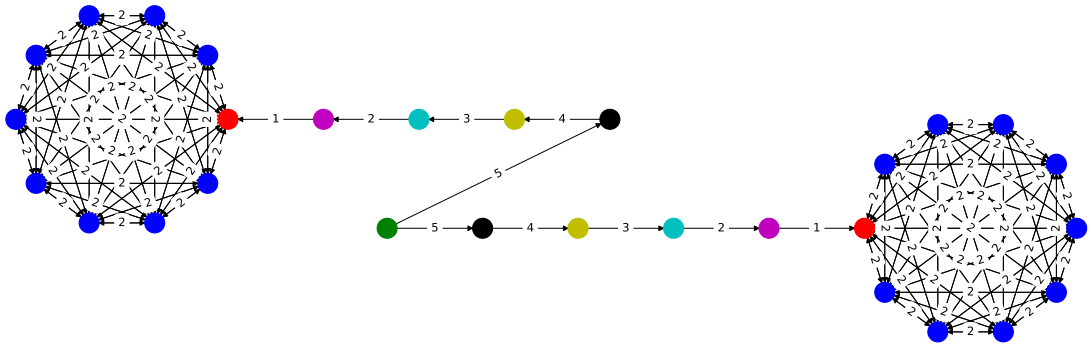


Fig. 8: Weighted directed barbell role assignment.

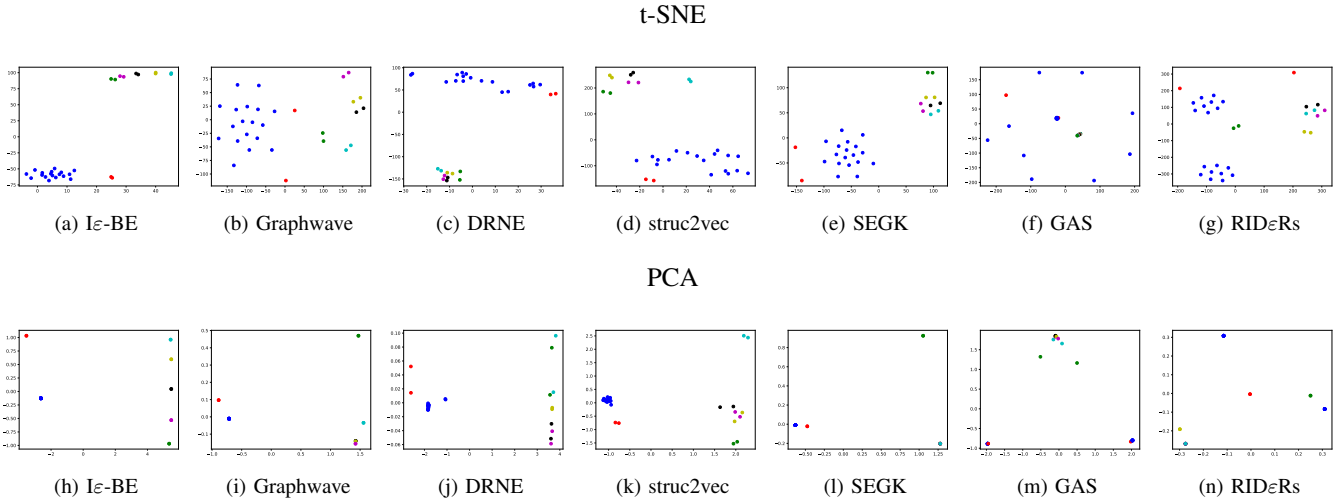


Fig. 9: Results of visualization task using t-SNE and PCA on unweighted undirected (UU) barbell network. Each plot has a caption indicating the embedding method.

network dimension N. We show in Table II, the list of the real-world networks considered for our analysis, specifying parameters utilized for computing the partitions with our approach and the dimension of the resulting embedding matrix. Additionally, we report the network dimension N and the type with the following abbreviations: unweighted undirected (UU), unweighted directed (UD), weighted undirected (WU), and weighted directed (WD).

B. Visualization

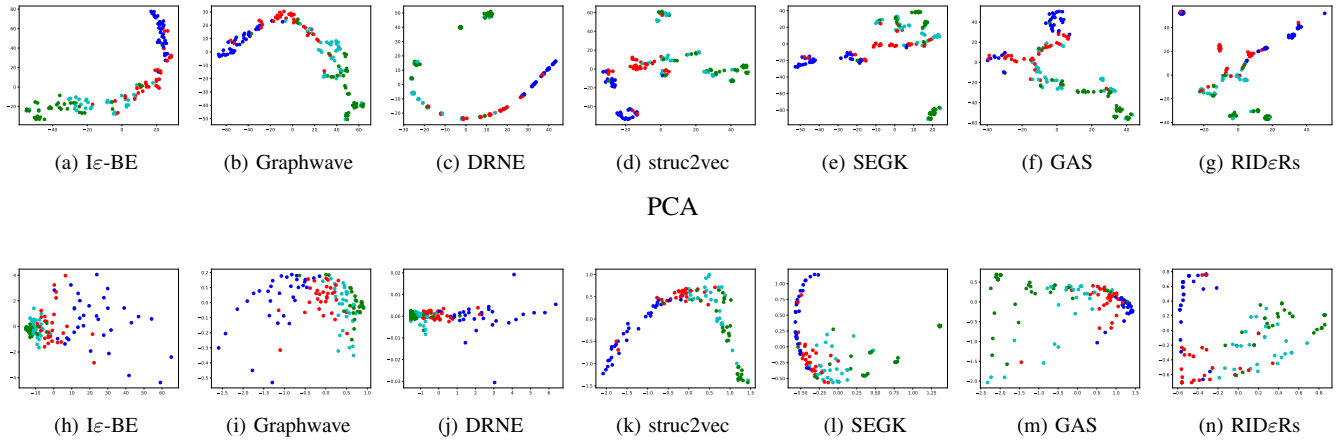
Network visualization is a fundamental task, particularly in scenarios where the roles of network nodes are predefined. The primary goal is to project these nodes into a two-dimensional space for visualization purposes. An effective embedding technique should position nodes with similar roles in close proximity within this reduced-dimensional space.

Iε-BE, RIDεRs, and Graphwave are not directly suitable for visualization. For this reason, we further reduce the embedding space of dimension d into two dimensions using PCA and t-SNE [63]. The former is a deterministic approach based on matrix factorization. The latter is a probabilistic method based on a variation of Stochastic Neighbor Embedding [64].

