Role Discovery in Networks

Ryan A. Rossi and Nesreen K. Ahmed

Abstract—Roles represent node-level connectivity patterns such as star-center, star-edge nodes, near-cliques or nodes that act as bridges to different regions of the graph. Intuitively, two nodes belong to the same role if they are structurally similar. Roles have been mainly of interest to sociologists, but more recently, roles have become increasingly useful in other domains. Traditionally, the notion of roles were defined based on graph equivalences such as structural, regular, and stochastic equivalences. We briefly revisit these early notions and instead propose a more general formulation of roles based on the similarity of a feature representation (in contrast to the graph representation). This leads us to propose a taxonomy of three general classes of techniques for discovering roles that includes (i) graph-based roles, (ii) feature-based roles, and (iii) hybrid roles. We also propose a flexible framework for discovering roles using the notion of similarity on a feature-based representation. The framework consists of two fundamental components: (a) role feature construction and (b) role assignment using the learned feature representation. We discuss the different possibilities for discovering feature-based roles and the tradeoffs of the many techniques for computing them. Finally, we discuss potential applications and future directions and challenges.

Index Terms—Roles, role discovery, role learning, feature-based roles, structural similarity, unsupervised learning

1 Introduction

ROLE DISCOVERY first arose in sociology [1], [2] where roles were used to explain the specific function of a person in society (such as a father, doctor, student, or an academic advisor) [3]. These roles as defined by sociologists are known specifically as social roles [4]. From there, role discovery naturally became an important topic in social network analysis [5], [6], [7].

In the past, role discovery has primarily been of interest to sociologists who studied roles of actors in extremely small offline social networks (e.g., graphs with tens of nodes) [7], [8], [9], [10]. Recently, role discovery is being explored in several other settings such as online social networks [11], technological networks [12], [13], biological networks [14], [15], web graphs [16], among many others [17]. While the concept of role discovery is indeed important for general graph mining and exploratory analysis, it can also be useful in many practical applications. For example, roles might be used for detecting anomalies in technological networks such as IP-traces [12], [13]. An anomaly in this setting might be a node that doesn't fit any of the roles (normal structural patterns) or it may also be defined as a role that deviates from the common/normal roles, so that any node assigned to this unusual role would be anomalous [18]. Another use might be in online advertising campaigns [19] for online social networks (Facebook, Groupon, Yelp) where ads could be customized based on the users' roles in the network. In addition, a business might only be interested in advertising to a person with a certain role in the network. Furthermore, roles are becoming an important tool with potential applications such as classification, active learning, network sampling, anonymization, among many others (see Table 1 for a summary). Despite the various applications, the task of role discovery has only received a limited amount of attention (e.g., compared to the task of community partitioning [20], [21], [22], [23], [24], [25]).

Role discovery was first defined as any process that divides the nodes of a graph into classes of structural equivalent nodes [4]. These classes of structural equivalent nodes are known as roles. Intuitively, two nodes are structurally equivalent if they are connected to the rest of the network in identical ways. There have been many attempts to relax the criterion of equivalence, e.g., regular equivalence [26], stochastic equivalence [27]. For practical purposes, the notion of equivalence can be generally relaxed to get at some form of *structural similarity*. Thus, in this work, we replace the notion of equivalence by the notion of similarity (a weaker but more practical notion).

We can now redefine role discovery appropriately using this relaxation. *Role discovery*, informally speaking, is any process that takes a graph and picks out sets of nodes with similar structural patterns. Intuitively, two nodes belong to the same role if they are structurally similar. Roles of a graph represent the main node-level connectivity patterns such as star-center/edge nodes, peripheral nodes, near-clique nodes, bridge nodes that connect different regions of the graph, among many other types of connectivity patterns (See Figure 1). In that example, the roles are defined in a strictly local sense, but in general roles may represent extremely complex node-centric structural patterns that fundamentally depend on the domain and underlying process that governs the graph. Feature-based roles naturally generalize beyond the notion of structural similarity to the notion of similarity between features which includes pure structural features as well as features learned from an initial set of

• *R. A. Rossi and N. K. Ahmed are with the Department of Computer Science, Purdue University, West Lafayette, IN, 47907.*
E-mail: rrossi@purdue.edu, nkahmed@purdue.edu

(non-relational) attributes (see Section 2.3 and Section 4 for more details).

