On Proximity and Structural Role-based Embeddings in Networks: Misconceptions, Techniques, and Applications

RYAN A. ROSSI, Adobe Research, USA
DI JIN, University of Michigan, USA
SUNGCHUL KIM, Adobe Research, USA
NESREEN K. AHMED, Intel Labs, USA
DANAI KOUTRA, University of Michigan, USA
JOHN BOAZ LEE, Worcester Polytechnic Institute, USA

Structural roles define sets of structurally similar nodes that are more similar to nodes inside the set than outside, whereas communities define sets of nodes with more connections inside the set than outside. Roles based on structural similarity and communities based on proximity are fundamentally different but important complementary notions. Recently, the notion of structural roles has become increasingly important and has gained a lot of attention due to the proliferation of work on learning representations (node/edge embeddings) from graphs that preserve the notion of roles. Unfortunately, recent work has sometimes confused the notion of structural roles and communities (based on proximity) leading to misleading or incorrect claims about the capabilities of network embedding methods. As such, this paper seeks to clarify the misconceptions and key differences between structural roles and communities, and formalize the general mechanisms (*e.g.*, random walks, feature diffusion) that give rise to community or role-based structural embeddings. We theoretically prove that embedding methods based on these mechanisms result in either community or role-based structural embeddings. These mechanisms are typically easy to identify and can help researchers quickly determine whether a method preserves community or role-based embeddings. Furthermore, they also serve as a basis for developing new and improved methods for community or role-based structural embeddings. Finally, we analyze and discuss applications and data characteristics where community or role-based embeddings are most appropriate.

CCS Concepts: • **Computing methodologies** → **Artificial intelligence**; **Machine learning**; **Logical and relational learning**; • **Mathematics of computing** → **Graph algorithms**; **Approximation algorithms**; *Combinatorics*; *Graph theory*; • **Information systems** → **Data mining**; • **Theory of computation** → **Graph algorithms analysis**.

Additional Key Words and Phrases: Role-based structural embedding, roles, structural similarity, community-based embedding, proximity, role discovery, positions, structural embeddings, structural node embeddings, proximity-based node embeddings, node embeddings, network representation learning, graphlets

ACM Reference Format:
Ryan A. Rossi, Di Jin, Sungchul Kim, Nesreen K. Ahmed, Danai Koutra, and John Boaz Lee. 2020. On Proximity and Structural Role-based Embeddings in Networks: Misconceptions, Techniques, and Applications. *ACM Trans. Knowl. Discov. Data.* 1, 1, Article 1 (January 2020), 36 pages. https://doi.org/10.1145/3397191

1 INTRODUCTION

Motivated by the proliferation of work on node representation learning and the confusion between the notions of communities and roles that existing methods capture, the goal of this manuscript is to clearly define and clarify the differences between community (proximity) and role-based

Authors' addresses: Ryan A. Rossi, Adobe Research, San Jose, CA, USA, rrossi@adobe.com; Di Jin, University of Michigan, Ann Arbor, MI, USA, dijin@umich.com; Sungchul Kim, Adobe Research, San Jose, CA, USA, sukim@adobe.com; Nesreen K. Ahmed, Intel Labs, Santa Clara, CA, USA, nesreen.k.ahmed@intel.com; Danai Koutra, University of Michigan, Ann Arbor, MI, USA, dkoutra@umich.com; John Boaz Lee, Worcester Polytechnic Institute, MA, USA, jtlee@wpi.edu.

2020. 1556-4681/2020/1-ART1 $15.00
https://doi.org/10.1145/3397191

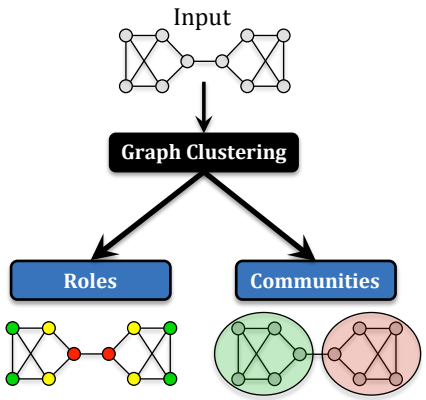


Fig. 1. This taxonomy intuitively illustrates the fundamental differences between the notion of roles (which are based on structural similarity) and communities (based on density, cohesion, and small proximity/distance). Table 1 gives a summary of the key properties of communities and roles captured by embedding methods. Note the input graph is the classical Borgatti-Everett network originally from [Borgatti and Everett 1992].

(structural) embeddings.[1] Towards this goal, we formalize these notions, discuss their differences, and show mathematically how various embedding mechanisms lead to either community or role-based structural embeddings. Given our definitions, we also categorize the existing embedding methods and discuss their suitability for a variety of downstream tasks.

In the following subsections, we begin by introducing the notion of communities and roles, which can be viewed as two different but complementary graph clustering problems. Then, we discuss how these two fundamentally different notions give rise to community and role-based embeddings. At the end of the introduction, we also present the scope of this article, its main contributions, and details about its organization.

1.1 Communities and Roles

Communities and roles lend themselves to many important real-world applications, which are discussed in the seminal survey on communities [Fortunato 2010a; Schaeffer 2007] and roles [Rossi and Ahmed 2015]. They can be viewed as cases of general graph clustering, a problem that is fundamental to the analysis and understanding of graphs. Its main goal is to find a partition of nodes in an input graph. We formalize the general definition of graph clustering as follows.

Definition 1 (Graph Clustering). *A clustering $C = \{C_1, \ldots, C_k\}$ of graph $G = (V, E)$ is a partition of the node set V into non-empty subsets $C_i \subset V$ such that $V = \bigcup_i^k C_i$.*

Definition 1 does not specify the objective of the clustering, but simply that it is a partitioning of the vertex set V into non-empty subsets C_i such that $V = \bigcup_i^k C_i$. Overall, there are two general objectives to graph clustering: (1) communities and (2) roles.

Definition 2 (Communities). *Communities are sets of nodes with more connections inside the set than outside.*[2] *That is, they are* dense cohesive subsets *of vertices $C = \{C_1, \ldots, C_k\}$. A community $C_i \subseteq V$ is "good" if the induced subgraph is dense (*i.e.*, there are many edges between the vertices in C_i) and there are relatively few edges from C_i to other vertices $\bar{C}_i = V \setminus C_i$ [Schaeffer 2007].*

[1] Structural node embeddings and role-based embeddings are used synonymously in the literature. Similarly, proximity-based embeddings and community-based embeddings are also used synonymously.

[2] To capture the general notion of communities (Def. 2), there are many specific objective functions for the quality of communities [Almeida et al. 2011] including modularity [Newman and Girvan 2004], graph conductance [Kannan et al. 2004], among others [Almeida et al. 2011; Emmons et al. 2016].

Table 1. Roles and communities are fundamentally different but complementary notions. Roles and communities are characterized below by their key properties.

Roles	**Communities**
Roles form based on *structural similarity*, *i.e.*, nodes with similar structural properties (*e.g.*, general subgraph patterns/graphlets)	Communities form based on node proximity/closeness (small distance), density, cohesiveness, sparse cuts
Roles are defined by structural properties/features	Communities are defined by node ids and proximity
Roles generalize/transfer across networks (and can be used for graph-based transfer learning tasks) since they are defined by general structural properties/features	Communities do not generalize/transfer across networks (since based on node ids). A community in G has no meaning in another arbitrary graph G'.
Roles characterize nodes that are structurally similar with respect to their general connectivity and subgraph patterns (*e.g.*, graphlets) and are independent of the distance/proximity to one another in the graph [Rossi and Ahmed 2015]. Hence, two nodes assigned to the same role can be in different communities or disconnected components of a single graph, or even different graphs	Nodes in the same community should all be close to one another with small graph distance/proximity [Fortunato 2010a]

Roles were first defined as classes of structurally equivalent nodes [Lorrain and White 1971]. Intuitively, two nodes are structurally equivalent if they are connected to the rest of the network in identical ways. However, structural equivalence is far too strong and restrictive to be useful in practice. Since then there have been many attempts to relax the criterion of equivalence, *e.g.*, regular equivalence [Everett and Borgatti 1994; White and Reitz 1983], stochastic equivalence [Holland and Leinhardt 1981]. For practical purposes, the notion of equivalence can be generally relaxed to get at some form of *structural similarity* [Rossi and Ahmed 2015]. Roles may represent node (or edge) connectivity patterns such as hub/star-center nodes, star-edge nodes, near-cliques or bridge nodes connecting different regions of the graph [Ahmed et al. 2017b; Henderson et al. 2012]. More formally,

Definition 3 (Roles [Rossi and Ahmed 2015]**).** *Roles define sets of nodes that are more structurally similar to nodes inside the set than outside. The term 'structurally similar' refers to nodes that have similar structural properties,* e.g.*, bridge-nodes (gatekeepers) that connect different communities. The terms role and position are used interchangeably.*[3]

In this work, the term *structural similarity* (Definition 3) is reserved for the notion of roles as it implies nodes that have similar structural properties whereas the term proximity and density are reserved for communities.

Based on the definitions above, roles are complementary but fundamentally different to the notion of communities. An intuitive example is shown in Figure 1 and their key differences are summarized in Table 1. While communities capture cohesive/tightly-knit groups of nodes and nodes in the same community are close together (small graph distance or high proximity) [Fortunato 2010a], roles characterize nodes that are structurally similar with respect to their general connectivity and subgraph patterns, and are independent of the distance/proximity to one another in the graph [Rossi

[3] An equivalent definition of role is in terms of a role assignment function $r : V \rightarrow R$ that maps nodes to a set of roles R. The role assignment function r induces a partition $C = \{C_1, \ldots, C_k\}$ of V by taking the inverse-images as sets/classes of nodes that play/have the same role. Further, if $\sim$ is an equivalence relation (binary relation on V that is reflexive, symmetric, and transitive), then the set of its equivalence classes is a partition of V (and conversely). Hence, it is equivalent to think of a role as a set of nodes (node partition), function (role assignment), or equivalence relation on V since these are just different, but equivalent mathematical formulations for the concept of roles.

and Ahmed 2015]. Hence, two nodes that share similar roles (*e.g.*, star-center nodes) can be in different communities and even in two disconnected components of the graph. Another difference is that communities are defined on a particular graph, whereas roles capture a more general notion that represents structural patterns and are able to generalize across networks *i.e.*, they can be learned on one network, and transferred to another, whereas communities do not [Henderson et al. 2012; Rossi and Ahmed 2015]. For instance, roles based on graphlets (or any structural features) can be naturally transferred to another graph as they are essentially general structural graph functions that can be computed on any arbitrary graph, independent of the specific nodes. We refer the interested reader to Section 5 for further details and examples.

1.2 Community- and Role-based Embeddings

Recently, there has been substantial work on learning node embeddings.[4] Learning an appropriate feature-based representation of the graph (*e.g.*, node/edge embeddings) lies at the heart and success of many graph-based machine learning tasks. In particular, they have proven to be important for many application tasks including node and link classification [Jin et al. 2019b; McDowell et al. 2009; Neville and Jensen 2000; Rossi and Neville 2012; Sen et al. 2008], link prediction [Al Hasan and Zaki 2011; Bilgic et al. 2007; Grover and Leskovec 2016; Perozzi et al. 2014; Rossi et al. 2017], regression [Gleich and Rossi 2014], anomaly detection [Akoglu et al. 2015; Jin et al. 2019c], dynamic network analysis [Kovanen et al. 2011; Nguyen et al. 2018; Nicosia et al. 2013], metric learning [Lee et al. 2020; Ma et al. 2019b], few-shot learning [Satorras and Estrach 2018], entity resolution/visitor stitching [Gilpin et al. 2013; Jin et al. 2019a; Rossi et al. 2018b], activity discovery [Safavi et al. 2020], visualization and sensemaking [Fang et al. 2017; Pienta et al. 2015; Rossi et al. 2018a], compression/graph summarization [Ahmed et al. 2018; Jin et al. 2019c; Liu et al. 2018a], network alignment [Heimann et al. 2018; Koyutürk et al. 2006], graph similarity [Ma et al. 2019a], and graph classification [Heimann et al. 2019; Nikolentzos et al. 2017; Yan et al. 2019; Ying et al. 2018].

While communities and roles are useful in themselves for many different and complementary applications, they have also become fundamentally important for learning embeddings that preserve the notion of community (proximity) or roles (structural similarity). Indeed, many works claim to preserve the notion of communities [Cavallari et al. 2019, 2017; Wang et al. 2017], roles [Jin et al. 2019b; Ribeiro et al. 2017; Rossi et al. 2018b], or even both [Grover and Leskovec 2016]. The embedding/feature vectors given as output from an embedding method can be thought of as either community [Fortunato 2010a; Henderson et al. 2010] or role membership vectors (assuming proper normalization) [Airoldi et al. 2008; Gilpin et al. 2013; Rossi et al. 2013]. In this light, recent embedding methods can be seen as approaches for modeling communities *or* (feature-based) roles [Rossi and Ahmed 2015]. For simplicity, we have described roles and communities in Section 1.1 with respect to hard assignments. However, more than a decade ago, methods for roles and communities that naturally output node embeddings have been investigated. The node embeddings from these methods are sometimes referred to as node mixed-membership vectors. In other words, community and role discovery are not different problems than node embeddings, since they both output node embeddings.

More recently, there has been an upsurge of interest in learning node embeddings that preserve structural roles [Rossi and Ahmed 2015] (as opposed to communities based on proximity). These works often claim to preserve structural equivalence [Lorrain and White 1971] or regular equivalence [Sailer 1978; White and Reitz 1983]. However, since the output of these methods are embeddings and not roles, these classical definitions are not appropriate since they are defined formally with respect to the graph as shown in Section 3.2. In particular, methods used to find

[4]Embedding, features, and representation are considered synonymous and used interchangeably throughout this manuscript.

role assignments that are regularly equivalent (or that preserve some other form of graph-based equivalence) use the graph directly and *not* embeddings/features. The recent work that learns embeddings is therefore more closely related to feature-based roles proposed by Rossi and Ahmed [2015]. For instance, instead of leveraging the graph directly, feature-based roles are assigned based on (structural) feature representations (node embeddings) that appropriately describe the structural characteristics of the nodes in the graph. Thus, the question becomes: given node embeddings learned from some method, do the embeddings preserve a feature/embedding-based role equivalence (or more generally, structural similarity) or do they preserve proximity (density, communities)?

Understanding whether a method preserves communities (proximity) or roles (structural similarity) helps identify and understand the applications and tasks where the embeddings might be useful. For instance, if we know a method outputs embeddings that capture communities better, then we can already begin to understand the types of applications where such embeddings are likely to perform well, *e.g.*, community-based embeddings work well for node classification on graphs with homophily (*i.e.*, neighbors of a node are more likely to share the same label than not) [La Fond and Neville 2010] whereas role-based embeddings work better for graphs with weak homophily or even heterophily. We discuss applications of community and role-based embeddings in Section 6.

1.3 Scope of this Article

This article formalizes the general mechanisms (techniques) that lie at the heart of nearly all existing embedding methods. We show that these general mechanisms give rise to methods that learn community/proximity-based embeddings (Section 4) or role-based structural embeddings (Section 5). See Table 2 for a summary of the key mechanisms behind community and structural role-based embeddings. Moreover, we clarify the misconceptions surrounding these two different notions of embeddings. This paper is not a survey of the abundance of work on communities [Fortunato 2010a; Schaeffer 2007] or roles [Rossi and Ahmed 2015], nor do we attempt to survey the abundance of work on graph embeddings/relational representation learning [Cai et al. 2018; Goyal and Ferrara 2018; Rossi et al. 2012; Wu et al. 2019a; Zhang et al. 2018].

1.4 Main Contributions

The main contributions of this work are:

- Formalizing the notion of communities and roles; and clarifying their fundamental differences
- Proposing *embedding-based node equivalences* for roles that are defined with respect to the learned node embeddings (feature vectors) as opposed to a graph G as traditionally done.
- Formalizing the general mechanisms that give rise to community or role-based structural embeddings. These mechanisms are typically easy to identify and can help researchers understand whether a method learns community or structural role-based embeddings. Furthermore, they can also be used to develop new and better community or role-based embedding methods.
- Theoretically demonstrating that embedding methods based on these mechanisms result in either community (proximity) or role-based structural embeddings.
- Categorizing embedding methods into community or role-based by highlighting the general mechanism used by it and why it gives rise to such embeddings.
- Analyzing and discussing the applications and data characteristics/assumptions where community-based or role-based embeddings are most appropriate.

Table 2. General mechanisms that give rise to community (proximity) *or* structural role-based embeddings.