We consider the barbell graph a standard benchmark in the literature for evaluating visualization tasks [38], [40], [43]. It consists of two cliques connected by a chain of nodes, where each node is assigned a distinct role represented by different colors. The nodes within each clique share the same role, except for the red nodes, which are additionally connected to the chain. Conversely, the roles within the chain are specular. We show in Figure 8, the directed weighted version where edges are labelled with weights and have a specific direction. The other simpler versions can be generated assuming weights equal to 1 or by removing the directionality of the edges.

We start considering the visualization task on the standard unweighted undirected barbell. In Figure 9, we show the visualizations achieved by different embedding methods using t-SNE. We observe notable differences in the performance of different network embedding approaches. GAS and RIDεRs exhibit poorer performance than the other methods, especially GAS, where the majority of nodes are clustered closely together in the embedding space, making it challenging to discern distinct groups. In contrast, SEGK and Graphwave have superior performance by effectively distinguishing nodes within cliques. However, they tend to separate the red nodes into two separate regions within the embedding space. On the

This article has been accepted for publication in IEEE Transactions on Knowledge and Data Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2026.3682970

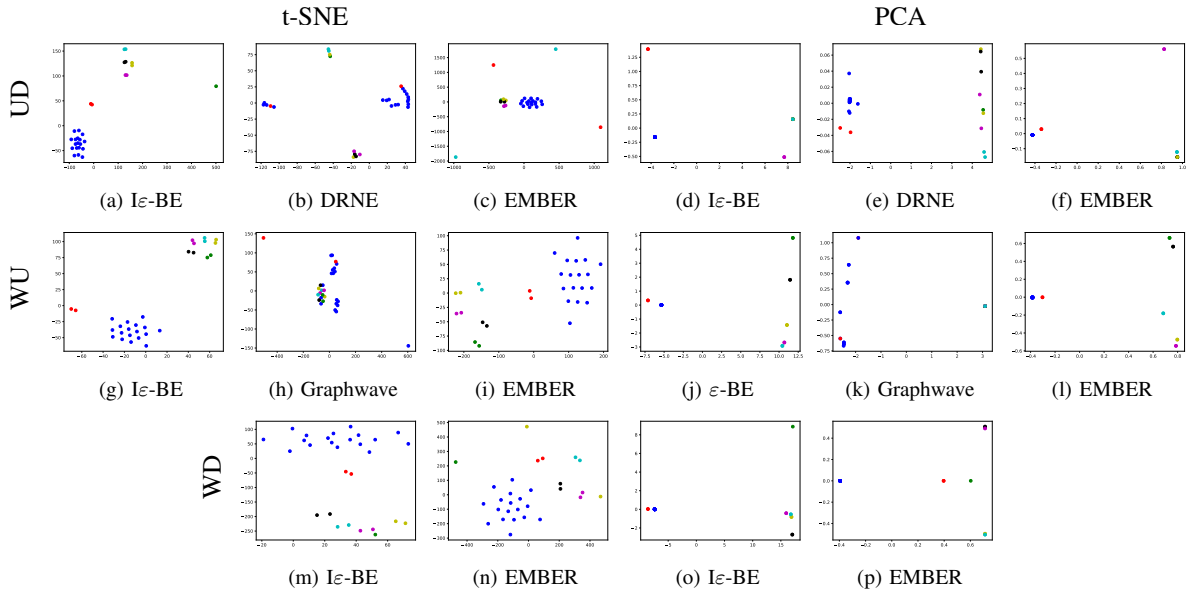


(a) Iε-BE (b) Graphwave (c) DRNE (d) struc2vec (e) SEGK (f) GAS (g) RIDεRs

PCA

(h) Iε-BE (i) Graphwave (j) DRNE (k) struc2vec (l) SEGK (m) GAS (n) RIDεRs

Fig. 10: Results of visualization task using t-SNE and PCA on Brazil airports network. Each plot has a caption indicating the embedding method.



(a) Iε-BE (b) DRNE (c) EMBER (d) Iε-BE (e) DRNE (f) EMBER

(g) Iε-BE (h) Graphwave (i) EMBER (j) ε-BE (k) Graphwave (l) EMBER

(m) Iε-BE (n) EMBER (o) Iε-BE (p) EMBER

Fig. 11: Results of visualization task using t-SNE and PCA on weighted directed barbell networks. Each plot has a caption indicating the embedding method. The caption at the beginning of each row indicates the type of the barbell network (UD = unweighted directed, WU = weighted undirected, WD = weighted directed).

other hand, struc2vec and DRNE perform well, providing a good spatial organization. Interestingly, our approach yields results similar to struc2vec with a cleaner separation within the 2-dimensional space.

Considering the inherent probabilistic nature of t-SNE, we also considered a deterministic approach using PCA, shown in Figure 9. In this case, nodes with the same encoding in the embedded space are mapped to the same point in the 2-dimensional space. We reduced the Barbell network setting ε equal to 0, which results in an embedding where nodes with the same role present the same embedding vector. Indeed, the theory guarantees that the difference between corresponding components in nodes with the same role remains below the specified tolerance ε. As a consequence, all nodes with the same role are mapped onto the same point for Iε-BE. Graphwave, RIDεRs, and SEGK show similar behavior, although they differentiate chain nodes less effectively. The other methods present a more sparse allocation without nodes that overlap perfectly. This is due to the fact that they rely on random walks and deep learning techniques. Consequently, it is unlikely to yield embedding vectors with identical components across these approaches. As a result, the application of PCA to visualize the embeddings generated by these methods yields less uniform results compared to other techniques.

To improve the robustness of our analysis, we evaluate the visualization task on the Brazil airports network. In Figure 10, we report the visualizations obtained with both t-SNE and PCA for each embedding method. In the case of t-SNE, both Iε-BE and Graphwave provide a meaningful spatial organization, with nodes of similar roles grouped into identifiable regions. Other methods tend to produce more scattered layouts, while they can often separate different classes, nodes from