As an aside, the notion of "structural" roles is significantly different than that of communities (and thus outside the scope of this survey). More specifically, communities are sets of nodes with more connections inside the set than outside [20], whereas roles are sets of nodes that are more structurally similar to nodes inside the set than outside. Intuitively, roles represent "significant" structural patterns (e.g., star-centers, near-cliques, bridge-nodes, star-edge nodes) and two nodes belong to the same role if they have similar structural patterns [3], [18]. In contrast, community methods assign nodes to sets based on the notions of density (and cohesion, proximity/closeness in the graph) [20]. Therefore, it is clear that community methods have a completely different objective (partition nodes into sets based on density/cohesion/proximity) and often assigns every node to a community. Figure 1 illustrates a few of these differences and gives further examples.

1.1 Scope of this Article

This article focuses on examining and categorizing techniques for computing roles from a feature-based representation. First, we propose a taxonomy for role-discovery which includes graph-based roles and feature-based roles. After briefly reviewing the traditional graph-based roles in Section 2, the remainder of the article focuses on feature-based roles. In particular, we propose the general problem of feature-based roles and provide a taxonomy and framework for it. We formulate feature-based roles from a machine learning perspective which provides a natural basis for surveying and reinterpreting techniques for use in discovering roles. Our feature-based role discovery framework succinctly characterizes the *a*pplication-specific decisions for discovering feature-based roles such as the types of features to use (graph-based, node, link or non-relational attributes), the graph feature operators to use (e.g., the specific aggregates, set ops, path/walk-based measures, subgraph patterns), whether to learn features automatically or manually, among many other decisions including role assignment methods for a feature-representation and selecting the number of roles (model selection).

Due to the no-free-lunch theorem [40], [41], it is impossible for a single role discovery method to always work better than another, given that there is no assumption on the data used in learning roles [40], [41], [42], [43], [44]. Therefore, we instead propose a general framework for feature-based roles that can serve as a fundamental basis for understanding, analyzing, and comparing methods for discovering feature-based roles. In addition, the framework provides a guide for designing role discovery techniques that are suitable for user-defined applications with known phenomenon/assumptions. Along these lines, we survey and discuss the relevant techniques, decisions, and their advantages and

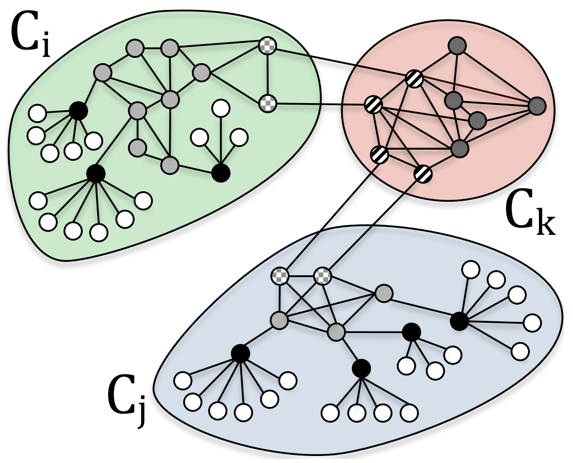


Fig. 1. In the example above, the three communities (i.e., C_i, C_j, C_k) are cohesive sets of nodes. We also note that the nodes with similar shades/markings are assigned the same node-centric roles. As shown above, roles represent structural patterns and two nodes belong to the same role if they have similar structural patterns. Furthermore, notice that the nodes in a community are close together (i.e., distance/proximity-wise) whereas the nodes in a particular role are not required to be connected or close in terms of graph distance, they only need to be more structurally similar to that role than any other. The example above is based on observed roles in large-scale technological networks such as the Internet AS topology. In that example, nodes are assigned to roles based on the notion of regular equivalence 2.1.3. As such, two nodes are "regularly equivalent" (play the same role) if they connect to equivalent others. Clearly, nodes in the same role (e.g., star-center nodes) may be in different communities and thus the notion of role is independent of distance/proximity whereas nodes in the same community must be close to one another (small distance/proximity as defined by the fundamental property/notion of a community). Another significant difference is that roles generalize as they may be learned on one graph and extracted over another, whereas communities do not.