Embedding Type	General Mechanism	Examples of Methods
Community-based (Section 4)	**Random Walks** (Sec. 4.1)	Spectral embedding [Chung 1997] deepwalk [Perozzi et al. 2014] node2vec [Grover and Leskovec 2016] LINE [Tang et al. 2015] GraRep [Cao et al. 2015] ComE+ [Cavallari et al. 2019]
	Feature Prop./Diffusion (Sec. 4.2)	GCN [Kipf and Welling 2017] GraphSage [Hamilton et al. 2017] MultiLENS [Jin et al. 2019c]
Role-based (Section 5)	**Graphlets** (Sec. 5.1)	deepGL [Rossi et al. 2017] MCN [Lee et al. 2018b] HONE [Rossi et al. 2018b]
	Feature-based Walks (Sec. 5.2)	role2vec [Ahmed et al. 2018] node2bits [Jin et al. 2019a] SimSum [Liu et al. 2018b, 2019]
	Feature-based MF (Sec. 5.3)	rolX [Henderson et al. 2012] HERO [Ahmed et al. 2017b] EMBER [Jin et al. 2019b]

1.5 Organization of this Article

The article is organized as follows: We first discuss background and preliminaries in Section 2. In Section 3, we formalize the notion of communities and roles, discuss issues relating to false claims about these notions, and propose new feature/embedding-based equivalences for embedding methods. In Sections 4 and 5, we formally describe the general mechanisms that lie at the heart of nearly all existing embedding methods. We show that these general mechanisms give rise to methods that learn community/proximity-based embeddings (Section 4) or role-based structural embeddings (Section 5). For the general mechanisms behind community or role-based structural embeddings (summarized in Table 2), we theoretically demonstrate why they are community-based or role-based. Section 6 discusses applications and the specific settings (*e.g.*, data characteristics, problem setting/constraints) that are well-suited for community or role-based embedding techniques. Finally, Section 7 concludes.

2 PRELIMINARIES

Given a graph $G = (V, E)$ where V represents the set of nodes and E represents the set of edges, we define node and edge embedding as follows:

Definition 4 (Node Embedding). *Node embedding aims to learn a function $f : V \rightarrow \mathbb{R}^k$ that maps each node to a k-dimensional embedding vector $\mathbf{x}$ where $k \ll |V|$.*

Definition 5 (Edge Embedding). *Edge embedding aims to learn a function $f : E \rightarrow \mathbb{R}^k$ that maps an edge (node pair) to a k-dimensional embedding vector $\mathbf{x}$ where $k \ll |E|$.*

Let $\mathbf{X}$ denote the node (or edge) embedding/feature matrix where the rows represent nodes (or edges) and the columns represent (latent) features. Hence, $\mathbf{x}_i$ is the k-dimensional embedding/feature vector for the i-th node (edge).

Many existing works derive edge embeddings based on the learned low-dimensional representations of nodes [Grover and Leskovec 2016; Perozzi et al. 2014] through element-wise operators such as *average, Hadamard*, etc., so we categorize them together. Approaches such as DeepGL [Rossi et al. 2017] that can learn edge embeddings directly from the graph are called *direct edge embedding methods* and are discussed separately.

There is also a line of works that aim to learn an embedding vector for an entire graph:

Definition 6 ((Whole-) Graph Embedding). *Given a set of graphs, the goal is to learn a function $f : \mathcal{G} \rightarrow \mathbb{R}^k$ that maps an entire input graph $G \in \mathcal{G}$ to a low-dimensional embedding vector $\mathbf{z}$ of length k where $\mathcal{G}$ is the input space of graphs. Similar graphs (*e.g.*, graphs belonging to the same class) should be embedded close to one another in the low k-dimensional space.*

Some existing works in the literature aim to embed an induced subgraph such as the subgraph rooted at a specific node [Narayanan et al. 2016, 2017]. These methods can be easily applied to embed the entire graph by treating the input as a subset of the union of all graphs. The graph embeddings can then be used as input for downstream applications such as graph classification [Lee et al. 2018a], regression [Duvenaud et al. 2015], and anomaly detection [Hu et al. 2016]. We refer interested readers to the comprehensive review on the traditional graph embedding methods [Fu and Ma 2012].

3 COMMUNITIES AND STRUCTURAL ROLES

This section formally defines the notions of communities and structural roles. We then discuss and summarize the fundamental differences between these notions. Despite the various applications and practical importance, the notion of *structural roles* has only received a limited amount of attention [Rossi and Ahmed 2015] compared to communities [Backstrom et al. 2006; Chakrabarti et al. 2006; Chen and Saad 2010; Newman 2004; Schaeffer 2007]. As such, we discuss roles in significantly more detail and show that classical node equivalences for assigning roles with respect to G are inappropriate for embeddings (by definition).

3.1 Communities

While there are many different methods for finding communities (*e.g.*, modularity maximization [Newman and Girvan 2004], cut-based methods [Kannan et al. 2004], among others [Almeida et al. 2011; Emmons et al. 2016]), it is generally agreed that a subset of vertices $S \subseteq V$ is a "good" community if the induced subgraph is dense (*e.g.*, many edges between the vertices in S) and there are relatively few edges from S to the other vertices $\bar{S} = V \setminus S$ [Schaeffer 2007]. Let $E(S)$ denote the set of edges between vertices in S (internal edges) and $E(S, \bar{S})$ be the set of edges between S and $\bar{S}$ (cut set, that is, the set of edges that if removed would disconnect S from $\bar{S}$). Note $E(S, \bar{S})$ is the set of external edges, that is, the set of edges that have their origin in S and destination in $\bar{S}$. Clearly, $|E(S, \bar{S})|$ should be small relative to $|E(S)|$ and $E(\bar{S})$ for any "good" and reasonable community detection method. An ideal situation is when communities are disjoint cliques. A summary of the key properties of communities are as follows:

(1) **Densely connected**: Nodes inside a community are more densely connected to nodes within the community than nodes in another community (By Definition 2).
(2) **Proximity/closeness**: Nodes in the same community are close to one another in the graph in terms of distance/proximity.
(3) **Walks**: Nodes in the same community have more walks to one another (*i.e.*, ways of going from node i to j) compared to nodes outside the community.

Intuitively, both density and proximity are also closely related. For instance, the more dense a community is in terms of the number of edges between nodes within the community, the more

close (shorter distance/more walks) the nodes must be in the community, and vice-versa. Both of these properties also imply more walks between nodes in the same community compared to nodes in another community. See the seminal survey by Schaeffer [2007] for discussion on other properties. Note the term community-based embeddings and proximity-based embeddings are used interchangeably in the literature.

3.2 Structural Roles

We first review and discuss the classical node equivalences used for structural roles. These classical node equivalences (*e.g.*, structural and regular equivalence [Luczkovich et al. 2003]) are defined directly on the graph G as opposed to the node embedding/feature matrix $\mathbf{X}$. Algorithms that return a structurally equivalent (or regularly equivalent) role assignment are known and have been widely studied [Lorrain and White 1971; White and Reitz 1983]. However, since these node equivalences are defined with respect to G and not embeddings, they cannot be used on embeddings/feature vectors, despite such claims in recent work.

In this work, we formalize the notion of *structural similarity* and introduce node equivalences that *can* be used on an embedding/feature representation of G. These embedding-based node equivalences serve two main purposes. First, they allow us to precisely formulate the notion of role mathematically which can be used to understand and theoretically analyze embedding methods and their representational power. Second, algorithms based on these notions of feature-based node equivalences can be developed to find such roles.

We start by defining a role assignment, which is used later in the formalization of the different graph-based role equivalences (*e.g.*, Definitions 8-9) and embedding-based role equivalences (Definitions 12-14) that lie at the heart of structural roles. In particular, a role assignment r of V is a surjective mapping $r : V \rightarrow R$ onto a set R of roles. An assignment r defines a partition[5] $\mathcal{C} = \{C_1, \ldots, C_k\}$ of V by taking the inverse-images as classes, *i.e.*, $r^{-1}(s) = \{i \in V \mid r(i) = s\}, \forall s \in R$. Clearly, any role assignment r is not useful alone, unless it satisfies one of the equivalences defined later in this section. For instance, if r satisfies regular equivalence (Definition 9), then we say the role assignment r is regularly equivalent.

Given a node embedding/feature matrix $\mathbf{X}$ from an arbitrary embedding/representation learning method f, we can find a role assignment $r : V \rightarrow R$ *indirectly* using another method $\mathcal{M}$ (*e.g.*, an approach that partitions nodes into disjoint sets $\mathcal{C} = \{C_1, \ldots, C_k\}$ such as k-means, or assigns nodes to classes). However, since the role assignment r is given by $\mathcal{M}$ and not f, it is difficult (if not impossible) to make any claims about f with respect to classical notions of node equivalence such as structural (Definition 8) or regular equivalence (Definition 9). In other words, there is no guarantee that a role assignment $r : V \rightarrow R$ is structurally or regularly equivalent since it is computed by some method $\mathcal{M}$ based on the learned node embedding/feature matrix $\mathbf{X}$ only, despite the fact that the definitions of structural and regular equivalence involve the graph G only (and more specifically, the neighbors of nodes) and not $\mathbf{X}$. Hence, the method $\mathcal{M}$ used to assign roles r using $\mathbf{X}$ would need to also consider the graph G to guarantee that such a role assignment is structurally equivalent or regular equivalent; and it is unclear that $\mathbf{X}$ would actually be useful in finding such a role assignment, since (efficient) algorithms that return such role assignments using only G have been known for years [Borgatti et al. 2018; Boyd and Everett 1999; Everett and Borgatti 1991; Lorrain and White 1971; White and Reitz 1983]. In other words, there is no guarantee that the role assignments r given by $\mathcal{M}$ based on node embeddings $\mathbf{X}$ from f will satisfy a graph-based equivalence such as structural equivalence or regular equivalence. However, given a role assignment r and a node equivalence (*e.g.*, structural or regular equivalence), we can verify if

[5]Recall a partition $C = \{C_1, \ldots, C_k\}$ is a set of non-empty, disjoint subsets $C_i \subset V$ such that $V = \bigcup_{i=1}^{k} C_i$.

the role assignment r is a valid and proper assignment under the chosen node equivalence (*i.e.*, the node equivalence holds for r). Though, as mentioned above, this is not useful since there are methods to find such role assignments directly from the graph G, and in practice, such strict role assignments are typically not very useful.

Definition 7 (Role graph). *Let $G = (V, E)$ be a graph with role assignment $r : V \to R$, then $G_R = (R, E_R)$ is the role graph with vertex set R (roles) and edge set $E_R \subseteq R \times R$ defined as:*

$$E_R = \left\{ (r_i, r_j) \,|\, (i, j) \in E \right\} \tag{1}$$

where G_R succinctly models the roles and the relationships between the roles R. The role graph summarizes the essential structural properties of G and thus can be viewed as a form of compression.

Role assignment definitions translate to partitions and equivalence relations for roles.

Definition 8 (Structural equivalence). *Let $G = (V, E)$ be a graph and $r : V \to R$ be a role assignment, then r is* strong structural *if equivalent vertices have the same neighbors. More formally, if $\forall i, j \in V$,*

$$r_i = r_j \implies N_i^+ = N_j^+ \wedge N_i^- = N_j^- \tag{2}$$

where N_i^+ and N_i^- are the out and in neighbors of i. In other words, two nodes are structurally equivalent iff they are connected to the same neighbors. Structural equivalence can only identify nodes close to each other in G.

A few trivial examples of structurally equivalent nodes are star-edge nodes of a k-star graph, nodes in a k-clique graph, or nodes in a complete bipartite graph. Structural equivalence is computationally and theoretically trivial. It is far too strict for general graphs and only nodes with distance at most two can be identified by it. This has led to many slightly more useful relaxations including regular equivalence:

Definition 9 (Regular equivalence). *Let $r(W) = \{r(i) \,|\, i \in W\}$ be the* role set *of $W \subseteq V$. A role assignment $r : V \to R$ is* regular *if $\forall i, j \in V$*

$$r_i = r_j \implies r(N_i^+) = r(N_j^+) \wedge r(N_i^-) = r(N_j^-) \tag{3}$$

where $r(N_i^+)$ is the set of roles *from the neighbor set N_i^+. Hence, i and j are* regularly equivalent *iff they are connected to the same role equivalent neighbors (*i.e.*, have the same set of roles from their neighbors).*

Definition 10 (Exact role assignment). *A role assignment $r : V \to R$ is exact iff*

$$r_i = r_j \implies r(N_i) = r(N_j) \tag{4}$$

where $N_i = \{j \in V \,|\, (i, j) \in E\}$ and $r(N_i)$ is the multi-set *of roles from set of neighbors N_i. Hence, nodes of the same role must contain the same number of each of the other roles in their neighborhood.*

Note the slight abuse of $r(N_i)$ in Definition 11 to mean the *multi-set* of roles from N_i whereas in regular equivalence (Definition 9) and elsewhere it is simply the *set* of roles. An example of an exact role assignment is shown in Figure 1.

Definition 11 (Strong structural role assignment). *Let $G = (V, E)$ and $G_R = (V_R, E_R)$ denote the role graph. A role assignment $r : V \to R$ is strong structural iff for all $i, j \in V$, there exists an edge (r_i, r_j) in the role graph G_R and $(i, j) \in E$.*

All of the classical node equivalences are defined strictly using the graph G alone, and not defined with respect to embeddings/features. For instance, the formal definitions of structural and regular equivalence (Definition 8-9) involve only the sets of neighbors of nodes in G, and algorithms for

finding such a role assignment r that is structurally or regularly equivalent are known and require only the graph G since that is all that is necessary by Definition 8-9. For a summary of other classical graph-based node role equivalences that have the same issues, see [Rossi and Ahmed 2015]. Nevertheless, all such classical node equivalences that are formally defined *w.r.t.* the graph G are obviously not helpful for assigning roles based on embeddings/features. Furthermore, given node embeddings from any method, it is also not possible to use these classical graph-based node equivalences to make claims on whether the embeddings preserve the equivalence or not (by definition).

From Equivalences on the Graph to Equivalences on Embeddings. While the classical node equivalences are defined *w.r.t.* the graph G (and thus not useful for embeddings), we now introduce the notion of an *embedding-based node equivalence* defined *w.r.t.* node embeddings (as opposed to graph-based node equivalences discussed previously). More importantly, we formalize the notion of structural similarity that serves as a basis for defining new node equivalences that involve node embeddings (features). We begin by defining an equivalence relation for features:

Definition 12 (Feature-based/embedding equivalence). *A feature(embedding)-based equivalence is an equivalence relation $\sim$ between two structural feature/embedding vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ for i and j.*

One of the most strict notions of node equivalence on embeddings/feature representation is *feature-based structural equivalence* defined as:

Definition 13 (Feature-based Structural Equivalence [Rossi and Ahmed 2015]). *Let $\mathbf{X}$ be a **structural** embedding/feature matrix (*e.g.*, features/embeddings that capture structural properties such as in/out/total degree, k-stars, k-paths, and other subgraph patterns/graphlets, etc). A role assignment $r : V \rightarrow R$ is called* feature-based structurally equivalent *if for all $v_i, v_j \in V$:*

$$r_i = r_j \implies \left[\forall k, 1 \leq k \leq d : x_{ik} = x_{jk}\right] \tag{5}$$

*and $\mathbf{X}$ are proper structural properties (and not based on proximity/cohesion) where x_{ik} is the k-*th *feature value of node i. Eq. 5 is strict since two nodes belong to the same role iff they have identical feature vectors.*

Notice a key difference between feature-based equivalences and the other graph-based equivalences is that there is no requirement on the *neighbors* of a node. This avoids roles being tied to one another based on proximity (cohesion/distance in the graph). A more practical notion of roles is called *feature-based structural similarity* [Rossi and Ahmed 2015] and is a relaxation of the notion of feature-based structural equivalence (Definition 13). This notion replaces the strict requirement that nodes in the same roles have identical feature vectors by the requirement that nodes in the same roles must be *ϵ-structurally similar* for any ϵ. More formally,

Definition 14 (Feature-based Structural Similarity). *Let $\mathbf{x}_i$ and $\mathbf{x}_j$ be structural embeddings and $\mathbb{K}$ be a similarity function, then we say node i and j are **ϵ-structurally similar** for any $\epsilon > 0$ iff*

*(1) $\mathbf{x}_i$ and $\mathbf{x}_j$ encode structural properties of i and j in G (*e.g.*, "role-based" features related to whether i is on the periphery, star-edge, star-center/hub, bridge, near-clique, and so on)*

(2) $\mathbf{x}_i$ and $\mathbf{x}_j$ are not correlated with graph proximity and/or density (communities) in G

(3) $\mathbb{K}(\mathbf{x}_i, \mathbf{x}_j) \geq 1 - \epsilon$ for any $\epsilon > 0$

Intuitively, i and j are "equivalent" (which implies a partitioning/role assignment) iff $\mathbb{K}(\mathbf{x}_i, \mathbf{x}_j) \geq 1 - \epsilon$ (*i.e.*, they are ϵ-structurally similar) and the features/embeddings are strictly structural

and representative of the structural properties/topology in G and not based on communities (*i.e.*, proximity/closeness and density).