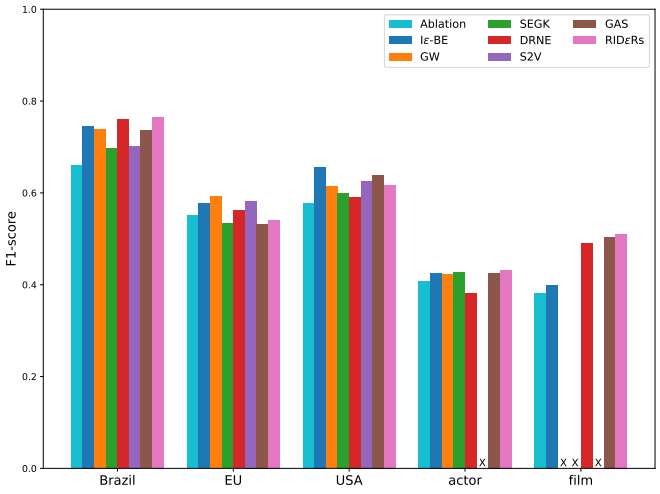


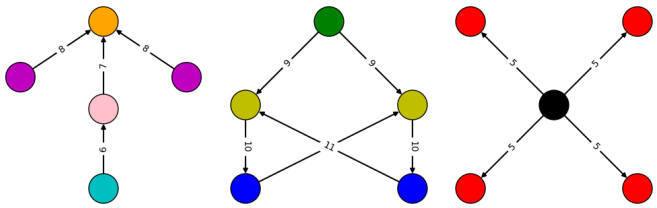
Fig. 12: Results of the classification task for the unweighted undirected benchmark networks. The bar indicated with 'X' refers to methods that timed out (3 h).

the same class are sometimes split into distant regions, or significantly overlap with nodes from other classes. For PCA, Iε-BE yields a clear separation between blue and red nodes. Cyan and green nodes form distinguishable regions, although they are located close to each other and partially overlap. In general, most methods struggle to fully separate green and cyan nodes. Embeddings such as Iε-BE, DRNE, struc2vec, Graphwave, and SEGK result in more compact spatial representations, whereas GAS and RIDεRs tend to distribute the nodes more sparsely across the projection space.

We further expand our analysis by considering weighted and directed versions of the barbell network. We show in Figure 11 the visualization results. As previously seen, DRNE and Graphwave are not able to compete with our approach, showing poor allocation of nodes with similar roles. On the other hand, EMBER is a competitive approach showing improvements respect to the other candidates. Despite this, sometimes the allocation is not optimal and nodes with different roles are aggregate close (Figure 11-c) or nodes with the same role are separated (Figures 11-c 11-n). In contrast, Iε-BE present always a clean and sorted allocation of the node in the embedding space. Notably, in the visualization of directed barbells with PCA both ε-BE and EMBER struggle to distinguish the nodes belonging to the segment among the two cliques overlapping these nodes in a single tiny region of the space. Overall, for these case studies Iε-BE approach emerges as the best strategy for accurate visualization using t-SNE and PCA.

C. Classification

The objective of classification is to train a model that can accurately assign appropriate roles to new nodes. We proceed by computing the embedding space for each network, followed by a 5-fold cross-validation to assess the accuracy of role prediction. We employ a k-nearest neighbor classifier with k



(a) arrow (b) house (c) star

Fig. 13: Three synthetic weighted directed basic structures. Different colors indicate different structural roles.

equal to 5 to predict node labels in the test set and the F1 score to evaluate the classification performance.

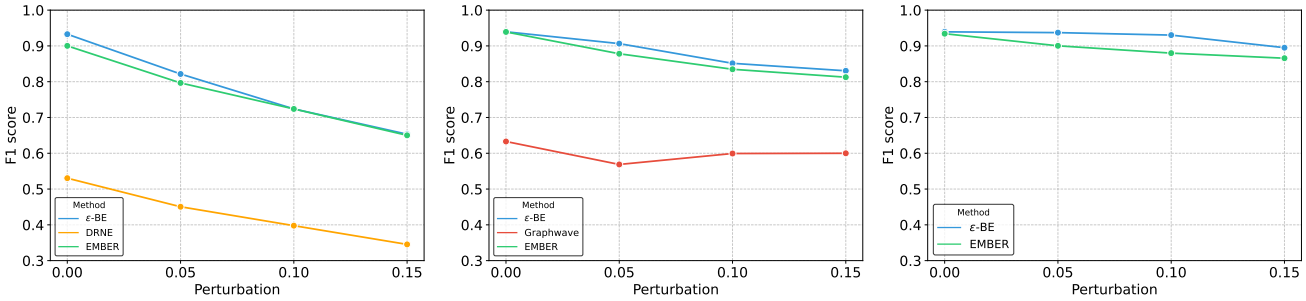
We start by considering 5 different unweighted undirected real-world networks. We started with the air traffic networks from Brazil, the United States, and Europe, presented in [43]; additionally, we use an actor co-occurrence network and the Film network from [4]. Air traffic networks are widely recognized within the network community and serve as standardized benchmarks for comparative evaluations [43]. All the considered networks come with ground truth labels that specify the role of each node.

For each method, we computed the corresponding network embedding. In [51], the authors note that embeddings derived from approximate equitable partitions can be complemented with additional structural features to improve role detection in node classification. Inspired by this principle, we enrich our embedding by incorporating the node degree as an additional feature for real-world networks.

We averaged the results over 50 5-fold cross-validations and reported them in Figure 12. In all the networks, ε-BE is able to achieve comparable results in the classification task. This is in line with recent findings claiming that there is currently no method that clearly outperforms the state of the art in this task [4]. The Film network proved challenging; however, three other methods (Graphwave, SEGK, and struc2vec) timed out in this case. Additionally, for the ablation study, we report the results of our method using our embedding computed without the iterative scheme. Overall, performance deteriorates, particularly on the Brazil and USA networks, while for the others, the decrease is less pronounced. Notably, on the EU network, our approach still outperforms SEGK, GAS, and RIDεRs.

At the state of the art, there are no standard benchmark networks for directed and weighted networks. Therefore, we decided to conduct the classification tasks on synthetic networks. We consider three simple structures as depicted in Figure 13 where the role of each node is pre-defined and arranged to form a more intricate network. Here, we report only the weighted directed version of these synthetic structures. For the undirected and unweighted, it is enough to remove the weights and or the directionality of the edges. We build the synthetic networks by considering a circle graph with 30 nodes, and we attach one of these basic structures to each node. More in detail, for each type of network, we generated 20 synthetic networks where specific sequences of three basic structures are repeated along the circle. Consequently, the roles are adjusted according to the new connections generated, attaching a basic

This article has been accepted for publication in IEEE Transactions on Knowledge and Data Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2026.3682970



(a) (UD) synthetic networks (b) (WU) synthetic networks (c) (WD) synthetic networks

Fig. 14: Results of classification task on synthetic networks.