disadvantages, and interpretation for roles. In addition, some techniques surveyed were used for other tasks and are reinterpreted/adapted for the role discovery problem (i.e., significant node-centric structural patterns). This article does not attempt to survey types of blockmodels [45] or other types of graph-based roles, nor do we attempt to survey community partitioning and related methods as these focus on an entirely different goal/objective than the one examined in this article.

1.2 Approach and Organization of Article

This article proposes a taxonomy for the role discovery problem which includes both learning of node and edge roles using a graph-based representation and a feature-based representation. Initially, roles have been computed using the graph representation directly – i.e, graph-based roles (graph $\rightarrow$ roles), (e.g. stochastic block models [5], [6], [7], [8], [32], [46]). Recently, there has been a trend of research for computing roles from a feature representation – i.e, feature-based roles (graph $\rightarrow$ features $\rightarrow$ roles). The feature representation process includes all the methods that can be used to transform the graph into an appropriate set of features for which roles can be defined over. More precisely, we transform the graph into a new representation (feature-based) from which the equivalences for the roles are computed. The

TABLE 1
Summary of the role tasks and applications. We summarize a few of the tasks where the feature-based roles might be useful.

Role Tasks	Goal/definition
Dynamic roles	Use roles for descriptive and predictive modeling of dynamic networks [18], [28].
Classification	Use role memberships as features for statistical relational learning algorithms (relational/collective classification) [29], [30], [31].
Visualizations	Roles may be used to visualize and capture the relevant differences and important patterns in big graph data [28], [32], [33].
Graph similarity	Given two graphs G and H, extract features and roles from each, and compare them [13].
Entity resolution	Given two graphs G_i and G_j (e.g., Facebook and twitter social networks), use feature-based roles to resolve node entities (predict which entity/node in G_i pertains to the exact same node in G_j) [34].
Anomaly detection	Given a graph G, find anomalous nodes (or links) with unusual role-memberships (static graph-based anomaly) or nodes with unusual role transitions (dynamic graph-based anomaly) [18].
Transfer learning	Learn roles on the graph G_i, and use these to learn the same set of roles on another network to improve accuracy [35], [36].
Graph compression	Roles capture the main structural patterns in a graph and may be used as a lossy graph compression technique [32], [33].
Queries/search	Find the top-k nodes (links) with the most similar roles (structural behavior) [33].
Active learning	Instead of selecting nodes via communities [37], roles may be used to select nodes with diverse structural patterns/roles.
Anonymization	Use roles to compute a new representation to preserves the privacy and anonymize the vertex/edge identities [38].
Network sampling	Sample a network based on the feature-based roles to ensure that all roles are represented in the sampled graph [39].

set of features can be computed from node or link features, and non-relational attributes. While this paper aims to survey both graph-based and feature-based roles, we additionally provide the foundations of a generic framework for feature-based role discovery. Note that one may also derive roles using a logical representation such as the work in [47], [48].

This survey also makes the following contributions:

- A precise formulation and taxonomy of the role discovery problem including motivation, techniques, evaluation, and applications
- Brief survey of the current research in role discovery
- A generic framework for feature-based role discovery
- A taxonomy for constructing features for roles

This article is organized as follows. Section 2 discusses the difference between traditional roles and our feature-based roles. This includes definitions and algorithms for computing each of these types of roles. We also provide examples of roles and communities and discuss the differences of each task. In Section 3, we propose a general framework for discovering *feature-based roles* and give an overview of it. Section 4 surveys and discusses past work on feature construction and reinterprets it for role discovery. Afterwards, Section 5 discusses techniques that can be used to assign roles to vertices based on the set of learned features. Section 6 discusses challenges and new directions.