One can also add additional conditions to Definition 14 to make it more strict (and possibly more useful for certain applications). For instance, we can add an additional constraint that neighbors must not be of the same role. While this constraint will strengthen condition 2 of Definition 14, it will also impact the roles we are able to capture, *e.g.*, we would be unable to capture roles of nodes that represent near-cliques. However, such a constraint will ensure that the features/embeddings do not simply capture communities, since if they did, then neighbors would likely be assigned to the same role. We can relax this further by allowing one such role to have neighbors of the same role (*e.g.*, to capture near-clique role).

3.3 Discussion

In the context of embeddings, community-based methods embed nodes that are close in the graph (proximity, density) in a similar fashion whereas structural role-based methods embed nodes that are structurally similar (based on structural properties such as graphlets) such that *structurally similar* nodes are close in some low k-dimensional space. In some papers, there are misleading or false claims made about the resulting node embeddings. For instance, some existing work claims that the resulting embeddings preserve the notion of structural equivalence or even regular equivalence. Unfortunately, none of these notions are defined on the level of embeddings/features. In fact, structural and regular equivalence are defined with respect to a graph G and not node embeddings $\mathbf{X}$ of the graph. Thus it is impossible to apply such equivalences or make claims about such equivalences with respect to any arbitrary embeddings as shown and discussed in Section 3.2. As an aside, roles and communities can also be defined to allow a node or edge to belong to multiple roles and communities. Typically, each node (edge) i is assigned a weight x_{ik} indicating the membership to the k^{th} cluster (community or role). These are typically referred to as role or community membership models. Overlapping communities is a special case of the above.

In Table 3, the general mechanisms behind community and role-based structural embeddings are summarized (which are discussed next in Section 4-5) along with a few representative embedding methods from each of the mechanisms as well as the input and output of the community and role-based embedding methods. Recall from Section 1.3 that this paper is not a survey of such embedding methods (as this is outside the scope of this work), and thus, Table 3 shows only a few methods from each community and role-based mechanism. Moreover, the vast majority of embedding methods are community-based and thus, if we included all such methods, it would be highly skewed towards community-based embeddings. As an aside, a recent work [Srinivasan and Ribeiro 2019] claims to show the equivalence between what they call node embeddings and (structural) graph representations.[6] However, the definitions introduced in that work deviate fundamentally from the definitions (and terminology) used in the literature and in this paper. Therefore, while the problems studied in [Srinivasan and Ribeiro 2019] sound related, they are fundamentally different, and the corresponding findings and conclusions should not be confused.

4 COMMUNITY-BASED EMBEDDING

We discuss the two main general mechanisms behind existing *community-based embedding methods*, namely, random walks (Section 4.1) and feature propagation/diffusion (Section 4.2).

[6]The term node embedding and representation are typically used interchangeably in the literature.

Table 3. Qualitative and quantitative comparison of a few representative community and role-based graph embedding methods from each of the general mechanisms.

			Input						Output			Mechanism				
Method	**Community-based**	**Role-based**	**Homogeneous graph**	**Heterogeneous graph**	**Temporal graph**	**Weighted graph**	**Directed graph**	**Features/attributes**	**Node embedding**	**Direct edge embedding**	**Graph embedding**	**Random walks**	**Feature Prop./Diffusion**	**Graphlet/Motif-based**	**Feature-based Walks**	**Feature-based MF**
DeepWalk [Perozzi et al. 2014]	✓	✗	✓	✗	✗	✓	✓	✗	✓	✗	✗	✓	✗	✗	✗	✗
Node2vec [Grover and Leskovec 2016]	✓	✗	✓	✗	✗	✓	✓	✗	✓	✗	✗	✓	✗	✗	✗	✗
Metapath2vec [Dong et al. 2017]	✓	✗	✓	✓	✗	✓	✗	✗	✓	✗	✗	✓	✗	✗	✗	✗
CTDNE [Nguyen et al. 2018]	✓	✗	✓	✗	✓	✓	✓	✗	✓	✗	✗	✓	✗	✗	✗	✗
LINE [Tang et al. 2015]	✓	✗	✓	✗	✗	✓	✗	✗	✓	✗	✗	✓	✗	✗	✗	✗
S2S-AE [Taheri et al. 2018]	✓	✗	✓	✓	✗	✓	✗	✗	✗	✗	✓	✓	✗	✗	✗	✗
GraRep [Cao et al. 2015]	✓	✗	✓	✗	✗	✓	✗	✗	✓	✗	✗	✓	✗	✗	✗	✗
ComE+ [Cavallari et al. 2019]	✓	✗	✓	✗	✗	✓	✗	✗	✓	✗	✗	✓	✗	✗	✗	✗
GCN [Kipf and Welling 2017]	✓	✗	✓	✗	✗	✓	✗	✓	✓	✗	✗	✗	✓	✗	✗	✗
graphSAGE [Hamilton et al. 2017]	✓	✗	✓	✗	✗	✓	✗	✓	✓	✗	✗	✗	✓	✗	✗	✗
MultiLENS [Jin et al. 2019c]	✓	✗	✓	✓	✗	✓	✓	✗	✓	✗	✗	✗	✓	✗	✗	✗
DeepGL [Rossi et al. 2017]	✗	✓	✓	✗	✗	✓	✓	✓	✓	✓	✗	✗	✗	✓	✗	✗
MCN [Lee et al. 2018b]	✗	✓	✓	✗	✗	✓	✓	✓	✓	✗	✗	✗	✗	✓	✗	✗
HONE [Rossi et al. 2018b]	✗	✓	✓	✓	✗	✓	✓	✓	✓	✗	✗	✗	✗	✓	✗	✗
role2vec [Ahmed et al. 2018]	✗	✓	✓	✗	✗	✓	✓	✓	✓	✗	✗	✗	✗	✓	✓	✗
node2bits [Jin et al. 2019a]	✗	✓	✓	✓	✗	✓	✓	✓	✓	✗	✗	✗	✗	✗	✓	✗
rolX [Henderson et al. 2012]	✗	✓	✓	✗	✗	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✓
GLRD [Gilpin et al. 2013]	✗	✓	✓	✗	✗	✓	✗	✓	✓	✗	✗	✗	✗	✗	✗	✓
DBMM [Rossi et al. 2013]	✗	✓	✓	✗	✓	✓	✗	✓	✓	✗	✗	✗	✗	✗	✗	✓
struc2vec [Ribeiro et al. 2017]	✗	✓	✓	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓
xNetMF [Heimann et al. 2018]	✗	✓	✓	✗	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗	✓
EMBER [Jin et al. 2019b]	✗	✓	✓	✗	✗	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✓
HERO [Ahmed et al. 2017b]	✗	✓	✓	✗	✓	✓	✗	✓	✓	✓	✗	✗	✗	✓	✗	✓

4.1 Random Walks

Random walks have been used as a basis in many community detection methods [Van Dongen 2000] for decades [Fortunato 2010b]. Recent embedding approaches are based on traditional random walks and thus are *unable* to capture roles (structural similarity) and instead capture the notion of communities [Cavallari et al. 2019, 2017; Dong et al. 2017; Grover and Leskovec 2016; Perozzi et al. 2014] as shown later in this section. Hence, these methods learn community/proximity-based embeddings, as opposed to structural role-based embeddings. In particular, these methods embed nodes that are close to one another in the graph in a similar way and therefore are largely capturing the notion of communities as opposed to roles. Recent empirical analysis shows that using random walks for embeddings primarily capture proximity among the vertices (see Goyal and Ferrara [2018]), so that vertices that are close to one another in the graph (in terms of distance)

are embedded together, *e.g.*, vertices that belong to the same community are embedded similarly. In contrast, random walks will likely visit nearby vertices first, which makes them suitable for finding communities (based on proximity/density), rather than roles (structural similarity). In fact, random walks are fundamental to many important community detection methods [Schaeffer 2007; Van Dongen 2000]. Indeed, components of the eigenevector corresponding to the second eigenvalue of the transition matrix of a random walk on a graph provide proximity measures that indicate how long it takes for a walk to reach each vertex. Obviously, vertices in the same community should be quickly reachable. Furthermore, a random walk starting from a vertex in one community is more likely to remain in the community than to move to another community. This is precisely the reason that random walks and communities are very closely related.

The normalized cut of a graph (used for community detection) can be expressed in terms of the transition probabilities and the stationary distribution of a random walk in the graph [Ahmed et al. 2018; Chung 1997; Dong et al. 2016; Meila and Shi 2001a,b; Orponen and Schaeffer 2005; Orponen et al. 2008]. This formally links the mathematics of random walks to cut-based community detection methods. Thus, random walks and communities are fundamentally tied.

The connection between walk-based embeddings and communities was formally shown by Ahmed et al. [2018]. We summarize it below. Suppose $\mathbf{P}$ is the transition matrix defined as

$$\mathbf{P}(u,v) = \begin{cases} \frac{1}{d_u} & \text{if } u \text{ and } v \text{ are adjacent} \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

The probability that a random walk $W(u)$ starting at u visits a vertex x at time i is $\mathbf{e}_u\mathbf{P}^i\mathbf{e}_x^T$ where $\mathbf{e}_x$ is the unit vector having 1 in coordinate x and 0 in every other coordinate. For a directed edge (u,v), the probability that a random walk $W(x)$ visits u at time i and then visits v at time $i+1$ can be denoted as

$$\frac{\mathbf{e}_x\mathbf{P}^i\mathbf{e}_u^T}{d_u}$$

Given an edge $(u,v) \in E$, let $I(u,v)$ denotes the total number of walks containing it. With d_v walks (each of which is of length l) starting at v, the sum of the probabilities that there exists a walk $W(x)$ that visits (u,v) is

$$\begin{aligned} I(u,v) &\leq \sum_{i=0}^{l-1}\sum_{x} d_x\mathbf{e}_x\mathbf{P}^i\mathbf{e}_u^T/d_u \\ &= \sum_{i=0}^{l-1}\mathbf{1}\mathbf{D}\mathbf{P}^i\mathbf{e}_u^T/d_u = \sum_{i=0}^{l-1}\mathbf{1}\mathbf{D}\mathbf{e}_u^T/d_u = \sum_{i=0}^{l-1}1 = l \end{aligned}$$

where $\mathbf{D}$ is the degree matrix $\mathbf{D}(u,u) = d_u$, $\mathbf{1}$ is the all-one vector and $\mathbf{1}\mathbf{D}\mathbf{P}^i = \mathbf{1}\mathbf{D}$. Therefore, if we start d_u random walks from $u \in V$, the expectation of $I(u,v)$ is no more than l.

Let C denote a community in the graph, $u, v \in C$ and $v' \in \bar{C} = V \setminus C$. The probability of the random walker staying in its community in the next step is:

$$\mathbf{P}(u,C) = \sum_{v\in C}\mathbf{P}(u,v) = \frac{d_{uC}}{d_u} \tag{7}$$

where d_{uC} denotes the number of edges originating from u within the same community. Similarly, the probability of leaving the community is $\sum_{v'\in\bar{C}}\mathbf{P}(u,v') = \frac{d_{u\bar{C}}}{d_u}$. By Definition 2, communities are densely connected with few edges across communities, which implies that at time i the probability of the random walker staying in the same community is larger than reaching nodes outside the community, *i.e.*

$$d_{uC} > d_{u\bar{C}} \Rightarrow \mathbf{P}(u,C) > \mathbf{P}(u,\bar{C}) \quad \forall u \in C \tag{8}$$

The above notion only reflects that at any time i, the random walk is more likely to sample neighbors of the same community. In order to measure the probability that all elements in a random walk stay in the same community, we introduce volume and conductance.

Definition 15. *Given a set of nodes $C \in V$ (partition of V), the volume of C is $\mu(C) = \sum_{v \in C} d_v$. The conductance can then be computed as the ratio of its external edges over the minimum of $\mu(C)$ and $\mu(\bar{C})$:*

$$\Phi(C) = \frac{|E(C, \bar{C})|}{\min(\mu(C), \mu(\bar{C}))} \tag{9}$$

where $|E(C, \bar{C})|$ denotes the number of external edges between C and $\bar{C} = V - C$

As shown by Spielman and Teng [2013], the probability that an ℓ-step walk starting from a random vertex in C stays entirely in C is bounded by

$$P_C^\ell \geq 1 - \frac{\ell \Phi(C)}{2} \tag{10}$$

This implies that if $\Phi(C)$ is small (which is usually the case for "good commumities"), then the ℓ-step walk will stay inside C with fairly high probability. Further, while the probability to "escape" the community increases with longer walks, ℓ has to be comparable to $\frac{1}{\Phi(C)}$, which is generally a large value. More recently, Andersen et al. [2016] further shows that under certain mild assumptions, this lower bound can be improved to $(1 - \frac{\Phi(C)}{2})^\ell$. Nevertheless, both bounds indicate that random walks visit nodes in the same community with high probability. Note that for disconnected communities their conductance values are always 0, *i.e.*, $\Phi(C) = 0$ since there are no external edges. Under this circumstance, both Equation (10) and lower bound $(1 - \frac{\Phi(C)}{2})^\ell$ produce probability 1, which indicate that nodes from disconnected components can never be embedded in a similar fashion.

The above shows that random walks capture communities and thus any walk-based embedding method that uses either implicit walks (sequences of node ids) or explicit walks (number of walks between two nodes) outputs community-based embeddings.

Explicit Walk-based Sampling: We first discuss methods that *sample explicit walks* (sequences of node ids) from G and then use these walks to derive embeddings. A walk in G is a sequence of nodes $v_1, v_2, \cdots, v_l$ s.t. $(v_i, v_{i+1}) \in E, \forall i$. Note that the term walk-based sampling (or explicit walks) [Kolaczyk and Csárdi 2009; Ribeiro et al. 2012] is used to distinguish techniques that sample explicit walks from the graph (representing sequences of node ids) from methods that are based on (implicit) walks, but do not explicitly sample them from the graph. The basic idea behind walk-based sampling is that nodes connecting with similar sets of neighbors (identified by ids) should be embedded closer. Therefore, these approaches first sample walks explicitly from the graph and then use these explicit sequences of ids to derive low-dimensional node embeddings that maximize the likelihood of predicting them.

Naturally, nodes and their neighbors in the walks are embedded closely in the vector space. DeepWalk [Perozzi et al. 2014] is the first such method that leveraged explicit walks (sequences of node ids) to learn community-based embeddings. Deepwalk employs Skip-Gram model to derive node embeddings that maximize the probability of neighbors identified in the explicit walks, *i.e.*,

$$\arg\max_f Pr(v_{i-k}, \cdots, v_{i-1}, v_{i+1}, \cdots, v_{i+k} | f(v_i)).$$

As a result, nodes in the same community will be embedded closely by DeepWalk. Based on DeepWalk, node2vec [Grover and Leskovec 2016] introduced a way to bias the random walks and claimed that both homophily (proximity/community-based embedding) and structural equivalency (Definition 8) can be preserved in the embeddings. However, the notion of structural equivalence is

defined only in terms of an actual role assignment, not an embedding, and therefore no claim can be made about whether an embedding is structurally equivalent (or regular equivalent). Furthermore, as we showed above, random walks naturally give rise to community-based embeddings. LINE [Tang et al. 2015] explicitly uses 1- and 2-hop neighbors (node contexts) to learn community/proximity-based node embeddings. LINE minimizes the KL-divergence between the first- and second-order joint probability distribution and the empirical distribution related to edge weights separately, and forms the output embeddings through concatenation. The embeddings derived by LINE incorporates local community information. As indicated in [Qiu et al. 2018], LINE can be seen as a special case of DeepWalk with the contextual size set to one. More recently, ComE+ [Cavallari et al. 2019] learns community-based embeddings by first sampling a fixed number of *explicit paths* of a fixed length from every node, then leverages DeepWalk to obtain initial node embeddings. Afterwards, the explicit paths are again used in an iterative fashion to obtain the final embeddings.

There are many extensions of DeepWalk to handle different types of graphs. All such extensions also use explicit walks with node ids. One extension called metapath2vec [Dong et al. 2017] is proposed to embed nodes in heterogeneous networks. This work relies on meta-path based random walk to capture contexts consisting of multiple node types following predefined meta-schemas. More recently, CTDNE [Nguyen et al. 2018] introduces the notion of temporal random walk and describes a general framework based on these temporal walks to learn temporally-valid embeddings at the finest temporal granularity. There are also walk-based sampling methods for graph embedding. One such method by Taheri et al. [2018] generates multiple sequences including random walks, shortest paths between node pairs, and paths rooted at specific nodes to approximate the global graph structure and leverage sequence-to-sequence LSTM autoencoder to derive the embeddings. Other examples include Patchy-san [Niepert et al. 2016] and random-walk-based sub2vec [Adhikari et al. 2018].