Method	Brazil		Europe		USA		actor		film	
	NMI	ARI	NMI	ARI	NMI	ARI	NMI	ARI	NMI	ARI
Iε-BE	0.4433	0.3111	**0.3406**	0.2207	0.2439	0.1080	0.1311	0.0621	**0.0610**	**0.0382**
Graphwave	0.4957	0.4107	0.3344	**0.2378**	0.2417	0.1602	**0.1579**	**0.1275**	—	—
SEGK	0.3814	0.2360	0.2393	0.1733	0.2194	0.1803	0.1148	0.1068	—	—
DRNE	0.4485	0.3381	0.3381	0.2199	0.2495	0.1182	0.0314	0.0319	0.0286	0.0333
struc2vec	0.3687	0.2324	0.2922	0.2207	0.2337	0.1679	—	—	—	—
GAS	0.4156	0.3540	0.2245	0.0815	**0.2709**	**0.2503**	0.0815	0.0581	0.0086	0.0018
Rid	**0.5122**	**0.4844**	0.2212	0.1733	0.1664	0.1459	0.0623	0.0520	0.0281	0.0276

Method	WU		UD		WD	
	NMI	ARI	NMI	ARI	NMI	ARI
Iε-BE	**0.9670**	**0.9665**	**0.9609**	**0.9616**	0.1672	0.0323
EMBER	0.9162	0.8238	0.9547	0.9429	**0.1677**	**0.0332**
DRNE	—-	—-	0.5679	0.3017	—-	—-
Graphwave	0.7606	0.5170	—-	—-	—-	—-

TABLE III: Average NMI and ARI metrics computed on 10 runs of clustering for each method.

structure to the circle graph. Following [40], we augment the complexity of the task by introducing various perturbations represented by the increase and decrease of the weights in the basic structures. Specifically, we randomly select 2 nodes in a structure around the circle, and we increase/decrease the weight of the edge by 1. If the weight reaches 0, the edge is removed. For unweighted networks, this process is limited to adding and deleting edges. Perturbations at different levels are considered, starting from no perturbation, followed by 5%, 10%, and 15% of the number of edges in the original network. For each set of 20 networks, we compute the cross-fold validations and we plot the average results in Figure 14. We can see that Graphwave and DRNE are not able to provide competitive performance in identifying the right role for the nodes neither with a few percentage of perturbations. EMBER and Iε-BE show an excellent classification capability. As expected, increasing the number of edges perturbed lead to a deterioration of the performance. Our approach has proven to be solid against uncertainty, providing always better results than EMBER.

D. Clustering

In the clustering task, the objective is to identify a node partition that closely aligns with the ground-truth class labels. To evaluate this, we first computed node embeddings and subsequently applied K-means clustering, with the number of clusters k set to the number of distinct ground-truth labels. We compared the different approaches using two established metrics: Normalized Mutual Information (NMI) and Adjusted Rand Index (ARI). Similarly, for the classification task, we enrich our embedding with the degree as an extra feature. Table III presents the average NMI and ARI scores across 10 independent runs of K-means clustering for each method and dataset.

Consistent with previous findings in the literature, no single embedding method consistently outperforms all others across all datasets. In our experiments, RIDεRs achieves the highest scores on the Brazil airport dataset and Graphwave on the actor dataset, while GAS obtained the best results on USA airports. Overall, our method showed slightly better performance, ranking first on two out of five networks, namely the European airports and the film dataset. Conversely, although RIDεRs performs strongly on Brazilian airports, it shows significantly lower performance on the remaining datasets. These results further support the notion that the effectiveness of an embedding method is highly dependent on both the characteristics of the network and the specific downstream task [4]. Notably, while our method was suboptimal for node classification in

UU	*Brazil*		*Europe*		*United States*		*Actor*		*Film*	
Method	EIG	BET	EIG	BET	EIG	BET	EIG	BET	EIG	BET
Iε-BE	**8.47E-05**	**2.79E-02**	**1.23E-05**	7.56E-03	**1.28E-05**	**4.12E-02**	9.93E-03	**1.73E-02**	**2.30E-03**	**8.10E-03**
Graphwave	5.30E-04	4.08E-02	8.51E-04	1.17E-02	2.95E-03	4.48E-02	2.36E-02	5.87E-02	*TO*	*TO*
SEGK	4.90E-03	6.17E-02	3.38E-03	1.41E-02	1.05E-02	5.12E-02	2.29E-02	6.06E-02	*TO*	*TO*
DRNE	3.41E-03	4.81E-02	5.97E-04	9.89E-03	3.51E-03	4.86E-02	**5.58E-03**	5.56E-02	1.02E-02	1.30E-02
struc2vec	3.68E-03	4.57E-02	2.47E-03	**7.19E-03**	5.66E-03	4.81E-02	*TO*	*TO*	*TO*	*TO*
GAS	7.88E-02	4.19E-01	1.56E-02	2.69E-02	1.62E-02	6.21E-02	1.87E-02	6.18E-01	1.04E-02	1.25E-02
RIDεRs	6.01E-03	5.03E-02	5.09E-03	1.50E-02	1.26E-02	5.35E-02	1.67E-02	6.01E-02	8.21E-3	1.10E-02
WU	*Miserables*		*New Zealand*		*Bible*		*HS*		*SITC*	
Iε-BE	**1.71E-03**	**1.27E-01**	**8.50E-07**	**1.19E-02**	**4.36E-04**	**3.76E-03**	**1.59E-04**	**2.13E-02**	**1.23E-04**	**3.55E-02**
Graphwave	1.76E-01	1.51E-01	2.23E-01	9.09E-02	7.13E-02	1.42E-02	1.36E-01	3.17E-02	1.50E-01	4.50E-02
EMBER	9.44E-02	1.35E-01	2.01E-01	8.42E-02	8.09E-02	1.48E-02	1.27E-02	2.22E-02	1.25E-01	3.96E-02
UD	*Anybeat*		*FilmTrust*		*FAAD*		*Ecoli*		*Email*	
Iε-BE	**3.95E-04**	**3.99E-03**	**3.10E-03**	**6.80E-03**	**5.77E-03**	**7.41E-03**	9.92E-01	**1.44E-04**	**1.52E-03**	**1.96E-03**
DRNE	7.17E-03	1.03E-01	3.50E-02	7.45E-03	3.07E-02	8.59E-03	1.04E+00	1.95E-04	7.47E-03	2.21E-03
EMBER	7.01E-03	1.05E-01	7.05E-02	1.47E-02	2.26E-02	1.33E-02	**8.55E-01**	1.71E-04	1.08E-02	3.62E-03
WD	*FB*		*USA airport*		*Advogato*		*Hall*		*CS-Hiring*	
Iε-BE	**2.94E-02**	**4.56E-02**	**1.41E-04**	**1.87E-01**	**1.21E-03**	**3.29E-03**	**2.49E-03**	**1.31E-02**	**1.40E-04**	**2.26E-04**
EMBER	1.59E-01	4.81E-02	1.13E-01	1.90E-01	2.26E-02	7.67E-03	3.01E-02	1.47E-02	4.95E-02	2.27E-04

TABLE IV: Mean squared errors for the network regression task. EIG and BET stand for eigenvector and betweenness centrality. We highlight the best and second-best results in green and blue, respectively. *TO*: timeout (3 h)

the film network, it achieves the best clustering performance on the same dataset, underscoring the task-dependent nature of embedding quality. Other methods, such as SEGK, DRNE, and struc2vec, in the best case, reach the second-best score. For weighted and directed networks, our approach proves more effective, outperforming competitors in 2 out of 3 networks. In this setting, Graphwave and DRNE, similarly to the classification task, never manage to deliver competitive results. Conversely, EMBER performs well, especially on weighted and directed networks, where it ranks first, although only by a narrow margin. Overall, Iε-BE consistently provides results that are best or second-best, making it the most reliable choice for directed and weighted networks.