2 Role Foundations

Role discovery can be generally defined as any process that divides the nodes into classes of equivalent nodes. These classes of equivalent nodes form the foundation of what are known as roles. Note that the classes here can be thought of social functions for the nodes in the network (e.g., a function may be a father or student). However, the above definition relies on some formal notion of node equivalence which is used to divide the nodes into their respective roles. As such, all nodes that have a certain role should be equivalent under the predefined node equivalence relation. More precisely, assume we are given a graph $G = (V, E)$, let $r(u)$ and $r(v)$ be the role class of nodes u and v respectively – $\forall u, v \in V, r(u) = r(v) \iff u \equiv v$. Now, the question is how to define the node equivalence relation. For example, are two nodes equivalent if they have connections to exactly the same neighbors?

We briefly review the fundamental node equivalences as they will be useful to analyze the different types of roles (graph-based, feature-based, and hybrid role methods) and the corresponding approaches for each type. See Figure 2 for an intuitive taxonomy of the three fundamental types of role discovery methods.

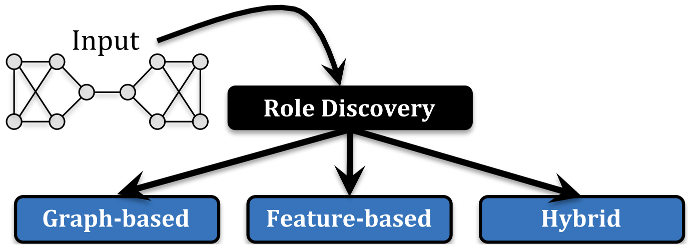


Fig. 2. **Taxonomy of Role Discovery Methods**. We categorize methods for computing roles into graph-based roles, feature-based roles, and hybrid approaches. Graph-based roles are computed from the graph directly (without incorporating any features). In contrast, feature-based roles are computed from a transformation of the graph data into a new feature representation for which roles are discovered. Naturally, there can also be hybrid approaches that leverage the advantages of both.

2.1 Graph-based Role Equivalences

We first review the important equivalences that have formed the basis for roles in much of the literature. The equivalences are discussed below from most strict to least. Let us note that one possible direction for future work will be the extension or formulation of these equivalences for role discovery in temporal or streaming networks [18], [28], [33], [49].

2.1.1 Structural equivalence

Structural equivalence [4] states that equivalent nodes have the same connection pattern to the exact same neighbors (i.e., in/out neighbors if edges are directed). Therefore, two nodes v_i and v_j are structurally equivalent if they have the same exact neighbors, hence

$\mathcal{N}(v_i) = \mathcal{N}(v_j)$. For example, the set W of nodes in Figure 3 are structurally equivalent since each node connects to exactly the same (identical) neighbors. This implies that structurally equivalent nodes are essentially indistinguishable in the sense that they have same degree, clustering coefficient, centrality, belong to the same cliques, etc. Obviously, structural equivalence is too strict of a notion and thus impractical for (large) real-world graphs. The fundamental disadvantage of structural equivalence is that it confuses similarity with closeness. This arises due to the requirement that two nodes have the same *exact* neighbors. In fact, nodes that are structurally equivalent can never be more than two links away. As a result, there have been many relaxations of structural equivalence [26], [27], [50], [51], [52].

2.1.2 Automorphic equivalence

A mapping p from one graph to another is an isomorphism if whenever $u \rightarrow v$, then $p(u) \rightarrow p(v)$. Isomorphisms are mappings from one graph to another that preserve the structure of a graph. An automorphism is an isomorphism from one graph to the *same* graph, thus preserving the symmetries. A node u is *automorphically* equivalent to node v if there exists an automorphism p such that $u = p(v)$ [27]. As an aside, automorphic equivalence can be viewed as a relaxation of structural equivalence since any set of structural equivalences are also automorphic equivalences. More intuitively, structural equivalence essentially asks if a *single node* can be exchanged for another while preserving the connections/relationships of that node, whereas automorphic equivalence is based on *sets of nodes* whom are exchangeable as subgraphs. In Figure 3, the nodes in the combined set $W \cup M$ form a single class of exchangeable nodes (also known as a role). Similarly, the nodes v_1 and v_2 from Figure 3 form another role since both nodes are exchangeable if nodes of other classes are also exchanged.