Implicit Walk-based: Now we discuss implicit walk-based embeddings. These are characterized by the following property:

$$(\mathbf{A}^k)_{ij} = \text{ number of walks of length k between } i \text{ and } j \tag{11}$$

Note that unlike embeddings that use *explicit walks*, that is, sequences of node ids (*e.g.*, DeepWalk, node2vec), implicit walk-based embeddings use the count of walks in some fashion. Eq. 11 obviously captures *proximity* (*i.e.*, communities) explicitly since $\mathbf{A}^k$ is the number of walks of length k between any two nodes. Hence, the quantity itself describes the proximity between nodes. Furthermore, $\mathbf{A}^k_{ij} > 0$ iff there is a walk from node i to j of length k, otherwise $(\mathbf{A}^k)_{ij} = 0$. The above property is important, as this implies that nodes within the same community will have many such available walks compared to nodes between communities. GraRep [Cao et al. 2015] is directly based on the above. In particular, GraRep computes $\mathbf{A}^k$ for $k = 1, \ldots, K$ and derives an embedding for each $\mathbf{A}^k$ using SVD. The K embeddings are then concatenated.

Besides $\mathbf{A}^k$, we can also derive a matrix denoted as $\mathbf{A}_k$ representing the sum of all walks up to length k and use this for embedding. More formally, the graph G^k with the adjacency matrix denoted as $\mathbf{A}_k$ given by the sum of the first k powers of the adjacency matrix $\mathbf{A}$ is:

$$\begin{aligned} \mathbf{A}_k &= \sum_{i=1}^{k} \mathbf{A}^i \\ &= \mathbf{A} + \mathbf{A}^2 + \cdots + \mathbf{A}^k \end{aligned} \tag{12}$$

where $(\mathbf{A}_k)_{ij} > 0$ iff there is a walk from i to j in at least k steps and $(\mathbf{A}_k)_{ij} = 0$ if no such walk between i and j exists of length $1, \ldots, k$. Setting $k = \text{diam}(G)$ in Eq. 12 gives a complete graph.

More generally, any matrix factorization method applied to $\mathbf{A}$ *directly* results in community-based embeddings as shown in [Rossi and Ahmed 2015].[7] This is true for the adjacency matrix $\mathbf{A}$ of G or any matrix function of $\mathbf{A}$ such as the normalized Laplacian $\mathbf{L}$ or probability transition matrix $\mathbf{P}$. One such example is spectral embedding/clustering that computes the k eigenvectors of the Laplacian matrix $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ of G [Ng et al. 2002; Tang and Liu 2011]. The intuition is the same as above. Another example is HOPE [Ou et al. 2016], which proposes 4 different ways to measure the proximity, which are Katz Index, personalized PageRank, Common neighbors and Adamic-Adar. TADW [Yang et al. 2015], HSCA [Zhang et al. 2016] leverage Pointwise Mutal Information (PMI) of word-context pair to denote proximity. CMF [Zhao et al. 2015] leverages Positive Pointwise Mutal information (PPMI) by omitting unrelated pairs of nodes with negative PMI values. Modularized NMF (M-NMF) [Wang et al. 2017] is an implicit walk-based approach for deriving community/proximity-based embeddings by jointly optimizing an NMF-based model with a modularity-based community detection model. All of these methods are community-based.

These embeddings have connections to eigenvectors and in particular the principle eigenvector. The proof of the Ergodic theorem is most frequently given as an application of the Perron Frobenius theorem, which states that the probabilities of being at a node are given as the coefficients of the *principal eigenvector* of the stochastic transition matrix $\mathbf{P}$ associated to the Markov chain computed as follows:

$$\lim_{k \to \infty} \mathbf{P}^k \mathbf{e} \tag{13}$$

where $\mathbf{e}$ is the unit vector. Recall that $\mathbf{Ae}$ gives the degree vector whereas $\mathbf{A}^2\mathbf{e}$ is the number of walks of length 2, and so on. In general, the operation $\mathbf{Ae}$ is essentially equivalent to a single BFS iteration (over all nodes), see [Kepner and Gilbert 2011] for more details.[8] The above has been used for decades in a variety of seminal community detection and community-based embedding methods [Andersen et al. 2006; Gibson et al. 1998; Schaeffer 2007].

4.2 Feature Propagation/Diffusion

While most community-based embeddings arise from explicit or implicit walks (Section 4.1), there are also many methods that use feature diffusions (*i.e.*, feature propagation) to learn community-based embeddings. These methods are fundamentally tied to communities as they are still related to walk-based methods (which we formally showed in Section 4.1 that such walk-based methods are fundamentally community-based). The only difference is that features are diffused through the neighborhoods. Thus, as $k \to \infty$, for any $(\mathbf{AX})^k$, then the features are smoothed over the graph. Using Eq. 10, we can see that nodes within the same community will have similar embeddings since the diffusion and resulting features primarily stay within the same community by definition. Thus, the nodes within the same community become more and more similar as features are diffused from further away (but primarily from nodes within the same community), making all such nodes in the same community appear increasingly similar to one another. This is precisely why such graph diffusion lies at the heart of many community detection methods such as heat kernel [Kloster and Gleich 2014], PageRank communities [Andersen et al. 2006], among many others [Kondor and Lafferty 2002; Raghavan et al. 2007]. Furthermore, these methods typically rely on selecting a good seed set of nodes to begin such diffusions. In theory, a good seed set should contain one or

[7] This is the reason that feature-based role methods were proposed in [Rossi and Ahmed 2015], which avoid using $\mathbf{A}$ directly, and instead derive a structural feature matrix $\mathbf{X}$ that captures the structural properties (*e.g.*, degree, triangles, betweenness, k-stars) of G and then uses this matrix $\mathbf{X}$ to derive roles.

[8] BFS and DFS are general graph traversal/search techniques used in the implementation of graph algorithms. These techniques are simply different ways to visit nodes in the graph, and are often used in the implementation of *both* community- and structural role-based embeddings.

more vertices from each "community", otherwise, some communities will be missed for precisely the same reason (it is unlikely that a walk starting from one community will end up in another community) as shown above in Eq. 10.

In general, propagation/diffusion-based methods make the assumption that some initial attributes are given as input and stored in $\mathbf{X}$. For instance, we may associate with each node in a social network features that were taken from the corresponding user's profile. Node embeddings are then generated via a k-step diffusion process. At each step, a node's features are diffused to its immediate neighbors and, after k rounds, each node obtains an embedding which is essentially an aggregation of the information in its k-order neighborhood. While the diffusion process is dependent on walks between node pairs, methods falling into this category do not explicitly leverage walks to approximate the graph structure. Instead, they propagate information over the graph structure up to k-orders, and characterize individual nodes by collecting the diffused feature values in the neighborhood [Rossi et al. 2017; Xiang et al. 2018]. Thus, embeddings based on feature diffusion are community-based as the diffusion process is fundamentally tied to proximity in the graph as opposed to structural properties of nodes. See Table 3 for a summary of a few representative network embedding methods based on feature propagation/diffusion.

The general form of this process is denoted as

$$\tilde{\mathbf{X}} = \Psi(\mathbf{A}, \mathbf{X}) \tag{14}$$

where Ψ denotes the feature expansion or diffusion function and $\tilde{\mathbf{X}}$ denotes the expanded features. In the simplest case, a feature matrix propagating over the graph structure can be written as the standard form of Laplacian smoothing:

$$\tilde{\mathbf{X}}^{(t)} = \mathbf{D}^{-1}\mathbf{A}\tilde{\mathbf{X}}^{(t-1)} = (\mathbf{D}^{-1}\mathbf{A})^t\tilde{\mathbf{X}}^{(0)} = (\mathbf{D}^{-1}\mathbf{A})^t\mathbf{X} \tag{15}$$

where $\mathbf{D}$ is the diagonal degree matrix and t represents iteration t of the diffusion process. $\tilde{\mathbf{X}}^{(0)}$ is generally set to be the initial feature matrix $\mathbf{X} \in \mathbb{R}^{N\times F}$, *i.e.*, $\tilde{\mathbf{X}}^{(0)} = \mathbf{X}$.

More complex feature diffusion processes can be denoted through the Laplacian diffusion process:

$$\tilde{\mathbf{X}}^{(t)} = (1-\theta)\mathbf{L}\tilde{\mathbf{X}}^{(t-1)} + \theta\mathbf{X} \tag{16}$$

where θ controls the weighting between features of a node itself and its neighbors. $\mathbf{L}$ is the normalized Laplacian:

$$\mathbf{L} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}} \tag{17}$$

The Laplacian smoothing process generates new features as the weighted average given a specific node itself and its neighbors.

Many of the recent propagation or diffusion-based methods [Hamilton et al. 2017; Kipf and Welling 2016, 2017; Veličković et al. 2018] also incorporate trainable parameters into the diffusion process to learn better community/proximity-based embeddings. For instance, the step-wise diffusion for GCN can be defined as follows:

$$\tilde{\mathbf{X}}^{(t)} = \sigma\left(\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}\tilde{\mathbf{X}}^{(t-1)}\mathbf{W}^{(t)}\right) \tag{18}$$

where $\mathbf{W}^{(t)}$ is the weight matrix for step t and σ is a non-linearity. Depending on specific propagation configuration, there is a number of recent works that fall under this category. For example, GAE (and its variant using variational training manner, VGAE) [Kipf and Welling 2016] leverages GCN to encode node features, and then uses a simple decoder to reconstruct the graph adjacency matrix so that the loss can be minimized. DySAT [You et al. 2019] casts the Boolean Satisfiability (SAT) problem as a problem of deriving latent bipartite graph representations and provides a solution in the GCN feature aggregation manner. HGCN [Chami et al. 2019] extends GCN into the hyperbolic space so that node features are learned with less distortion. Some other variants are devoted to

capturing both spatial and temporal dependency between nodes in the graph, such as [Wu et al. 2019b]. In order to gain interpretability of these models, GNNExplainer [Ying et al. 2019] was proposed as a model-agnostic approach to explain the prediction/inference on machine learning tasks, while GroupINN [Yan et al. 2019] introduced a grouping or summarization layer that explains the classification by exposing the most relevant edges. PRUNE [Lai et al. 2017] is another recent proximity/community-based embedding method that uses a Siamese neural network structure to preserve proximity among the nodes.

Now we show that repeated application of a smoothing operator results in the features converging to the same quantity. In other words, the features of nodes within the same community become indistinguishable from one another as the number of iterations (feature propagations) become large. In particular, we prove that by iteratively applying the smoothing operator $\mathbf{D}^{-1}\mathbf{A}$, the feature vectors associated with nodes in a connected graph G will converge in the end. The same applies when $\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ is used as a smoothing operator over every node and its neighbors in the graph.

Theorem 4.1. *Assuming G is connected and non-bipartite, then for any feature/embedding matrix $\mathbf{X} \in \mathbb{R}^{n\times F}$:*

$$\lim_{t\to\infty}(\mathbf{D}^{-1}\mathbf{A})^t\mathbf{X} = \mathbf{1}\mathbf{y}^T \tag{19}$$

and

$$\lim_{t\to\infty}(\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}})^t\mathbf{X} = \mathbf{D}^{-\frac{1}{2}}\mathbf{1}\mathbf{y}^T \tag{20}$$

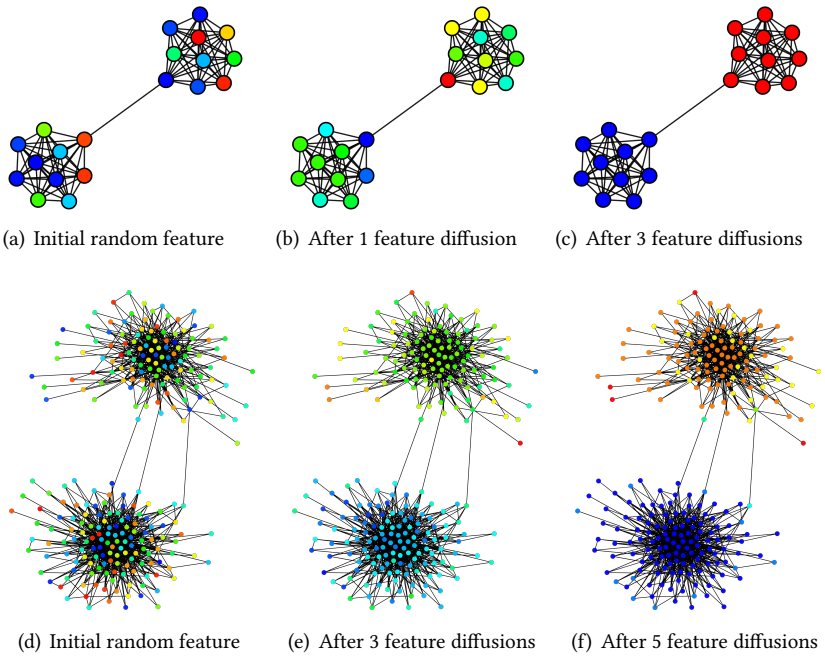
where $\mathbf{y} \in \mathbb{R}^F$. Hence, Eq. 19 converges to identical feature vectors for all nodes whereas the features of nodes smoothed using Eq. 20 (normalized Laplacian) converge to be proportional to the square root of the node degree.

Note that $\mathbf{D}^{-1}\mathbf{A}$ in Eq. 19 is for the random walk Laplacian matrix whereas $\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ in Eq. 20 is for the normalized Laplacian.

Proof. Let $\mathbf{L}_{rw} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{A}$ and $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ (Eq. (17)). $\mathbf{L}_{rw}$ and $\mathbf{L}$ have the same n eigenvalues by multiplicity with different eigenvectors with eigenvalues in $[0, 2)$ [Chung 1997]. The eigenspaces corresponding to eigenvalue 0 are thus spanned by $\mathbf{1}$ and $\mathbf{D}^{-\frac{1}{2}}\mathbf{1}$, respectively. Thus the eigenvalues of $\mathbf{D}^{-1}\mathbf{A}$ $(\mathbf{I} - \mathbf{L}_{rw})$ and $\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ $(\mathbf{I} - \mathbf{L})$ would fall into $(-1, 1]$. Since the absolute value of the eigenvalues are less than or equal to 1, the repeating multiplication will converge to the largest eigenvector corresponding to eigenvalue 1, which is $\mathbf{1}$ and $\mathbf{D}^{-\frac{1}{2}}\mathbf{1}$, respectively. ■

The above obviously holds for $k = |C|$ communities $C = \{C_1, \ldots, C_k\}$ such that $|E(C_i, C_j)| = 0, \forall i, j$, *i.e.*, there are no edges between any pair of the communities. This is an extreme case. However, when the number of feature propagations is small, it is easy to see that the resulting embedding vectors of nodes within the same community become more and more similar due to the smoothing of the nodes within the same community. Intuitively, when t is small, then after t feature propagations, the resulting diffused feature vectors of nodes within the same community are more similar to each other than to the diffused feature vectors of nodes in another community. This occurs since the nodes inside a community are not as impacted by the features in another community due to the sparse edges between communities, *i.e.*, $|E(C_i, C_j)| \leq |E(C_i)|, |E(C_j)|$, hence the number of edges between nodes in the same community is significantly larger than the number of edges between communities (by Definition 2). Thus, nodes in the same community become more and more similar to each other.

In Figure 2, we provide an illustration of Theorem 4.1 for further intuition. This example clearly shows why feature diffusion gives rise to community-based embeddings. In particular, after only 3



(a) Initial random feature (b) After 1 feature diffusion (c) After 3 feature diffusions

(d) Initial random feature (e) After 3 feature diffusions (f) After 5 feature diffusions

Fig. 2. Feature diffusion example via $\mathbf{D}^{-1}\mathbf{A}$ in a barbell graph (top) and a graph following the Block Chung-Lu model (bottom). For demonstration/visualization purposes, we use a single feature, the value of which is denoted by the node color. Feature values were drawn from the uniform distribution on the open interval $(0, 1)$. For the Block Chung-Lu model, the graph was generated with exp(1.7). See text for discussion.

iterations of feature diffusion, the diffused features of nodes in either community become indistinguishable from one another. More precisely, the diffused feature values of nodes within the same community are identical to one another after only a few iterations. Furthermore, even after the first iteration of feature diffusion (Figure 2(b)), the features of nodes in the same community appear more similar and after only 3 iterations, the features of nodes in each community are indistinguishable from one another (Figure 2(e)).

Graph diffusion has also been used in community detection methods for decades [Barbieri et al. 2013; Kloster and Gleich 2014; Lin et al. 2015]. The diffusion process adopted by GCN is based on the Laplacian which has traditionally been used for community detection [Schaeffer 2007]. Furthermore, recent work has also shown that embeddings from GCN are useful for community detection [Bruna and Li 2017; Chen et al. 2018; Shchur and Günnemann 2018]. The Laplacian-based diffusion used by GCN essentially uses a weighted sum to aggregate features from a node's neighbors.