E. Regression

In regression, the objective is to predict a continuous or numerical value based on input features. Following [47], we define the target score for regression by incorporating two well-recognized centrality measures: eigenvector and betweenness centrality. The idea is to use the encoding of each node in the embedding space as a predictor for the centrality measures in the original network.

The goal is to compare methods with respect to their ability to preserve the centrality measures in the lower-dimensional space. This is significant because the centrality measures have established correlations with the fundamental notions of role and position within a network [47].

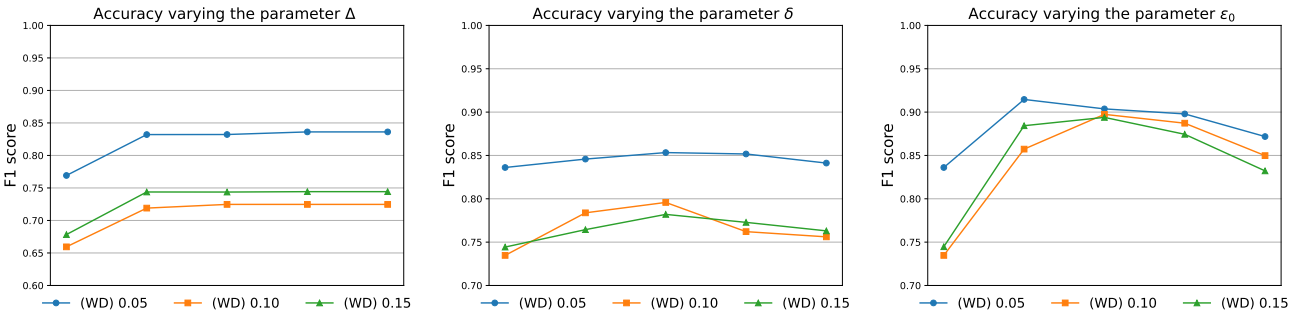
We train a linear regression, allocating 20% of the nodes as the designated test set and employing the remainder for training. We use the trained linear regression model to predict the centrality score for each node in the test set. In line with [47], we employ the Mean Squared Error (MSE) metric to quantify the discrepancy between the predicted and the true centrality. The former corresponds to the centrality predicted for each node in the test set. The latter corresponds to the centrality of these nodes in the real networks, where their encoding corresponds to the rows of the adjacency matrix A. To normalize for the different scales inherent to centrality measures, the resultant value is divided by the average value.

The analysis results are shown in Table IV. Iε-BE consistently exhibits optimal performance in eigenvector and betweenness centrality across nearly all network instances. This outcome is aligned with the intrinsic properties of the BE reduction methodology, which effectively preserves eigenvector centrality, Katz centrality, and PageRank centrality, as reported in the work [65]. This property represents one of the key strengths of our approach: by relying on formal definitions and fully deterministic, explainable embeddings, our method can guarantee structural preservation properties that are often difficult to ensure with more opaque techniques, such as deep learning-based embeddings. Although RIDεRs should inherit this property too, since it is based on approximate equitable partitions, it demonstrated good performance (second-best) for eigenvalue centrality only in the Film network, while it was outperformed by the competition in the other networks. Finally, we observe that ε-BE is highly competitive since it is the method with the best or second-best performance. Struc2vec, Graphwave, and DRNE display good performances but are not comparable with our approach. Additionally, some of them timed out when analyzing the larger networks. The results on weighted and directed networks strengthen the evident capability of our approach in aggregating nodes with similar centrality measures. In this case, the Iε-BE outperforms in almost all cases the other tools with differences in the mean squared errors up to 6 orders of magnitude.

F. Sensitivity Analysis

We evaluate the performance of our approach by varying the parameters of Algorithm 2. In particular, we focus on the

This article has been accepted for publication in IEEE Transactions on Knowledge and Data Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2026.3682970



(a) Δ-sensitivity (b) δ-sensitivity (c) ε_0-sensitivity

Fig. 15: Sensitivity analysis for weighted directed networks at different perturbation levels.

	Networks						
UU *Method*	*Brazil* $n = 131$	*USA* $n = 1190$	*BlogCatalog* $n = 10312$	*Brightkite* $n = 58228$	*GitHub* $n = 177316$	*Stackoverflow* $n = 641876$	*Song* $n = 1085612$
Iε-BE	**1.43E-1**	**2.93E-1**	**2.04E+0**	**7.72E+0**	**3.01E+1**	**7.86E+1**	**6.20E+2**
Graphwave	2.52E+0	1.03E+1	1.47E+3	*TO*	*TO*	*TO*	*TO*
SEGK	1.92E+1	4.58E+3	*TO*	*TO*	*TO*	*TO*	*TO*
DRNE	1.07E+1	2.50E+1	1.96E+2	1.26E+3	*TO*	*TO*	*TO*
struc2vec	1.83E+1	9.93E+3	*TO*	*TO*	*TO*	*TO*	*TO*
GAS	2.30E+0	7.95E+0	4.90E+2	4.98E+2	2.65E+3	*TO*	*TO*
RIDεRs	3.51E+1	2.73E+2	3.96E+2	7.38E+3	*TO*	*TO*	*TO*

TABLE V: Network embedding running times (in s) for unweighted undirected networks. *TO*: timeout (3 h).