2.1.3 Regular equivalence

Regular equivalence relaxes the notion of role further to capture the social role concept better. In particular, regular equivalence is based on the idea that nodes play the same role if they are connected to *role-equivalent nodes*. This is in contrast to structural equivalence where the nodes have to be connected to identical nodes. In other words, regular equivalence states that nodes play the same role if they have similar connections to nodes of other roles[1] [26], [51]. Intuitively, two nodes that are regularly equivalent (i.e., same role) do not necessarily have to connect to the same neighbors or even the same number of neighbors, but they must be connected to *role-equivalent neighbors*. From Figure 3, notice that the nodes in the set $W \cup M$ are regularly equivalent (unique role), while v_1 and v_2 form another class of regularly

1. Nodes that are structurally or automorphically equivalent are also regularly equivalent, but the inverse is not true.

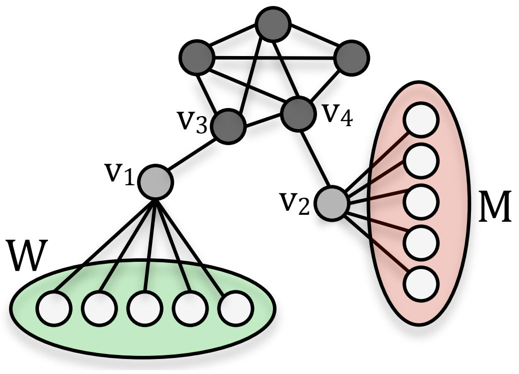


Fig. 3. This illustration reveals the fundamental differences between the main role equivalences including structural, automorphic, regular, and stochastic equivalences. For example, the set of nodes W are structurally equivalent forming a role whereas the nodes in M are structurally equivalent with one another forming another distinct role, whereas the combined set of nodes $W \cup M$ are automorphically equivalent thus representing a single role when using the relaxed automorphic equivalence.

equivalent nodes (2nd role). Further, v_3 and v_4 represent the 3rd role while the last three nodes form the fourth class of regularly equivalent nodes. For instance, the nodes in the third role are v_3 and v_4, since all nodes in that set have at least one edge to a node in the 2nd role, while also connected to nodes in the fourth role. Notice that regular equivalences can be exact or approximate, and hence there might be many valid ways of grouping nodes into equivalence sets for a given graph. However, this definition is still strict in the sense that the role of a node is tied to all nodes of that role, rather than some nodes [53].

2.1.4 Stochastic equivalence

For this reason, stochastic equivalence was introduced [27], which informally says that for a probability distribution of edges in a graph, an assignment of roles is a stochastic equivalent, if nodes with the same role have the same probability distribution of edges with other nodes. Intuitively, the nodes are organized into roles such that the the probability of a node linking with all other nodes of the graph are the same for nodes of the same role. Another interpretation is that the probability distribution of the graph must remain the same when equivalent nodes are exchanged [46]. From the example in Figure 3, observe that stochastic equivalence gives rise to the same equivalence classes obtained using the notion of regular equivalence due to the simplicity of the graph. This notion gives rise to stochastic blockmodels [5], [6], [7], [8], [46], which comes from weakening and extending the algebraic notion of structural equivalent nodes.

2.2 Methods for Graph-based Roles

We can define graph-based roles as those computed directly from the graph representation, which is typically in the form of an adjacency matrix. This is in contrast to feature-based roles which are computed indirectly from the graph by first transforming the graph representation into a feature-vector representation. Feature-based roles are the main focus of this paper, but for completeness we briefly discuss a few approaches for computing graph-based roles.