While the diffusion process is dependent on walks between node pairs, methods falling into this category do not explicitly leverage walks to approximate the graph structure. Instead, they propagate features over the graph structure up to t-orders, and characterize individual nodes by collecting the diffused feature values in the neighborhood [Xiang et al. 2018]. Thus, embeddings based on feature diffusions are community-based as the diffusion process is fundamentally tied to proximity in the graph as opposed to structural properties of nodes.

While all work described above essentially uses sum as a diffusion operator, DeepGL proposed the idea of using general aggregation functions. Intuitively, DeepGL [Rossi et al. 2017] replaces the

sum aggregator (which is naturally represented by a matrix-vector or matrix-matrix multiplication) with a general aggregation function ϕ. More generally, unlike previous work that used only sum, DeepGL uses multiple aggregation functions. Examples of ϕ include min, max, product, mean, median, mode, L_1, L_2, RBF or more generally, any function that can be defined between a node i and its neighborhood (or k-hop neighborhood). More recently, this idea has been adopted in other works such as GCN-GraphSage [Hamilton et al. 2017] and MultiLENS [Jin et al. 2019c]. However, replacing sum with a different aggregation function (or even multiple aggregation functions) does not change the fact that these methods are community-based in general for large k.

There are a few cases where feature diffusion can be used to derive role-based embeddings. Note that when $\mathbf{X}$ is motif/graphlet features and the number of feature propagations k is 1, then role-based embeddings can be derived. Intuitively, the motif/graphlet features are not smoothed out for $k = 1$. However, as k increases, the impact of the motif/graphlet features in their ability to capture the structural features are lost since they become increasingly similar to their neighbors. Clearly, role-based embeddings can also be derived if $\mathbf{X}$ representing motif/graphlet features is used without any diffusion (degenerate case).

5 ROLE-BASED EMBEDDING

This section discusses the main mechanisms behind role-based embeddings. One notable observation is that the general mechanisms behind role-based embeddings all exploit an initial (small) set of structural features in some fashion. As such, these mechanisms give rise to feature-based role methods [Rossi and Ahmed 2015] where nodes with similar network neighborhoods will have similar embeddings when such mechanisms are used, despite that the nodes may be in different parts of the graph or even different graphs altogether. We summarize the role-based embedding mechanisms in Table 2 along with a few representative methods that use each mechanism.

5.1 Graphlets

We first give the definition of graphlets (network motifs/induced subgraphs) and orbits, then show how they can be used for learning role-based embeddings.

Definition 16 (Graphlet). *A k-vertex graphlet $H = (V_k, E_k)$ is an induced subgraph consisting of a subset $V_k \subset V$ of k vertices from $G = (V, E)$ together with all edges whose endpoints are both in this subset $E_k = \{\forall e \in E \mid e = (u, v) \wedge u, v \in V_k\}$.*

The edges of a graphlet can be partitioned into a set of automorphism groups called orbits based on the *position* (or "role") of an edge in a graphlet [Ahmed et al. 2015; Pržulj 2007]. Formally,

Definition 17 (Orbit). *An automorphism of a k-node graphlet $H_t = (V_k, E_k)$ is defined as a permutation of the nodes in H_t that preserves edges and non-edges. The automorphisms of H_t form an automorphism group denoted as $Aut(H_t)$. A set of nodes V_k of graphlet H_t define an* orbit *iff (i) for any node $u \in V_k$ and any automorphism π of H_t, $u \in V_k \iff \pi(u) \in V_k$; and (ii) if $v, u \in V_k$ then there exists an automorphism π of H_t and a $\gamma > 0$ such that $\pi^\gamma(u) = v$.*

Graphlets naturally capture the key structural properties of edges and nodes in the graph as shown in Figure 3. In particular, Figure 3 shows the full spectrum of connected graphlets with $\{2, 3, 4\}$-nodes; each set of k-node graphlets are ordered from least to most dense. Notice that graphlets are the fundamental building blocks of graphs since any graph (or ℓ-hop neighborhood subgraph surrounding a node/edge) can be decomposed into its smaller subgraph patterns (graphlets). In other words, any (sub)graph can be represented using only k-node graphlets. Therefore, by definition, the graphlets and their counts must capture the structural properties that are important to a node or edge. As such, graphlets (and their edge orbits) capture precisely the notion of role. This

can be trivially verified from Figure 3 as many of the individual graphlets can even capture the traditional examples of roles used in the literature. Recall roles represent node (or edge [Ahmed et al. 2017b]) connectivity patterns such as hub/star-center nodes, star-edge nodes, near-cliques or bridge nodes connecting different regions of the graph. Graphlets capture the full spectrum of possible connectivity/subgraph patterns arising in graphs as shown in Figure 3, which lies at the heart of the notion of roles (Section 3.2). Intuitively, two nodes belong to the same role if they are structurally similar with respect to their general connectivity/subgraph patterns [Rossi and Ahmed 2015]. Therefore, it is only natural to consider graphlet features when learning role-based embeddings. There is a broad class of embedding methods that represents structural information using graphlets to learn role-based embeddings. Graphlets and the statistics (*e.g.*, frequencies) carry significant information about the structural properties of nodes and edges and have been used for many applications [Ahmed et al. 2017a; Faust 2010; Holland and Leinhardt 1976; Milo et al. 2002]. The set of decomposed graphlets $\{H_1, H_2, \cdots H_d\}$ can be used to characterize both the whole graph and individual nodes/edges by counting the number of times in the embedded d-dimensional vector. Intuitively, nodes associated with similar graphlet/motif types (*e.g.*, triangle, star) and counts are structurally similar and thus are embedded closer in some low-dimensional space.

The graphlet features are computed for each node (or edge) in the graph and naturally generalize across graphs (for transfer learning tasks) since they represent "structural graph functions" that are easily computed on any arbitrary graph. Fast algorithms for counting such graphlets/network motifs in very large graphs have become common place, *e.g.*, PGD [Ahmed et al. 2015] takes a few seconds to count graphlets in very large networks with hundreds of millions of edges. Furthermore, since most embedding methods are not exact and we typically only care about the relative/approximate magnitude of the count (*e.g.*, whether the count is on the order of 10^1, 10^2, 10^3 and so on), and not the actual exact count, we can also leverage provably accurate graphlet estimators to obtain graphlet counts even faster.

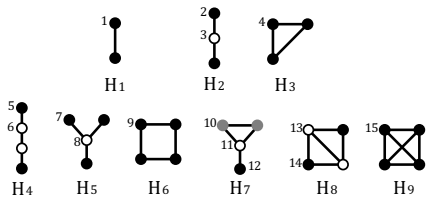
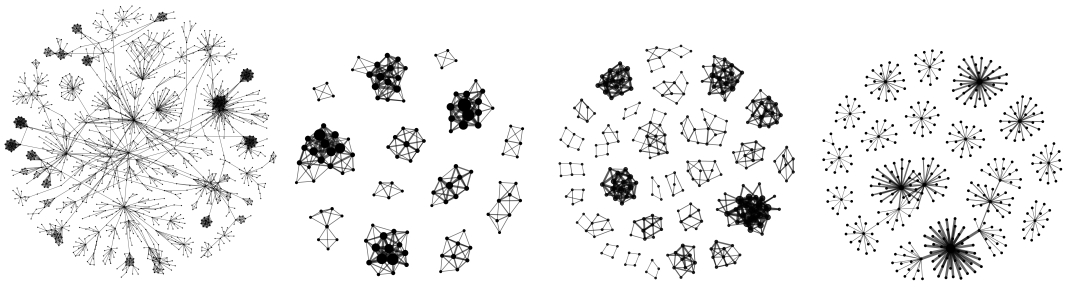


Fig. 3. All 9 graphlets and 15 node orbits with $\{2, 3, 4\}$-nodes. Each unique node position of a graphlet is labeled, *e.g.*, nodes in the 4-star graphlet (H_5) have two unique positions, namely, the star-center (hub) position or the star-edge (peripheral) position.

Representative approaches falling into this category are as follows. Role2vec [Ahmed et al. 2018] proposed an end-to-end inductive framework to learn role embeddings that capture the structural similarity among nodes and generalizes across networks. To achieve this, role2vec introduced the general notion of *feature-based random walks*, to replace the traditional random walks (*i.e.*, random sequence of node ids) by walks that represent the structural similarity among nodes, where each walk is a sequence of features or functions of multiple features. Thus, the feature-based walks find nodes with similar structure identified by structural properties and higher-order features (*e.g.*, graphlets), and enable space-efficient role embeddings. DeepGL [Rossi et al. 2017] learns inductive graph functions where each represent a composition of relational operators/aggregator functions applied to a graphlet/motif feature. HONE [Rossi et al. 2018b] proposed the notion of higher-order network embeddings and described a framework based on weighted k-step motif graphs to learn the low-dimensional role-based embeddings. More recently, higher-order motif-based GCN's



(a) Initial graph $\mathbf{A}$ (b) Weighted 4-clique graph (c) Wt. 4-cycle graph (d) Wt. 4-star graph

Fig. 4. Graphlet/motif graphs differ in structure *and* weight. Size (weight) of nodes and edges in the motif graphs correspond to the frequency of the motif. In this example (web-google), the initial (edge motif) graph is fractured into many disconnected components when deriving the motif graphs. This is due to the constraint that each edge in an arbitrary motif graph contain at least a single motif. Edges are removed if they do not participate in at least one 4-clique (b), 4-cycle (c), or 4-star (d).

called MCN were proposed by Lee et al. [2018b]. MCN leverages the weighted motif-based matrix functions introduced in HONE [Rossi et al. 2018b] to learn role-based embeddings. While HONE and MCN (a higher-order generalization of GCN) also use different forms of feature diffusion, they are nevertheless role-based since they do not leverage $\mathbf{A}$ directly, but instead use $\mathbf{A}$ to derive a set of *weighted motif graphs* (*i.e.*, weighted motif adjacency matrices $\mathbf{W}_1, \mathbf{W}_2, \ldots, \mathbf{W}_k$) that are then used to derive node embeddings. In particular, given a motif H, the weighted motif adjacency matrix of H denoted $\mathbf{W}_H$ is defined as:

$$(\mathbf{W}_H)_{ij} = \text{number of instances of motif } H \text{ that contain nodes } i \text{ and } j \tag{21}$$

The weighted motif adjacency matrices differ fundamentally in structure (and weight) when compared to the original graph as shown in Figure 4. In particular, the motif graphs typically consist of many connected components, *i.e.*, the graph shatters into many connected components due to the requirement that each edge have at least $\delta > 0$ motifs. This requirement acts as a filter, removing many of the unimportant edges, and highlighting only the edges (and structures) in the graph that contain at least one such occurrence of H (or more generally δ occurrences of H). In Figure 4, the motif graphs immediately reveal nodes with similar structural properties. For instance, the weighted 4-star graph (Figure 4(d)) fractures the graph into many connected components where each connected component consists of nodes that are either (i) *star-center* (hub) nodes of some larger star structure or (ii) *star-edge* (peripheral) nodes.[9] Furthermore, the graphlet/motif graphs immediately reveal larger subgraph patterns, *e.g.*, the 4-star graph shown in Figure 4(d) shows large stars made up of many 4-stars. Both of these properties are important when considering feature diffusion (via a graph smoothing operator such as the normalized Laplacian $(\mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}})\mathbf{X}$ or $(\mathbf{D}^{-1}\mathbf{A})\mathbf{X}$) and its meaning and impact when used with these weighted motif graphs. As such, the diffusion process is performed over each connected component in a motif graph, and has less of an impact since all nodes in the motif graph by definition have similar structural properties.

[9]Recall from Section 3 that star-centers (hubs) and star-edges are two of the classic examples used for roles.

5.2 *Feature-based* Walks

While we theoretically showed in Section 4 that walk-based embedding methods derive community / proximity-based embeddings, Ahmed et al. [2017c] proposed the notion of a feature-based walk (attributed random walk) that allows existing walk-based methods to be generalized for learning role-based embeddings. The general idea is to generate walks that represent sequences of feature values as opposed to sequences of node ids as done in DeepWalk, node2vec, among many others. Intuitively, the feature values in the walks naturally generalize across graphs since they can represent general graph functions (like degree, number of triangles, 4-node cycles). By applying a mapping function Φ to map sequences of node ids (walks) to their associated feature-values, the skip-gram model (or any other model that uses the walks) would preserve the similarity in terms of attributes in the embeddings. Formally, the walk consisting of node feature-values/types/labels/attributes is defined as follows:

Definition 18 (Feature-based Walk). *Let $\mathbf{x}_i$ be a K-dimensional feature vector for vertex v_i. A feature-based (or attributed) walk of length L is a sequence of adjacent feature-values,*

$$\Phi(\mathbf{x}_1), \Phi(\mathbf{x}_2), \cdots, \Phi(\mathbf{x}_L) \tag{22}$$

induced by a sequence of indices $(v_1, v_2, \cdots, v_L)$ generated by a random walk of length L starting at v_1, and a function Φ that maps a feature vector $\mathbf{x}$ to a type $\Phi(\mathbf{x})$.

Definition 18 can be used in a variety of ways and gives rise to many interesting role-based methods [Ahmed et al. 2018]. For instance, instead of using a mapping function (which can be thought of as replacing Φ with the identity function), one can simply use one or more features to derive feature-based walks for each different feature.

Recently, an approach called role2vec [Ahmed et al. 2018] was proposed that learns role-based node embeddings by first mapping each node to a type (or role) via a function and then uses the notion of attributed random walks to derive role-based embeddings for the nodes that capture structural similarity. Since the "feature-based" random walks by definition capture the structural properties (the features can be thought of as describing the topological/structural characteristics of a node), node embeddings (representations, encodings) learned from these attributed/feature-based walks are able to capture roles as opposed to communities. Attributed random walks [Ahmed et al. 2017c, 2018] have since become increasingly popular with applications in entity resolution [Jin et al. 2019a], churn prediction [Liu et al. 2018b, 2019], among many others [Al Etaiwi and Awajan 2018; Chen et al. 2020a; Rozemberczki et al. 2019; Zhang et al. 2019]. In particular, node2bits [Jin et al. 2019a] builds on the idea of feature-based walks and also the notion of temporal walks that obey time from CTDNE [Nguyen et al. 2018] in order to obtain temporal, feature-based context around the nodes (based on both structural features and node attributes). It then aggregates the contexts into node-level histograms and obtains compact, binary node representations that preserve context similarity via locality sensitive hashing (LSH) and uses the final embeddings for the application of visitor stitching/entity resolution. More recently, [Liu et al. 2018b, 2019] proposed SimSum that leverages attributed random walks to analyze churn in mobile games at both the micro and macro levels. Another extension of role2vec called RiWalk [Xuewei et al. 2019] introduced a new role mapping function based on the shortest path and Weisfeiler-Lehman kernels. Many other works have also used the notion of attributed random walks for a variety of applications and problem settings [Al Etaiwi and Awajan 2018; Huang et al. 2019; Rozemberczki et al. 2019; Zhang et al. 2019].

5.3 *Feature-based* Matrix Factorization

There are also role-based structural embeddings that use a form of matrix factorization over a matrix of structural features [Rossi and Ahmed 2015]. This is in contrast to community-based

embeddings that use matrix factorization over the graph's adjacency matrix. This class of role-based embeddings largely depends on the structural feature matrix used in the factorization. For instance, suppose the features in the matrix are all correlated with communities (as opposed to roles; and thus they are not *structural features*), then the resulting embeddings would in fact be community-based and not role-based. Thus, the most critical step in these methods is to ensure the initial set of structural features are appropriate and are most suitable for capturing roles.

This class of feature-based role embeddings along with a general framework for computing them was introduced in [Rossi and Ahmed 2015]. One such work by Henderson et al. [2012] starts with degree/egonet-based features, aggregates them recursively, and then uses Non-negative Matrix Factorization (NMF) over the feature matrix to derive roles. More recently, Rossi et al. [2013] proposed modeling feature-based roles in dynamic networks whereas Gilpin et al. [2013] used a sparsity regularized NMF (as well as other convex constraints) to learn better roles. While all previous methods directly learn node embeddings, Ahmed et al. [2017b] learns role-based *edge* embeddings. This approach starts with higher-order graphlet features that explicitly captures the notion of roles (see Section 5.1) and iteratively computes additional higher-order features via relational aggregates over the neighborhood, and then factorizes this matrix of structural features to obtain role-based embeddings of the edges. More recently, xNetMF [Heimann et al. 2018] computes degree-based features from a node's neighborhood at different hops, and uses implicit matrix factorization over this feature matrix to obtain generalizable embeddings that are suitable for network alignment. EMBER [Jin et al. 2019b] generalizes xNetMF to weighted and directed graphs by extending the degree-based distributions to weighted distributions defined over directed neighborhood contexts. SEGK [Nikolentzos and Vazirgiannis 2019] also computes the similarity between nodes by leveraging different hops of neighborhoods, but utilizes graph kernels for the comparison, and then factorizes the resultant kernel matrix to obtain the structural node embeddings. While most of the approaches discussed in this section so far first compute a tall-and-skinny "structural" feature matrix, struc2vec [Ribeiro et al. 2017] computes multiple (large and dense) node-by-node feature matrices between all pairs of nodes using dynamic time warping (DTW) distance based on sequences of node degrees (*i.e.*, degrees of the neighbors of a node). Afterwards, explicit walks over the resultant dense multi-layer graph are sampled and used to derive embeddings in a similar fashion to DeepWalk.