	Networks				
UD *Method*	*Physicians* $n = 241$	*Weka* $n = 2124$	*Anybeat* $n = 12645$	*Caida* $n = 26389$	*Standford* $n = 281904$
Iε-BE	**1.62E-1**	**3.51E-1**	**1.09E+0**	**1.45E+0**	**2.83E+1**
DRNE	7.41E+0	3.62E+1	2.14E+2	1.64E+3	*TO*
EMBER	1.42E+0	6.09E+0	7.87E+2	3.19E+3	*TO*
WU *Method*	*SITC* $n = 774$	*New Zealand* $n = 1511$	*Bible* $n = 1773$	*Arxiv* $n = 40421$	*Words* $n = 276739$
Iε-BE	**1.79E-1**	**2.89E-1**	**2.92E-1**	**1.73E+1**	**7.77E+1**
Graphwave	2.97E+0	2.41E+1	2.15E+1	*TO*	*TO*
EMBER	2.73E+0	1.94E+1	1.48E+1	1.97E+2	*TO*
WD *Method*	*Hall* $n = 217$	*FB* $n = 1899$	*Hemibrain* $n = 21739$	*Agency* $n = 42951$	*Libimseti* $n = 220970$
Iε-BE	**1.81E-1**	**3.31E-1**	**1.46E+1**	**3.16E+1**	**2.70E+2**
EMBER	1.38E+0	2.65E+1	*TO*	7.41E+2	*TO*

TABLE VI: Network embedding running times (in s) for weighted directed networks. *TO*: timeout (3 h).

classification task over directed weighted networks with perturbation levels ranging from 5% to 15%. We systematically vary one parameter at a time, namely the initial tolerance ε_0, the step size δ, and the maximum tolerance Δ, while keeping the others fixed. Figure 15 reports the classification accuracy under these variations.

The results show that very small values of Δ lead to reductions that fail to capture the underlying structure. Moreover, small Δ values, such as in the extreme case of exact reduction, are unable to consistently reduce the network dimension, making the approach impractical for downstream tasks. Increasing Δ improves performance until it quickly converges to a stable plateau. This behavior arises because, once the resulting partition contains few singleton blocks, further increasing Δ yields similar final partitions. In such cases, if the results are unsatisfactory or a more aggressive reduction is needed, adjusting ε_0 or δ is more effective. For the parameters δ and ε_0, the accuracy curves exhibit a parabolic trend. Both overly aggressive and overly conservative reductions result in poorer performance, indicating that excessively fine or excessively coarse partitions negatively affect classification accuracy. We also observe greater variability in networks with higher perturbation levels. This is due to the fact that larger perturbations require higher tolerances to be captured, which in turn necessitates exploring a broader parameter range. Interestingly, in some cases, the accuracy for networks with a

perturbation of 10% is lower than for those with 15%. This can be explained by the random nature of the perturbations, which may be captured better with certain parameter combinations than with others.

Overall, our experiments indicate that achieving a reduction greater than 70%, while avoiding embeddings with too few features, consistently leads to strong performance. Given the efficiency of our method, which can explore different parameter ranges within just a few minutes, we recommend that practitioners select parameters using a simple grid search. This is a common practice also in other approaches (see, e.g., [43], [48]).

G. Scalability

We empirically analyzed the scalability of Iε-BE by comparing its running times against the other methods on several networks of different types and of increasing size.

The running time of Iε-BE primarily depends on the number of iterations performed by the reduction process. To ensure a meaningful network reduction while keeping the number of iterations consistent with the values reported in Table II, we calibrated the parameters accordingly to the size of the network.

Results are reported in Table V, showing that Iε-BE consistently achieves superior runtime performance and is able to process networks that competing methods cannot handle within a reasonable time. Specifically, SEGK and struc2vec timed out on the BlogCatalog network, while Graphwave timed out on Brightkite. On the GitHub network, DRNE encountered a runtime error that prematurely interrupted the embedding process. RIDεRs timed out on networks with more than 10^6 nodes, and in all other cases its running time was at least two orders of magnitude higher than that of Iε-BE. The GitHub network could be effectively processed only by Iε-BE and GAS. However, GAS required running times two orders of magnitude larger than Iε-BE and failed to scale to larger networks. Similarly, as shown in Table VI, Iε-BE outperforms the other tools for weighted and directed networks. EMBER can handle networks with thousands of nodes, but waiting times are larger than 2 orders of magnitude. Additionally, it goes out of time for the largest network Libimseti, but also with Hemibrain. Despite the limited number of nodes of the Hemibrain network, it presents a huge number of connections that require hours of computation for EMBER.

V. Conclusion

Structural network embeddings are useful for tasks that exploit properties of role similarities of the nodes in a network. Our network embedding is based on Iε-BE, which partitions the nodes of a network into approximately equitable blocks using an efficient partition refinement algorithm embedded in an iterative framework that controls the impact of the tolerance parameter ε. Additionally, Iε-BE is suitable for the embedding of weighted and directed networks, a capability that few methods can show in the literature. The experimental evaluation on visualization, classification, clustering, and regression tasks demonstrates that Iε-BE consistently achieves comparable or superior performance compared to existing methods at shorter running times compared to other state-of-the-art methods. We plan to enhance Iε-BE by considering different initial partitions based on the problem context. While this study adopts an initial partition where all nodes fall into a single block to ensure a fair comparison with related works, incorporating problem-specific initial partitions may lead to improved performance. Additionally, we are interested in extending our technique for temporal networks, where the edges among the nodes are equipped with timestamps. In this context, it is interesting to study the evolution of the behavior and the role of a node over time. This could expand the range of applicability of our approach towards tasks, like anomaly detection and temporal role classification.

Acknowledgment

This work was partially supported by Poul Due Jensen Foundation grant no. 883901, the Villum Investigator Grant S4OS, the project SERICS (PE00000014), the Fsc regional Tuscan project AISLEA2 J54D23000780005, the project Tuscany Health Ecosystem (THE), CUP: B83C22003920001, the project SMaRT COnSTRUCT (CUP J53C24001460006), within FAIR (CUP I53C22001380006). The last two projects are under the MUR National Recovery and Resilience Plan funded by the EU - NextGenerationEU.