Moreover, we note that any of the previous structural embeddings from Sections 5.1-5.2 can be used as input into matrix factorization to learn more compact and space-efficient role-based embeddings [Rossi and Ahmed 2015].

6 APPLICATIONS

In this section, we describe applications for community (proximity) and role-based embeddings. For each application, we discuss conditions including data characteristics, noise, and variants of the applications where community and role-based embeddings are most suitable. Notably, community or role-based embeddings are shown to be useful for the same applications such as classification, link prediction, and anomaly detection. The fundamental difference of whether community or role-based embeddings are preferred depends entirely on the underlying data characteristics (*e.g.*, homophily vs. heterophily, noisy/missing data vs. clean/accurate data) and problem setting/assumptions.

6.1 Node classification

Embeddings have been used to improve node classification performance. We discuss a few different node classification tasks below and mention the key differences and data characteristics that make community-based or role-based embeddings more appropriate.

6.1.1 Community-based embeddings. Semi-supervised classification in graphs typically performs best with community-based embeddings since these methods iteratively predict labels of neighbors and propagate them to neighboring nodes [Sen et al. 2008]. In other words, the labels of neighboring nodes are repeatedly diffused to the neighbors until convergence. The overall process is similar to feature diffusion-based methods from Section 4.2. Typically, these methods assume a small fraction of nodes with known labels are given (for training), and since neighboring nodes are assumed to be labeled the same, these methods are most useful for graphs with significant homophily/large relational autocorrelation, *i.e.*, graphs where the node labels are highly correlated with their immediate neighbors [Koutra and Faloutsos 2017; La Fond and Neville 2010; Neville et al. 2004]. Examples of such graphs with strong homophily include Cora, CiteSeer, among many others [McDowell et al. 2009]. As such, if such *strong homophily* exists, then community-based embedding methods are most appropriate. This is the reason why many community-based embedding methods such as GCN [Kipf and Welling 2017] are evaluated for semi-supervised classification using graphs with *strong homophily* such as Cora, CiteSeer, PubMed, and others. Nevertheless, community-based embeddings are also preferred for general node classification with homophily [Cavallari et al. 2017; Grover and Leskovec 2016; Perozzi et al. 2014; Tang et al. 2015], going beyond the semi-supervised classification setting.

6.1.2 Role-based embeddings. Role-based embeddings are based on **structural** similarity (Definition 14) and thus appropriate for classifying nodes with similar functionality (roles) in terms of their structural properties, *e.g.*, triangles, betweenness, stars, etc. For collective/semi-supervised classification, there are some instances where role-based embeddings can perform better than community-based. For instance, role-based embeddings are often useful for graphs with weak/low homophily. Such graphs may have weak homophily due to noise, incompleteness, or other data collection/sampling issues, or graphs with heterophily where node labels (and attributes) are not correlated with the labels of their neighbors [Gatterbauer et al. 2015; Peel 2017; Rogers and Bhowmik 1970; Rossi et al. 2018]. For instance, molecular, chemical, and protein networks often have between 2 and 20 class labels, which are highly correlated with the structural properties (*e.g.*, graphlets/network motifs) and behavior surrounding a given node or edge in the graph [Gardiner et al. 2000; Vishwanathan et al. 2010]. Networks of email communication in the workplace is another example where the professional roles of nodes (e.g., C-suite employees vs. managers) correlate with their structural properties in the network [Jin et al. 2019b]. In these cases, the nodes whom share class labels are often not directly connected, or even in the same community, but share similar structural properties and behavior (or role/position) in the network [Rossi and Ahmed 2015].

While community-based embeddings are primarily useful for semi-supervised classification, role-based embeddings are also well-suited for *across-network (relational) classification* where the goal is to learn a classification model on one graph and then use it to predict the labels of nodes in an entirely different graph that may not share any of the same nodes. The two graphs could have completely different nodes (*i.e.*, node ids) or may have some nodes in common between the two graphs, *e.g.*, in temporal networks where there is a sequence of graphs over time. This application is sometimes called relational classification as opposed to semi-supervised classification, see [Rossi et al. 2012] for more details.

6.2 Link prediction

Link prediction is another important application where embeddings can be used to improve performance over simpler approaches such as common neighbors, Jaccard similarity and the ilk [Ahmed et al. 2018; Grover and Leskovec 2016; Kipf and Welling 2016]. Given a graph $G = (V, E)$, the link prediction task is to predict a set of (top-k) missing (unobserved) or future links E' such that

$E' \cap E = \emptyset$. Given node embeddings (either community-based or role-based), links can be predicted by computing edge feature vectors $g(\mathbf{x}_i, \mathbf{x}_j), \forall i, j$ pairs (*i.e.*, in the training set), and then learning a model based on these, which is then used to predict the likelihood that a link exists between any arbitrary pair of nodes. Alternatively, we can directly compute a score without learning a model.

6.2.1 Community-based Embeddings. In many cases, the missing or future links are assumed to arise between nodes that share many of the same neighbors. Hence, given nodes i and j such that $(i, j) \notin E$, community-based embeddings are useful when $|N_i \cap N_j| > 0$, that is, i and j share at least one neighbor (1-hop away). The above condition implies that i and j are near one another in the graph due to the sharing of at least one neighbor among them. We call such predicted links short-range, since they are between nodes that are close to one another in the graph. It is because of this property that community-based embeddings will work best for predicting such links (especially if the pair of nodes are both in the same community, and therefore will be embedded in a similar fashion) [Grover and Leskovec 2016; Kipf and Welling 2017].

6.2.2 Role-based Embeddings. There are also many settings and applications where role-based embeddings perform best for link prediction. In some cases, the graph data may be noisy or incomplete due to sampling or data collection issues [Ahmed et al. 2014], and therefore the actual links may not be close in terms of graph distance. More formally, $|N_i \cap N_j| = 0$. In fact, the actual links could be between nodes that are far from one another in the graph (long-range) or even in different connected components [Ahmed et al. 2018; Gilpin et al. 2013; Jin et al. 2019a]. We call such links long-range as opposed to short-range. Furthermore, links may also be predicted between nodes far away in the graph to improve relational autocorrelation or similarity, see [Gallagher et al. 2008; Lassez et al. 2008; Neville and Jensen 2005; Rossi et al. 2012].

6.3 Graph alignment and classification

Node embeddings have also been used for graph-level tasks, such as graph alignment and classification. Network alignment seeks to find the corresponding nodes across two or more networks. It can be thought of as a link prediction problem, where the links are predicted between two nodes i and j such that i is in one graph G and j is a node in another graph G'. Graph classification aims to categorize graphs into classes based on their structure. We discuss the properties that make community-based or role-based embeddings appropriate for these tasks.

6.3.1 Community-based Embeddings. Methods that learn proximity-based node embeddings via diffusion or propagation, such as GCN [Kipf and Welling 2017] and GraphSAGE [Hamilton et al. 2017], can be used to obtain *supervised* network representations by aggregating (e.g., concatenation) the node embeddings. Supervision makes these representations suitable for graph-level tasks such as network classification. Some graph neural network-based methods tailored to graph classification supervise layer-wise node pooling to learn expressive, hierarchical network representations [Ying et al. 2018], or supervise node grouping (or graph summarization) in order to learn more robust-to-noise network representations, provide interpretability, and achieve better scalability [Yan et al. 2019]. Transductive community-based embeddings (that do not leverage structural or other node features) are specific to a network, and fail to tackle cross-network tasks such as unsupervised network alignment [Heimann and Koutra 2017] or unsupervised network classification (e.g., by simply aggregating unsupervised community-based node embeddings into a network-level vector representation). However, recent work inspired by machine translation has shown that transductive proximity-based embeddings can be used to improve the performance in network alignment by encouraging the alignment of local neighborhoods across different graphs (rather than greedily

matching nodes with the same structural properties), but *only after* appropriately aligning/rotating the embedding spaces of the networks [Chen et al. 2020b].

6.3.2 Role-based Embeddings. Role-based embeddings generalize across networks [Rossi and Ahmed 2015], so they are useful in network alignment and identity resolution (or user stitching, which can be seen as a node correspondence problem within a single network or across multiple networks) [Gilpin et al. 2013; Heimann et al. 2018; Jin et al. 2019a; Zhang et al. 2013]. In such applications, transductive community-based embeddings are unable to be used off-the-shelf since the corresponding embeddings belong to different, *unaligned* latent spaces [Heimann and Koutra 2017]. However, since role-based node embeddings are based on structural properties (*e.g.*, degree, graphlet counts, betweenness) that generalize over any graph, they can naturally be used in such settings, referred to as graph-based transfer learning [Rossi et al. 2017]. Role-based embeddings also generalize across different parts of a single network, making them suitable for matching nodes corresponding to the same entity (i.e., identity resolution or user stitching) [Jin et al. 2019a]. Beyond alignment and identity resolution, aggregating role-based node embeddings can provide a powerful descriptor of an entire network, which is useful for network classification. For example, a graph descriptor can be obtained by using role-based embeddings (*e.g.*, inductive extension of xNetMF [Heimann et al. 2019, 2018], SEGK [Nikolentzos and Vazirgiannis 2019]) and creating a graph-level feature vector based on the distribution or spatial overlap of the embeddings [Heimann et al. 2019; Nikolentzos et al. 2017] or a graph kernel [Nikolentzos and Vazirgiannis 2019]. On the other hand, embeddings based on proximity are not suitable for this setting.

6.4 Anomaly detection

Community or role-based embeddings also have applications in graph-based anomaly detection where the goal is to identify (node/edge/subgraph) anomalies that do not conform to the expected behavior in the graph [Abello et al. 2010; Akoglu et al. 2015; Fond et al. 2018]. Such nodes/edges/subgraphs with non-conforming behavior are known as anomalies, outliers, or exceptions [Chandola et al. 2009]. This problem also has connections to change detection in temporal networks. There are specific problem settings in anomaly detection where community-based or role-based embeddings are more appropriate. We discuss these settings below.

6.4.1 Community-based Embeddings. One anomaly detection application of community-based embeddings is in the detection of anomalous global changes between different static snapshot graphs derived from the temporal network (sequence of edge timestamps) [Idé and Kashima 2004; Jin et al. 2019c]. One particular instance of this problem uses communities (groups of nodes that are tightly/densely connected) and considers an anomaly (or change-point) to occur when the nodes and their community-based embeddings differ significantly from the previous time at $t-1$ [Chen et al. 2012; Idé and Kashima 2004]. The underlying assumption is that the communities and community-based embeddings will remain largely stationary over time, *i.e.*, there is minor differences between time $t-1$ and time t [Akoglu et al. 2015]. Thus, when a group of nodes suddenly becomes more similar to another community, a flag is raised and a change-point is detected [Sun et al. 2007].

6.4.2 Role-based Embeddings. There are many applications and problem settings where role-based embeddings are more useful for graph-based anomaly detection. Role-based embeddings are often most useful for applications where anomalies can be defined with respect to the structural properties and behavior in the network. For instance, an anomaly in this setting might be when a node's structural properties/behavior differ significantly from all other nodes in the network [Akoglu et al. 2015]. Another slight variation of this problem can be defined for temporal networks as well. In particular, suppose the goal is to detect nodes with sudden changes in their structural

behavior. In this example, node anomalies may represent users (or computers) that become infected with a virus/malware in the network and thus the nodes structural properties/behavior abruptly changes [Fu et al. 2009; Ranshous et al. 2015; Rossi et al. 2013].

Many anomaly detection applications might benefit from the use of external knowledge (structural behavior/profile) that was found to be important in the detection of a new anomaly that has recently been detected in some other graph (*e.g.*, an IP communication/traceroute network from another organization/company) [Henderson et al. 2012]. Role-based embeddings are therefore most useful for this application since they represent general structural properties important in the detection of the anomaly and the specific structural properties captured in the embedding can be transferred to another arbitrary network (and thus used as a signature) for detecting this new recent anomaly (*e.g.*, the anomaly may represent a recent zero-day attack vector).

6.5 Summarization / compression

The overall goal of summarization/compression is to describe the input graph G with a compact representation [Ahmed et al. 2017; Liu et al. 2018a]. The precise way to do this fundamentally differs depending on whether communities or roles are preserved.

6.5.1 Community-based Embeddings. The majority of work in graph summarization are naturally based on community-based embeddings as they leverage the notion of communities directly. Though there are various different summarization techniques, many methods leverage grouping or clustering, and represent each group of densely connected nodes that are nearby one another (cluster or community) as a super-node and the edges between such nodes as super-edges [Koutra et al. 2014; Liu et al. 2018a; Shah et al. 2015]. More recently, the notion of latent network summarization was introduced [Jin et al. 2019c]; it leverages community-based embeddings by propagating structural features in the network via relational functions, and achieves compression by storing low-rank, size-independent structural feature matrices and the relational functions as the latent network summary.

6.5.2 Role-based Embeddings. Role discovery methods typically output a role graph (Definition 7) that succinctly represents the key structural roles and the dependencies between them [Carrington et al. 2005; Rossi and Ahmed 2015]. The role graph consists of super-nodes that represent roles and the edges between the roles are super-edges and encode the dependencies between the different roles. The role graph represents a summary of the roles and relationships between the roles. It can also be seen as a smaller model of the original graph and therefore can be viewed as a compressed representation of the overall graph as it succinctly represents the main structural patterns and the relationships between them (*e.g.*, if a star-center node connects to many star-edge nodes, then the super-node that captures the star-center role will have an edge to the super-node representing the star-edge role).

6.6 Visualization

Community and role-based embeddings are also useful in visualization applications, especially as they relate to reducing information overload and in the visualization of large graphs [Abello et al. 2006; Keim et al. 2008; Von Landesberger et al. 2011]. Recall that communities and roles are complimentary concepts (Table 1) and therefore both provide useful information when used to summarize the graph for visualization purposes.

6.6.1 Community-based Embeddings. In many visualization applications, community-based embeddings are useful when the graph/network data is too large to visualize all-at-once [Hu and Shi 2015; Newman and Girvan 2004]. In such problem settings, community-based embeddings can

be used to derive communities which are then displayed to the user in the initial visualization of the graph. This is used as a way to navigate large graphs and avoid the computational and visual problems that arise when visualizing large-scale graphs [Von Landesberger et al. 2011]. The user can then visually select the community of interest, which is then displayed to the user. In this example, once a community is selected, the user can view the nodes and edges that belong to it, while avoiding all other nodes and edges that are not of interest to the specific user/query [Abello et al. 2006].

6.6.2 Role-based Embeddings. In a similar fashion, role-based embeddings can be used for navigating large networks [Koutra and Faloutsos 2017; Rossi et al. 2018a]. Suppose the user is only interested in hub (star-center) nodes, then we can immediately visualize all such nodes and their roles while avoiding the computational and visual issues that arise when trying to visualize large graphs. Work on vocabulary- or motif-based summarization of static or time-evolving graphs can also be leveraged in visualization of structural roles [Dunne and Shneiderman 2013; Koutra et al. 2014; Shah et al. 2017]. Other similar types of (structural role-based) queries and filtering can be performed to answer other questions of interest to the user.

6.7 Clustering

Community and role-based embeddings are also clearly useful in graph clustering (Section 1.1). For simplicity, we have described roles and communities in Section 1.1 with respect to hard assignments. However, over a decade ago, methods for roles and communities that naturally output node embeddings have been investigated. The node embeddings from these methods are sometimes referred to as node mixed-membership vectors. In other words, community and role discovery are not different problems than node embeddings, since they both output node embeddings.

In this section, we discuss a few applications of communities and roles that make use of the hard assignments. We also note that some of the previous applications leveraged the hard assignments. However, we focus mainly on discussing other applications that have not yet been discussed above.

6.7.1 Community-based Embeddings. Given the community-based embeddings, we can use a clustering algorithm to derive hard cluster assignments. These hard cluster assignments have been useful for sampling-based applications. In particular, Bilgic et al. [2010] used them for relational active learning where nodes are actively sampled from every community. Hard cluster assignments can also be used to improve scalability of a variety of applications. For instance, given a node embedding of interest, we can use the k cluster centroids to find the significantly smaller set (community cluster) of relevant nodes, without having to compare to all such nodes. Similar ideas may also be useful for improving search and recommendation systems [Li and Kim 2003; Sarwar et al. 2002]. In this context, similarity in representations can also be used to identify the top-K related objects, without explicitly defining clusters. For instance, by learning propagation-based representations of objects in the personal web (i.e., heterogeneous personal information network), it is possible to identify the objects that are most relevant to specific activities or topics (e.g., work projects, vacation) [Safavi et al. 2020], and then automatically organize them into groups.

6.7.2 Role-based Embeddings. Given the structural role-based embeddings, we can derive hard cluster assignments, which can also be used in a variety of applications. In general, there are many applications where it would also make sense to sample a set of nodes that belong to the same structural role or a diverse set of nodes from different structural roles. In the above, structural roles are used for sampling. Sampling nodes from the same structural role might be useful for identifying nodes that are similar to a given node of interest. In contrast, one possibility for sampling nodes from different structural roles is for the active learning setting. Another important application

where hard assignment of roles can be used is in selecting users with similar structural behavior for recommendation or influence maximization applications. In online advertisement campaigns, the specific advertisement can be personalized better based on the role of a user in the network (*e.g.*, Facebook, Yelp) [Farahat et al. 2012]. Furthermore, a business might only be interested in targeting an individual with a certain role in the network. Role-based embeddings and the hard assignment of roles from them are also useful in a wide variety of transfer learning tasks such as across-network classification, link prediction, and finding similar nodes in general.

7 CONCLUSION

In this work, communities and roles are formally defined and used as a basis for analysis of the main mechanisms behind popular embedding methods for graph data. We have described a general framework for the study of embedding methods based on whether they are community or role-based. We have also shown formally why the mechanisms (*e.g.*, random walks, feature diffusion) behind many of the popular embedding methods give rise to community (proximity) or role-based (structural) embeddings. This formalization and theoretical analysis allows for a deeper understanding of the key mechanisms used by many existing embedding methods, and gives intuition for where such methods are most appropriate, but more importantly, provides intuition for how to develop better embedding methods for specific applications that may favor either communities (proximity) or roles (structural). In addition, we discuss applications, problem settings, and data characteristics that are best for community-based or role-based embeddings. We believe a main contribution of this work is that it allows researchers to not only gain a deeper understanding of the main mechanisms behind embedding methods (and whether they are community or role-based), but also gain insight and understanding of how to develop better embedding methods for specific applications that favor community-based or structural role-based embeddings.

ACKNOWLEDGEMENTS

The authors would like to thank Mark Heimann for his insightful feedback on this manuscript. This material is based upon work supported by the National Science Foundation under Grant No. IIS 1845491, Army Young Investigator Award No. W911NF1810397, an Adobe Digital Experience faculty research award, an Amazon faculty award, and a Google faculty award. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or other funding parties. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

REFERENCES

James Abello, Tina Eliassi-Rad, and Nishchal Devanur. 2010. Detecting novel discrepancies in communication networks. In *ICDM*. 8–17.

James Abello, Frank Van Ham, and Neeraj Krishnan. 2006. Ask-GraphView: A large scale graph visualization system. *IEEE Transactions on Visualization and Computer Graphics* 12, 5 (2006), 669–676.

Bijaya Adhikari, Yao Zhang, Naren Ramakrishnan, and B Aditya Prakash. 2018. Sub2vec: Feature learning for subgraphs. In *PAKDD*. Springer, 170–182.

Nesreen K. Ahmed, Nick Duffield, Theodore L. Willke, and Ryan A. Rossi. 2017. On Sampling from Massive Graph Streams. In *VLDB*. 1430–1441.

Nesreen K Ahmed, Jennifer Neville, and Ramana Kompella. 2014. Network sampling: From static to streaming graphs. *TKDD* 8, 2 (2014), 7.

Nesreen K. Ahmed, Jennifer Neville, Ryan A. Rossi, and Nick Duffield. 2015. Efficient Graphlet Counting for Large Networks. In *ICDM*. 10.

Nesreen K Ahmed, Jennifer Neville, Ryan A Rossi, Nick G Duffield, and Theodore L Willke. 2017a. Graphlet Decomposition: Framework, Algorithms, and Applications. *Knowledge and Information Systems (KAIS)* 50, 3 (2017), 689–722.

Nesreen K. Ahmed, Ryan A. Rossi, Theodore L. Willke, and Rong Zhou. 2017b. Edge Role Discovery via Higher-order Structures. In *PAKDD*. 291–303.

Nesreen K. Ahmed, Ryan A. Rossi, Rong Zhou, John Boaz Lee, Xiangnan Kong, Theodore L. Willke, and Hoda Eldardiry. 2017c. A Framework for Generalizing Graph-based Representation Learning Methods. In *arXiv:1709.04596*.

Nesreen K. Ahmed, Ryan A. Rossi, Rong Zhou, John Boaz Lee, Xiangnan Kong, Theodore L. Willke, and Hoda Eldardiry. 2018. Learning Role-based Graph Embeddings. In *IJCAI*.

Edoardo M. Airoldi, David M. Blei, Stephen E. Fienberg, and Eric P. Xing. 2008. Mixed Membership Stochastic Blockmodels. *Journal of Machine Learning Research* 9 (2008), 1981–2014. Issue Sep.

Leman Akoglu, Hanghang Tong, and Danai Koutra. 2015. Graph based anomaly detection and description: a survey. *DMKD* 29, 3 (2015), 626–688.

Wael Al Etaiwi and Arafat Awajan. 2018. Learning Graph Representation: A Comparative Study. In *International Arab Conference on Information Technology (ACIT)*. IEEE, 1–6.

Mohammad Al Hasan and Mohammed J Zaki. 2011. A survey of link prediction in social networks. In *Social Network Data Analytics*. 243–275.

Hélio Almeida, Dorgival Guedes, Wagner Meira, and Mohammed J Zaki. 2011. Is there a best quality metric for graph clusters?. In *ECML/PKDD*. Springer, 44–59.

Reid Andersen, Fan Chung, and Kevin Lang. 2006. Local graph partitioning using pagerank vectors. In *FOCS*. IEEE, 475–486.

Reid Andersen, Shayan Oveis Gharan, Yuval Peres, and Luca Trevisan. 2016. Almost optimal local graph clustering using evolving sets. *Journal of the ACM (JACM)* 63, 2 (2016), 15.

L. Backstrom, D. Huttenlocher, J. Kleinberg, and X. Lan. 2006. Group formation in large social networks: Membership, growth, and evolution. In *Proceeding of the 12th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*. 44–54.

Nicola Barbieri, Francesco Bonchi, and Giuseppe Manco. 2013. Cascade-based community detection. In *WSDM*. ACM, 33–42.

M. Bilgic, L. Mihalkova, and L. Getoor. 2010. Active learning for networked data. *ICML'10* (2010).

Mustafa Bilgic, Galileo Mark Namata, and Lise Getoor. 2007. Combining Collective Classification and Link Prediction. In *ICDM Workshops*. 381–386.

S.P. Borgatti and M.G. Everett. 1992. Notions of position in social network analysis. *Sociological methodology* 22, 1 (1992), 1–35.

Stephen P Borgatti, Martin G Everett, and Jeffrey C Johnson. 2018. *Analyzing social networks*. Sage.

J.P. Boyd and M.G. Everett. 1999. Relations, residuals, regular interiors, and relative regular equivalence. *Social Networks* 21, 2 (1999), 147–165.

Joan Bruna and X Li. 2017. Community detection with graph neural networks. *stat* 1050 (2017), 27.

Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. 2018. A comprehensive survey of graph embedding: Problems, techniques, and applications. *TKDE* 30, 9 (2018), 1616–1637.

Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. GraRep: Learning graph representations with global structural information. In *CIKM*. ACM, 891–900.

Peter J Carrington, John Scott, and Stanley Wasserman. 2005. *Models and methods in social network analysis*. Vol. 28. Cambridge university press.

Sandro Cavallari, Erik Cambria, Hongyun Cai, Kevin Chen-Chuan Chang, and Vincent W Zheng. 2019. Embedding Both Finite and Infinite Communities on Graphs. *IEEE Computational Intelligence Magazine* 14, 3 (2019), 39–50.

Sandro Cavallari, Vincent W Zheng, Hongyun Cai, Kevin Chen-Chuan Chang, and Erik Cambria. 2017. Learning community embedding with community detection and node embedding on graphs. In *CIKM*. 377–386.

D. Chakrabarti, R. Kumar, and A. Tomkins. 2006. Evolutionary clustering. In *Proceeding of the 12th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*.

Ines Chami, Zhitao Ying, Christopher Ré, and Jure Leskovec. 2019. Hyperbolic graph convolutional neural networks. In *Advances in Neural Information Processing Systems*. 4869–4880.

Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly detection: A survey. *Comput. Surveys* 41, 3 (2009), 15.

Jie Chen and Yousef Saad. 2010. Dense subgraph extraction with application to community detection. *IEEE Transactions on knowledge and data engineering* 24, 7 (2010), 1216–1230.

Lei Chen, Shunwang Gong, Joan Bruna, and Michael M. Bronstein. 2020a. Attributed Random Walk as Matrix Factorization. In *NeurIPS Workshop on Graph Representation Learning*.

Xiyuan Chen, Mark Heimann, Fatemeh Vahedian, and Danai Koutra. 2020b. Consistent Network Alignment via Proximity-Preserving Node Embedding. In *arXiv:2005.04725*.

Zhengzhang Chen, William Hendrix, and Nagiza F Samatova. 2012. Community-based anomaly detection in evolutionary networks. *Journal of Intelligent Information Systems* 39, 1 (2012), 59–85.

Zhengdao Chen, Lisha Li, and Joan Bruna. 2018. Supervised Community Detection with Line Graph Neural Networks. (2018).

Fan RK Chung. 1997. *Spectral graph theory*. Number 92. AMS.

Xingping Dong, Jianbing Shen, Ling Shao, and Luc Van Gool. 2016. Sub-Markov random walk for image segmentation. *IEEE Transactions on Image Processing* 25, 2 (2016), 516–527.

Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable Representation Learning for Heterogeneous Networks. In *KDD*. 135–144.

Cody Dunne and Ben Shneiderman. 2013. Motif Simplification: Improving Network Visualization Readability with Fan, Connector, and Clique Glyphs. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 3247âĂŞ3256.

David K. Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alan Aspuru-Guzik, and Ryan P. Adams. 2015. Convolutional networks on graphs for learning molecular fingerprints. In *NIPS*. 2224–2232.

Scott Emmons, Stephen Kobourov, Mike Gallant, and Katy Börner. 2016. Analysis of network clustering algorithms and cluster quality metrics at scale. *PloS one* 11, 7 (2016), e0159161.

M.G. Everett and S. Borgatti. 1991. Role colouring a graph. *Mathematical Social Sciences* 21, 2 (1991), 183–188.

M.G. Everett and S.P. Borgatti. 1994. Regular equivalence: General theory. *Journal of Mathematical Sociology* 19, 1 (1994), 29–52.

Dezhi Fang, Matthew Keezer, Jacob Williams, Kshitij Kulkarni, Robert Pienta, and Duen Horng Chau. 2017. Carina: Interactive million-node graph visualization using web browser technologies. In *WWW*. 775–776.

Ayman Farahat, Nesreen Ahmed, and Uptal Dholakia. 2012. Does a daily deal promotion signal a distressed business? an empirical investigation of small business survival. *An Empirical Investigation of Small Business Survival* (2012).

Katherine Faust. 2010. A puzzle concerning triads in social networks: Graph constraints and the triad census. *Social Networks* 32, 3 (2010), 221–233.

Timothy La Fond, Jennifer Neville, and Brian Gallagher. 2018. Designing Size Consistent Statistics for Accurate Anomaly Detection in Dynamic Networks. *TKDD* 12 (2018), 46:1–46:49.

S. Fortunato. 2010a. Community detection in graphs. *Physics Reports* 486, 3-5 (2010), 75–174.

Santo Fortunato. 2010b. Community detection in graphs. *Physics Reports* 486, 3 (2010), 75–174.

Wenjie Fu, Le Song, and Eric P Xing. 2009. Dynamic mixed membership blockmodel for evolving networks. In *Proceedings of the 26th Annual International Conference on Machine Learning*. 329–336.

Yun Fu and Yunqian Ma. 2012. *Graph embedding for pattern analysis*. Springer.

B. Gallagher, H. Tong, T. Eliassi-Rad, and C. Faloutsos. 2008. Using Ghost Edges for Classification in Sparsely Labeled Networks. In *KDD*. 256–264.

Eleanor J. Gardiner, Peter Willett, and Peter J. Artymiuk. 2000. Graph-theoretic techniques for macromolecular docking. *Journal of Chemical Information and Computer Sciences* 40, 2 (2000), 273–279. https://doi.org/10.1021/ci990262o

Wolfgang Gatterbauer, Stephan Günnemann, Danai Koutra, and Christos Faloutsos. 2015. Linearized and Single-Pass Belief Propagation. *Proc. VLDB Endow.* 8, 5, 581–592.

David Gibson, Jon Kleinberg, and Prabhakar Raghavan. 1998. Inferring web communities from link topology. In *HyperText*. 225–234.

Sean Gilpin, Tina Eliassi-Rad, and Ian Davidson. 2013. Guided Learning for Role Discovery (GLRD): Framework, Algorithms, and Applications. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 113âĂŞ121.

David F. Gleich and Ryan A. Rossi. 2014. A Dynamical System for PageRank with Time-Dependent Teleportation. *Internet Mathematics* 10, 1-2 (2014), 188–217.

Palash Goyal and Emilio Ferrara. 2018. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems* 151 (2018), 78–94.

Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *SIGKDD*. 855–864.

William Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. *arXiv:1706.02216* (2017).

Mark Heimann and Danai Koutra. 2017. On Generalizing Neural Node Embedding Methods to Multi-Network Problems. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, MLG Workshop*.

Mark Heimann, Tara Safavi, and Danai Koutra. 2019. Distribution of Node Embeddings as Multiresolution Features for Graphs. In *IEEE International Conference on Data Mining*, Jianyong Wang, Kyuseok Shim, and Xindong Wu (Eds.). 289–298.

Mark Heimann, Haoming Shen, Tara Safavi, and Danai Koutra. 2018. Regal: Representation learning-based graph alignment. In *CIKM*. 117–126.

Keith Henderson, Tina Eliassi-Rad, Spiros Papadimitriou, and Christos Faloutsos. 2010. HCDF: A hybrid community discovery framework. In *SDM*. SIAM, 754–765.

Keith Henderson, Brian Gallagher, Tina Eliassi-Rad, Hanghang Tong, Sugato Basu, Leman Akoglu, Danai Koutra, Christos Faloutsos, and Lei Li. 2012. RolX: Structural Role Extraction & Mining in Large Graphs. In *SIGKDD*. 1231–1239.

P.W. Holland and S. Leinhardt. 1981. An exponential family of probability distributions for directed graphs. *J. Amer. Statist. Assoc.* (1981), 33–50.

Paul W Holland and Samuel Leinhardt. 1976. Local structure in social networks. *Sociological methodology* 7 (1976), 1–45.

Renjun Hu, Charu C Aggarwal, Shuai Ma, and Jinpeng Huai. 2016. An embedding approach to anomaly detection. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*. IEEE, 385–396.

Yifan Hu and Lei Shi. 2015. Visualizing large graphs. *Wiley Interdisciplinary Reviews: Computational Statistics* 7, 2 (2015), 115–136.

Xiao Huang, Qingquan Song, Yuening Li, and Xia Hu. 2019. Graph recurrent networks with attributed random walks. In *KDD*. 732–740.

Tsuyoshi Idé and Hisashi Kashima. 2004. Eigenspace-based anomaly detection in computer systems. In *KDD*. 440–449.

Di Jin, Mark Heimann, Ryan A. Rossi, and Danai Koutra. 2019a. Node2BITS: Compact Time- and Attribute-aware Node Representations for User Stitching. In *ECML/PKDD*. 22.

Di Jin, Mark Heimann, Tara Safavi, Mengdi Wang, Wei Lee, Lindsay Snider, and Danai Koutra. 2019b. Smart Roles: Inferring Professional Roles in Email Networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Di Jin, Ryan A. Rossi, Eunyee Koh, Sungchul Kim, Anup Rao, and Danai Koutra. 2019c. Latent Network Summarization: Bridging Network Embedding and Summarization. In *KDD*.

Ravi Kannan, Santosh Vempala, and Adrian Vetta. 2004. On clusterings: Good, bad and spectral. *Journal of the ACM (JACM)* 51, 3 (2004), 497–515.

Daniel Keim, Gennady Andrienko, Jean-Daniel Fekete, Carsten Görg, Jörn Kohlhammer, and Guy Melançon. 2008. Visual analytics: Definition, process, and challenges. In *Information visualization*. Springer, 154–175.

Jeremy Kepner and John Gilbert. 2011. *Graph algorithms in the language of linear algebra*. SIAM.

Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. *arXiv:1611.07308* (2016).

Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. *Proceedings of the 5th International Conference on Learning Representations* (2017).

Kyle Kloster and David F Gleich. 2014. Heat kernel based community detection. In *KDD*. ACM, 1386–1395.

Eric D Kolaczyk and Gábor Csárdi. 2009. *Statistical analysis of network data with R*. Vol. 65. Springer.