References

[1] L. Freeman, "The development of social network analysis," *A Study in the Sociology of Science*, vol. 1, no. 687, pp. 159–167, 2004.
[2] G. A. Pavlopoulos, M. Secrier, C. N. Moschopoulos, T. G. Soldatos, S. Kossida, J. Aerts, R. Schneider, and P. G. Bagos, "Using graph theory to analyze biological networks," *BioData mining*, vol. 4, pp. 1–27, 2011.
[3] L. Lü, M. Medo, C. H. Yeung, Y.-C. Zhang, Z.-K. Zhang, and T. Zhou, "Recommender systems," *Physics reports*, vol. 519, no. 1, pp. 1–49, 2012.
[4] P. Jiao, X. Guo, T. Pan, W. Zhang, Y. Pei, and L. Pan, "A survey on role-oriented network embedding," *IEEE Transactions on Big Data*, vol. 8, no. 4, pp. 933–952, 2022.
[5] P. Cui, X. Wang, J. Pei, and W. Zhu, "A survey on network embedding," *IEEE TKDE*, vol. 31, no. 5, pp. 833–852, 2018.
[6] J. Jin, M. Heimann, D. Jin, and D. Koutra, "Toward understanding and evaluating structural node embeddings," *ACM TKDD*, vol. 16, no. 3, pp. 1–32, 2021.
[7] R. K. Merton, *Social theory and social structure*. Simon and Schuster, 1968.
[8] D. R. White and K. P. Reitz, "Graph and semigroup homomorphisms on networks of relations," *Social Networks*, vol. 5, no. 2, pp. 193–234, 1983.
[9] S. P. Borgatti and M. G. Everett, "Notions of position in social network analysis," *Sociological Methodology*, vol. 22, pp. 1–35, 1992.
[10] F. Lorrain and H. C. White, "Structural equivalence of individuals in social networks," *The Journal of mathematical sociology*, vol. 1, no. 1, pp. 49–80, 1971.
[11] R. Jin, V. E. Lee, and H. Hong, "Axiomatic ranking of network role similarity," in *Proceedings of 17th ACM SIGKDD KDD*, 2011, pp. 922–930.
[12] C. D. Godsil, "Compact graphs and equitable partitions," *Linear Algebra and its Applications*, vol. 255, no. 1-3, pp. 259–266, 1997.
[13] M. G. Everett and S. P. Borgatti, "Exact colorations of graphs and digraphs," *Social networks*, vol. 18, no. 4, pp. 319–331, 1996.
[14] D. M. Cvetković, M. Doob, and H. Sachs, *Spectra of graphs: theory and applications*, A. Press, Ed., 1980.
[15] L. Cardelli, M. Tribastone, M. Tschaikowski, and A. Vandin, "Maximal aggregation of polynomial dynamical systems," *PNAS*, vol. 114, no. 38, pp. 10 029 – 10 034, 2017.
[16] P. Buchholz, "Exact and ordinary lumpability in finite markov chains," *Journal of applied probability*, vol. 31, no. 1, pp. 59–75, 1994.

[17] L. Cardelli, M. Tribastone, M. Tschaikowski, and A. Vandin, "Symbolic computation of differential equivalences," in *Proceedings of the 43rd ACM SIGPLAN-SIGACT POPL*, 2016.

[18] S. Tognazzi, M. Tribastone, M. Tschaikowski, and A. Vandin, "EGAC: a genetic algorithm to compare chemical reaction networks," in *GECCO*, P. A. N. Bosman, Ed. ACM, 2017, pp. 833–840.

[19] N. Gast, L. Bortolussi, and M. Tribastone, "Size expansions of mean field approximation: Transient and steady-state analysis," *ACM SIGMETRICS Performance Evaluation Review*, vol. 46, no. 3, pp. 25–26, 2019.

[20] L. Cardelli, I. C. Pérez-Verona, M. Tribastone, M. Tschaikowski, A. Vandin, and T. Waizmann, "Exact maximal reduction of stochastic reaction networks by species lumping," *Bioinform.*, vol. 37, no. 15, pp. 2175–2182, 2021.

[21] A. Valmari and G. Franceschinis, "Simple $O(m \log n)$ time markov chain lumping," in *Proceedings of 16th TACAS*, vol. 6015, 2010, pp. 38–52.

[22] R. Paige and R. E. Tarjan, "Three partition refinement algorithms," *SIAM Journal on computing*, vol. 16, no. 6, pp. 973–989, 1987.

[23] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.

[24] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th international conference on world wide web*, 2015, pp. 1067–1077.

[25] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 701–710.

[26] D. Jin, Z. Liu, W. Li, D. He, and W. Zhang, "Graph convolutional networks meet markov random fields: Semi-supervised community detection in attribute networks," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 152–159.

[27] Y. Chang, H. Ma, L. Chang, and Z. Li, "Community detection with attributed random walk via seed replacement," *Frontiers of Computer Science*, vol. 16, no. 5, p. 165324, 2022.

[28] E. Pan and Z. Kang, "Beyond homophily: Reconstructing structure for graph-agnostic clustering," in *International conference on machine learning*. PMLR, 2023, pp. 26 868–26 877.

[29] Z. Shen and Z. Kang, "When heterophily meets heterogeneous graphs: Latent graphs guided unsupervised representation learning," *IEEE Transactions on Neural Networks and Learning Systems*, 2025.

[30] W. Bi, L. Du, Q. Fu, Y. Wang, S. Han, and D. Zhang, "Make heterophilic graphs better fit gnn: A graph rewiring approach," *IEEE Transactions on Knowledge and Data Engineering*, 2024.

[31] X. Zheng, Y. Wang, Y. Liu, M. Li, M. Zhang, D. Jin, P. S. Yu, and S. Pan, "Graph neural networks for graphs with heterophily: A survey," *arXiv preprint arXiv:2202.07082*, 2022.

[32] W. Lin, F. He, F. Zhang, X. Cheng, and H. Cai, "Initialization for network embedding: A graph partition approach," in *Proceedings of the 13th International Conference on Web Search and Data Mining*, 2020, pp. 367–374.

[33] W. Lin, "Large-scale network embedding in apache spark," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 3271–3279.

[34] P. V. Gupte, B. Ravindran, and S. Parthasarathy, "Role discovery in graphs using global features: Algorithms, applications and a novel evaluation strategy," in *2017 IEEE 33rd ICDE*, 2017, pp. 771–782.

[35] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[36] K. Henderson, B. Gallagher, T. Eliassi-Rad, H. Tong, S. Basu, L. Akoglu, D. Koutra, C. Faloutsos, and L. Li, "Rolx: structural role extraction & mining in large graphs," in *Proceedings of the 18th ACM SIGKDD KDD*, 2012, pp. 1231–1239.

[37] S. Gilpin, T. Eliassi-Rad, and I. Davidson, "Guided learning for role discovery (GLRD) framework, algorithms, and applications," in *Proceedings of the 19th ACM SIGKDD KDD*, 2013, pp. 113–121.

[38] C. Donnat, M. Zitnik, D. Hallac, and J. Leskovec, "Learning structural node embeddings via diffusion wavelets," in *Proceedings of 24th ACM SIGKDD KDD*, 2018, pp. 1320–1329.

[39] M. Heimann, H. Shen, T. Safavi, and D. Koutra, "Regal: Representation learning-based graph alignment," in *Proceedings of the 27th ACM CIKM*, 2018, pp. 117–126.