R.I. Kondor and J. Lafferty. 2002. Diffusion kernels on graphs and other discrete input spaces. In *Machine Learning*. 315–322.

Danai Koutra and Christos Faloutsos. 2017. *Individual and Collective Graph Mining: Principles, Algorithms, and Applications*. Morgan & Claypool Publishers.

Danai Koutra, U Kang, Jilles Vreeken, and Christos Faloutsos. 2014. VoG: Summarizing and Understanding Large Graphs. In *SIAM International Conference on Data Mining*. 91–99.

Lauri Kovanen, Márton Karsai, Kimmo Kaski, János Kertész, and Jari Saramäki. 2011. Temporal motifs in time-dependent networks. *Journal of Statistical Mechanics: Theory and Experiment* 2011, 11 (2011), P11005.

Mehmet Koyutürk, Yohan Kim, Umut Topkara, Shankar Subramaniam, Wojciech Szpankowski, and Ananth Grama. 2006. Pairwise alignment of protein interaction networks. *JCB* 13, 2 (2006), 182–199.

Timothy La Fond and Jennifer Neville. 2010. Randomization Tests for Distinguishing Social Influence and Homophily Effects. In *WWW*. ACM, 601–610.

Yi-An Lai, Chin-Chi Hsu, Wen Hao Chen, Mi-Yen Yeh, and Shou-De Lin. 2017. Prune: Preserving proximity and global ranking for network embedding. In *Advances in neural information processing systems*. 5257–5266.

Jean-Louis Lassez, Ryan A. Rossi, and Kumar Jeev. 2008. Ranking Links on the Web: Search and Surf Engines. *New Frontiers in Applied Artificial Intelligence (IEA/AIE)* (2008), 199–208.

John Boaz Lee, Xiangnan Kong, Constance M Moore, and Nesreen K Ahmed. 2020. Deep Parametric Model for Discovering Group-cohesive Functional Brain Regions. In *SIAM International Conference on Data Mining (SDM)*. SIAM, 631–639.

John Boaz Lee, Ryan A. Rossi, and Xiangnan Kong. 2018a. Graph Classification using Structural Attention. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

John Boaz Lee, Ryan A. Rossi, Xiangnan Kong, Sungchul Kim, Eunyee Koh, and Anup Rao. 2018b. Higher-order Graph Convolutional Networks. In *arXiv:1809.07697*. 1–8.

Qing Li and Byeong Man Kim. 2003. Clustering approach for hybrid recommender system. In *Proceedings IEEE/WIC International Conference on Web Intelligence (WI 2003)*. IEEE, 33–38.

Shuyang Lin, Qingbo Hu, Guan Wang, and S Yu Philip. 2015. Understanding community effects on information diffusion. In *PAKDD*. Springer, 82–95.

Xi Liu, Muhe Xie, Xidao Wen, Rui Chen, Yong Ge, Nick Duffield, and Na Wang. 2018b. A semi-supervised and inductive embedding model for churn prediction of large-scale mobile games. In *ICDM*. IEEE, 277–286.

Xi Liu, Muhe Xie, Xidao Wen, Rui Chen, Yong Ge, Nick Duffield, and Na Wang. 2019. Micro-and macro-level churn analysis of large-scale mobile games. *Knowledge and Information Systems* (2019), 1–32.

Yike Liu, Tara Safavi, Abhilash Dighe, and Danai Koutra. 2018a. Graph summarization methods and applications: A survey. *ACM Computing Surveys (CSUR)* 51, 3 (2018), 62.

F. Lorrain and H.C. White. 1971. Structural equivalence of individuals in social networksâĂă. *Journal of Mathematical Sociology* 1, 1 (1971), 49–80.

J.J. Luczkovich, S.P. Borgatti, J.C. Johnson, and M.G. Everett. 2003. Defining and measuring trophic role similarity in food webs using regular equivalence. *Journal of Theoretical Biology* 220, 3 (2003), 303–321.

Guixiang Ma, Nesreen K Ahmed, Theodore L Willke, Dipanjan Sengupta, Michael W Cole, Nicholas B Turk-Browne, and Philip S Yu. 2019b. Deep graph similarity learning for brain data analysis. (2019), 2743–2751.

Guixiang Ma, Nesreen K Ahmed, Theodore L Willke, and Philip S Yu. 2019a. Deep Graph Similarity Learning: A Survey. *arXiv:1912.11615* (2019).

Luke K McDowell, Kalyan Moy Gupta, and David W Aha. 2009. Cautious collective classification. *JMLR* 10, Dec (2009), 2777–2836.

Marina Meila and Jianbo Shi. 2001a. Learning segmentation by random walks. In *NIPS*. 873–879.

Marina Meila and Jianbo Shi. 2001b. A random walks view of spectral segmentation. (2001).

Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. 2002. Network motifs: simple building blocks of complex networks. *Science* 298, 5594 (2002), 824–827.

Annamalai Narayanan, Mahinthan Chandramohan, Lihui Chen, Yang Liu, and Santhoshkumar Saminathan. 2016. subgraph2vec: Learning distributed representations of rooted sub-graphs from large graphs. *arXiv:1606.08928* (2016).

Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu, and Shantanu Jaiswal. 2017. graph2vec: Learning distributed representations of graphs. *arXiv:1707.05005* (2017).

J. Neville, O. Şimşek, and D. Jensen. 2004. Autocorrelation and Relational Learning: Challenges and Opportunities. In *Proceedings of the Workshop on Statistical Relational Learning*.

Jennifer Neville and David Jensen. 2000. Iterative classification in relational data. In *AAAI SRL Workshop*. 13–20.

Jennifer Neville and David Jensen. 2005. Leveraging relational autocorrelation with latent group models. In *ICDM*. 8.

M.E.J. Newman. 2004. Fast algorithm for detecting community structure in networks. *Physical Review E* 69, 6 (2004), 066133.

M.E.J. Newman and M. Girvan. 2004. Finding and evaluating community structure in networks. *Physical review E* 69, 2 (2004), 26113.

Andrew Y Ng, Michael I Jordan, and Yair Weiss. 2002. On spectral clustering: Analysis and an algorithm. In *NIPS*. 849–856.

Giang Hoang Nguyen, John Boaz Lee, Ryan A Rossi, Nesreen K Ahmed, Eunyee Koh, and Sungchul Kim. 2018. Continuous-time dynamic network embeddings. In *WWW*. 969–976.

Vincenzo Nicosia, John Tang, Cecilia Mascolo, Mirco Musolesi, Giovanni Russo, and Vito Latora. 2013. Graph metrics for temporal networks. In *Temporal Networks*. Springer, 15–40.

Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. 2016. Learning convolutional neural networks for graphs. In *ICML*. 2014–2023.

Giannis Nikolentzos, Polykarpos Meladianos, and Michalis Vazirgiannis. 2017. Matching Node Embeddings for Graph Similarity. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*.

Giannis Nikolentzos and Michalis Vazirgiannis. 2019. Learning Structural Node Representations using Graph Kernels. *IEEE Transactions on Knowledge and Data Engineering* (2019).

Pekka Orponen and Satu Elisa Schaeffer. 2005. Local clustering of large graphs by approximate Fiedler vectors. In *International Workshop on Experimental and Efficient Algorithms*. Springer, 524–533.

Pekka Orponen, Satu Elisa Schaeffer, and Vanesa Ávalos Gaytán. 2008. Locally computable approximations for spectral clustering and absorption times of random walks. *arXiv:0810.4061* (2008).

Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. 2016. Asymmetric transitivity preserving graph embedding. In *KDD*. 1105–1114.

Leto Peel. 2017. Graph-based semi-supervised learning for relational networks. In *SDM*. SIAM, 435–443.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *SIGKDD*. 701–710.

Robert Pienta, James Abello, Minsuk Kahng, and Duen Horng Chau. 2015. Scalable graph exploration and visualization: Sensemaking challenges and opportunities. In *BigComp*. 271–278.

Nataša Pržulj. 2007. Biological network comparison using graphlet degree distribution. *Bioinfo.* 23, 2 (2007), e177–e183.

Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. 2018. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *WSDM*. ACM, 459–467.

Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. 2007. Near linear time algorithm to detect community structures in large-scale networks. *Physical review E* 76, 3 (2007), 036106.

Stephen Ranshous, Shitian Shen, Danai Koutra, Steve Harenberg, Christos Faloutsos, and Nagiza F Samatova. 2015. Anomaly detection in dynamic networks: a survey. *Wiley Interdisciplinary Reviews: Computational Statistics* 7, 3 (2015), 223–247.

Bruno Ribeiro, Pinghui Wang, Fabricio Murai, and Don Towsley. 2012. Sampling directed graphs with random walks. In *INFOCOM*. IEEE, 1692–1700.

Leonardo F.R. Ribeiro, Pedro H.P. Saverese, and Daniel R. Figueiredo. 2017. Struc2Vec: Learning Node Representations from Structural Identity. In *KDD*.

Everett M Rogers and Dilip K Bhowmik. 1970. Homophily-heterophily: Relational concepts for communication research. *Public opinion quarterly* 34, 4 (1970), 523–538.

Ryan A. Rossi and Nesreen K. Ahmed. 2015. Role Discovery in Networks. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 27, 4 (April 2015), 1112–1131.

Ryan A. Rossi, Nesreen K. Ahmed, Hoda Eldardiry, and Rong Zhou. 2018a. Interactive Visual Graph Mining and Learning. *ACM Transactions on Intelligent Systems and Technology (TIST)* (2018), 1–30. http://graphML.com

Ryan A. Rossi, Nesreen K. Ahmed, Eunyee Koh, Sungchul Kim, Anup Rao, and Yasin Abbasi-Yadkori. 2018b. HONE: Higher-Order Network Embeddings. *WWW* (2018).

Ryan A. Rossi, Brian Gallagher, Jennifer Neville, and Keith Henderson. 2013. Modeling Dynamic Behavior in Large Evolving Graphs. In *WSDM*. 667–676.

Ryan A. Rossi, Luke K. McDowell, David W. Aha, and Jennifer Neville. 2012. Transforming graph data for statistical relational learning. *Journal of Artificial Intelligence Research* 45, 1 (2012), 363–441.

Ryan A. Rossi and Jennifer Neville. 2012. Time-Evolving Relational Classification and Ensemble Methods. In *PAKDD*. Vol. 7301. Springer, 1–13.

Ryan A. Rossi, Rong Zhou, and Nesreen K. Ahmed. 2017. Deep Feature Learning for Graphs. In *arXiv:1704.08829*.

Ryan A. Rossi, Rong Zhou, Nesreen K. Ahmed, and Hoda Eldardiry. 2018. Relational Similarity Machines (RSM): A Similarity-based Learning Framework for Graphs. In *IEEE BigData*. 10.

Benedek Rozemberczki, Carl Allen, and Rik Sarkar. 2019. Multi-scale Attributed Node Embedding. *arXiv:1909.13021* (2019).

Tara Safavi, Adam Fourney, Robert Sim, Marcin Juraszek, Shane Williams, Ned Friend, Danai Koutra, and Paul N. Bennett. 2020. Toward Activity Discovery in the Personal Web. In *The Thirteenth ACM International Conference on Web Search and Data Mining*, James Caverlee, Xia (Ben) Hu, Mounia Lalmas, and Wei Wang (Eds.). 492–500.

Lee Douglas Sailer. 1978. Structural equivalence: Meaning and definition, computation and application. *Social Networks* 1, 1 (1978), 73–90.

Badrul M. Sarwar, George Karypis, Joseph Konstan, and John Reidl. 2002. Recommender Systems for Large-Scale E-Commerce: Scalable Neighborhood Formation Using Clustering. In *Proceedings of the 5th International Conference on Computer and Information Technology (ICCIT)*.

Victor Garcia Satorras and Joan Bruna Estrach. 2018. Few-Shot Learning with Graph Neural Networks. In *ICLR*.

Satu Elisa Schaeffer. 2007. Graph clustering. *Computer science review* 1, 1 (2007), 27–64.

P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad. 2008. Collective classification in network data. *AI Magazine* 29, 3 (2008), 93.

Neil Shah, Danai Koutra, Lisa Jin, Tianmin Zou, Brian Gallagher, and Christos Faloutsos. 2017. On Summarizing Large-Scale Dynamic Graphs. *IEEE Data Eng. Bull.* 40, 3 (2017), 75–88.

Neil Shah, Danai Koutra, Tianmin Zou, Brian Gallagher, and Christos Faloutsos. 2015. Timecrunch: Interpretable dynamic graph summarization. In *Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1055–1064.

Oleksandr Shchur and Stephan Günnemann. 2018. Overlapping Community Detection with Graph Neural Networks. (2018).

Daniel A Spielman and Shang-Hua Teng. 2013. A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning. *SIAM J. Comput.* 42, 1 (2013), 1–26.

Balasubramaniam Srinivasan and Bruno Ribeiro. 2019. On the Equivalence between Node Embeddings and Structural Graph Representations. *arXiv:1910.00452* (2019).

J. Sun, C. Faloutsos, S. Papadimitriou, and P.S. Yu. 2007. Graphscope: Parameter-free mining of large time-evolving graphs. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Aynaz Taheri, Kevin Gimpel, and Tanya Berger-Wolf. 2018. Learning graph representations with recurrent neural network autoencoders. In *Proc. KDD Deep Learn. Day*. 1–8.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale Information Network Embedding. In *WWW*. 1067–1077.

Lei Tang and Huan Liu. 2011. Leveraging social media networks for classification. *Data Mining and Knowledge Discovery* 23, 3 (2011), 447–478.

Stijn Marinus Van Dongen. 2000. *Graph clustering by flow simulation.* Ph.D. Dissertation.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.

S Vichy N Vishwanathan, Nicol N Schraudolph, Risi Kondor, and Karsten M Borgwardt. 2010. Graph kernels. *JMLR* 11 (2010), 1201–1242.

Tatiana Von Landesberger, Arjan Kuijper, Tobias Schreck, Jörn Kohlhammer, Jarke J van Wijk, J-D Fekete, and Dieter W Fellner. 2011. Visual analysis of large graphs: state-of-the-art and future research challenges. In *Computer graphics forum*, Vol. 30. Wiley Online Library, 1719–1749.

Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. 2017. Community Preserving Network Embedding. In *AAAI*. 203–209.

D.R. White and K.P. Reitz. 1983. Graph and semigroup homomorphisms on networks of relations. *Social Networks* 5, 2 (1983), 193–234.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. 2019a. A comprehensive survey on graph neural networks. *arXiv:1901.00596* (2019).

Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019b. Graph wavenet for deep spatial-temporal graph modeling. *arXiv:1906.00121* (2019).

Biao Xiang, Ziqi Liu, Jun Zhou, and Xiaolong Li. 2018. Feature Propagation on Graph: A New Perspective to Graph Representation Learning. *arXiv:1804.06111* (2018).

Ma Xuewei, Geng Qin, Zhiyang Qiu, Mingxin Zheng, and Zhe Wang. 2019. RiWalk: Fast Structural Node Embedding via Role Identification. In *IEEE 19th International Conference on Data Mining*.

Yujun Yan, Jiong Zhu, Marlena Duda, Eric Solarz, Chandra Sripada, and Danai Koutra. 2019. GroupINN: Grouping-based Interpretable Neural Network for Classification of Limited, Noisy Brain Data. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 772–782.

Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Chang. 2015. Network representation learning with rich text information. In *IJCAI*.

Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. 2019. GNNexplainer: Generating explanations for graph neural networks. In *Advances in Neural Information Processing Systems*. 9240–9251.

Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. 2018. Hierarchical graph representation learning with differentiable pooling. In *Proceedings of the 31st Annual Conference on Neural Information Processing Systems*. 4800–4810.

Jiaxuan You, Haoze Wu, Clark Barrett, Raghuram Ramanujan, and Jure Leskovec. 2019. G2SAT: Learning to Generate SAT Formulas. In *Advances in Neural Information Processing Systems*. 10552–10563.

Chunyou Zhang, Xiaoqiang Wu, Wei Yan, Lukun Wang, and Lei Zhang. 2019. Attribute-Aware Graph Recurrent Networks for Scholarly Friend Recommendation Based on Internet of Scholars in Scholarly Big Data. *IEEE Transactions on Industrial Informatics* (2019).

Daokun Zhang, Jie Yin, Xingquan Zhu, and Chengqi Zhang. 2016. Homophily, structure, and content augmented network representation learning. In *ICDM*. IEEE, 609–618.

Daokun Zhang, Jie Yin, Xingquan Zhu, and Chengqi Zhang. 2018. Network representation learning: A survey. *IEEE transactions on Big Data* (2018).

Jiawei Zhang, Xiangnan Kong, and S Yu Philip. 2013. Predicting social links for new users across aligned heterogeneous social networks. In *ICDM*. IEEE, 1289–1294.

Yu Zhao, Zhiyuan Liu, and Maosong Sun. 2015. Representation learning for measuring entity relatedness with rich information. In *AAAI*.