[40] G. Nikolentzos and M. Vazirgiannis, "Learning structural node representations using graph kernels," *IEEE TKDE*, vol. 33, no. 5, pp. 2045–2056, 2019.

[41] P. Drineas and M. W. Mahoney, "Approximating a gram matrix for improved kernel-based learning," in *18th Annual Conference on Learning Theory, COLT*, 2005, pp. 323–337.

[42] Y. Pei, G. Fletcher, and M. Pechenizkiy, "Joint role and community detection in networks via l 2, 1 norm regularized nonnegative matrix trifactorization," in *Proceedings of the 2019 IEEE/ACM ASONAM*, 2019, pp. 168–175.

[43] L. F. Ribeiro, P. H. Saverese, and D. R. Figueiredo, "struc2vec: Learning node representations from structural identity," in *Proceedings of 23rd ACM SIGKDD KDD*, 2017, pp. 385–394.

[44] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in Neural Information Processing Systems*, vol. 26, 2013.

[45] N. K. Ahmed, R. A. Rossi, J. B. Lee, T. L. Willke, R. Zhou, X. Kong, and H. Eldardiry, "Role-based graph embeddings," *IEEE TKDE*, vol. 34, no. 5, pp. 2401–2415, 2020.

[46] D. Jin, M. Heimann, R. A. Rossi, and D. Koutra, "Node2bits: Compact time-and attribute-aware node representations for user stitching," in *Joint European conference on Machine Learning and Knowledge Discovery in Databases*, 2019, pp. 483–506.

[47] K. Tu, P. Cui, X. Wang, P. S. Yu, and W. Zhu, "Deep recursive network embedding with regular equivalence," in *Proceedings of 24th ACM SIGKDD KDD*, 2018, pp. 2357–2366.

[48] X. Guo, W. Zhang, W. Wang, Y. Yu, Y. Wang, and P. Jiao, "Role-oriented graph role oriented-encoder guided by structural information," in *Database Systems for Advanced Applications: 25th International Conference, DASFAA 2020*, 2020, pp. 466–481.

[49] Y. Jin, G. Song, and C. Shi, "GraLSP: Graph neural networks with local structural patterns," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 4361–4368.

[50] J. Qiu, Q. Chen, Y. Dong, J. Zhang, H. Yang, M. Ding, K. Wang, and J. Tang, "Gcc: Graph contrastive coding for graph neural network pre-training," in *Proceedings of 26th ACM SIGKDD KDD*, 2020, pp. 1150–1160.

[51] M. Scholkemper and M. T. Schaub, "An optimization-based approach to node role discovery in networks: approximating equitable partitions," *Advances in Neural Information Processing Systems*, vol. 36, pp. 71 358–71 374, 2023.

[52] G. Squillace, M. Tribastone, M. Tschaikowski, and A. Vandin, "Approximate regular equivalence by partition refinement," *Applied Network Science*, vol. 10, no. 1, p. 39, 2025. [Online]. Available: https://doi.org/10.1007/s41109-025-00726-7

[53] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.

[54] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th international conference on world wide web*, 2015, pp. 1067–1077.

[55] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, "Asymmetric transitivity preserving graph embedding," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 1105–1114.

[56] G. Squillace, M. Tribastone, M. Tschaikowski, and A. Vandin, "Efficient network embedding by approximate equitable partitions," in *2024 IEEE International Conference on Data Mining (ICDM)*, 2024, pp. 440–449.

[57] L. Cardelli, G. Squillace, M. Tribastone, M. Tschaikowski, and A. Vandin, "Formal lumping of polynomial differential equations through approximate equivalences," *Journal of Logical and Algebraic Methods in Programming*, vol. 134, p. 100876, 2023.

[58] T. P. Peixoto, "The netzschleuder network catalogue and repository," *Zenodo10*, vol. 5281, 2020.

[59] J. Kunegis, "KONECT – The Koblenz Network Collection," in *Proc. Int. Conf. on World Wide Web Companion*, 2013, pp. 1343–1350. [Online]. Available: http://dl.acm.org/citation.cfm?id=2488173

[60] T. Opsahl and P. Panzarasa, "Clustering in weighted networks," *Social Networks*, vol. 31, no. 2, pp. 155–163, 2009.

[61] V. Colizza, R. Pastor-Satorras, and A. Vespignani, "Reaction–diffusion processes and metapopulation models in heterogeneous networks," *Nature Physics*, vol. 3, no. 4, pp. 276–282, 2007.

[62] W. Gu, A. Tandon, Y.-Y. Ahn, and F. Radicchi, "Principled approach to the selection of the embedding dimension of networks," *Nature Communications*, vol. 12, no. 1, p. 3772, 2021.

[63] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.

[64] G. E. Hinton and S. Roweis, "Stochastic neighbor embedding," *Advances in neural information processing systems*, vol. 15, 2002.
[65] S. Tognazzi, M. Tribastone, M. Tschaikowski, and A. Vandin, "Differential equivalence yields network centrality," in *ISOLA*, 2018, pp. 186–201.

Giuseppe Squillace received the Ph.D. degree in Computer Science from IMT School for Advanced Studies Lucca in 2024. He is currently a postdoc at IMT Lucca under the supervision of Mirco Tribastone. His research interests include formal methods, machine learning, optimization, multi-agent systems, and network theory.

Mirco Tribastone is a Professor at IMT Lucca, Italy. Prior to joining IMT Lucca he was Associate Professor at the University of Southampton, UK, and Assistant Professor at the Ludwig-Maximilians University of Munich, Germany. He received his Ph.D. in Computer Science from the University of Edinburgh, UK, in 2010. He graduated in Computer Engineering at the University of Catania, Italy.

Max Tschaikowski is a Poul Due Jensen Associate Professor at Aalborg University, Denmark. Prior to it, he was a Lise Meitner Fellow at TU Wien, Austria, an Assistant Professor at IMT Lucca, Italy, a Research Fellow at the University of Southampton, UK, and a Research Assistant at the Ludwig-Maximilians University in Munich, Germany. He was awarded a Diplom in mathematics (equivalent to a Master) and a Ph.D. in computer science by the LMU in 2010 and 2014, respectively.

Andrea Vandin is an Associate Professor at Sant'Anna School for Advanced Studies, Pisa, Italy, and an Adjunct Associate Professor at DTU Technical University of Denmark. Prior to it he was an Associate Professor at DTU and an Assistant Professor at IMT Lucca, Italy. In 2013-2015 he was a Senior Research Assistant at the University of Southampton, UK. He received his PhD in Computer Science and Engineering from IMT Lucca, Italy. He graduated in Computer Science at the University of Pisa, Italy.