2.2.1 Blockmodels

Blockmodels are by far the most popular class of role techniques in social network analysis [5], [6], [9], [28], [32], [46], [49], [54]. Blockmodels naturally represent a network by a compact representation known as a role-interaction graph (or image matrix) where the nodes represent roles (or blocks, positions) and the edges are interactions between roles. This smaller comprehensible structure can be interpreted more readily. These methods attempt to group nodes according to the extent that they are structurally equivalent or stochastically equivalent. Recently, many types of blockmodels have been proposed such as stochastic blockmodels [10], generalized blockmodels [8], and mixed-membership stochastic blockmodels [32] (MMSB). They have also been extended for various applications [28], [32], [49], [54]. Nevertheless, a complete survey of these graph-based role methods are beyond the scope of this paper, see [45] for more details. However, we do provide a brief overview of some of the main methods below.

One of the first methods used for computing a type of blockmodel was CONCOR (convergence of iterated correlations), which was proposed by Breiger *et al.* [55]. This method initially computes the correlation of the adjacency matrix $\mathsf{Corr}(\mathbf{A}, \mathbf{A})$ which we denote as $\mathbf{C}_0$, then the correlation matrix C_1 is computed from the previous correlation matrix $\mathbf{C}_0$, and this process is repeated until all entries are either 1 or -1. Let us note that even though this method was originally designed for blockmodels, it is fundamentally based on similarity using the adjacency matrix and does not have an explicit criterion function in the usual sense of an optimization problem. On the other hand, blockmodels are typically formulated as optimization problems with a well-formed objective function. However, in general, there are many ways to compute blockmodels (e.g., direct and indirect approaches) [8], [56], [57], [58].

Stochastic blockmodels (SBM) [46] on the other hand adopts the notion of stochastic equivalence [27]. The benefit is that these probabilistic models allow for deviations between the observations and relaxes the idealized concept of exact equivalence. These models are a type of latent space models [59], but also fall into the category of random graph models [45]. One simple example of a blockmodel is a model that assigns each node to one of the several roles (or blocks)[2]. In addition, there is a set of probabilities $p_{i,j}$ that specifies the probability of an edge between a node in role i and a node in role j. In other words, how likely is it that a node in role i has an edge between a node in role j? This is typically represented as a matrix $\mathbf{P} \in \mathbb{R}^{k \times k}$ and can be either specified by the user or inferred from the data. In general, these models can be used to generate a random graph by specifying the probabilities in $\mathbf{P}$ or these probabilities may be inferred from data. Essentially, MMSB allows nodes to take part in multiple roles while also allowing roles to link to other roles probabilistically.), then one could place large weights on the diagonal of $\mathbf{P}$, which indicates that nodes of the same role have a large probability of having an edge between each other and low probability of having an edge to a node in another role.

More recently, Airoldi *et al.* [32] proposes a mixed-membership stochastic blockmodel (MMSB) that relaxes the assumption of nodes belonging to only a single role. The model is instantiated using a blockmodel and combines this with mixed-membership [60]. We also note that there are other various models such as the latent space models [61].

2.2.2 Row/Column Similarity of Adjacency Matrix

While the blockmodels are the most popular, there are also other techniques for computing graph-based roles that use some form of similarity between the rows of the adjacency matrix [62]. These graph-based similarity role methods have two general steps: First, the similarity (or distance) is computed between each pair of rows in the adjacency matrix. For this, any traditional similarity measure such as euclidean distance or correlation can be used. After computing the similarity matrix, the second step clusters the nodes using this similarity matrix. For the clustering, any traditional method can be used (e.g., hierarchical clustering and multi-dimensional scaling (MDS) [63] are the most common).

There are also spectral methods that compute the eigenvectors of the adjacency matrix (or a similarity matrix) then uses some subset of these to derive roles [64]. For computing roles (structural patterns), the interesting eigenvectors are not only those with the largest eigenvalues, but rather a subset of the eigenvectors that represent distinct structural patterns (stars-centers, star-edges, bridges, near-cliques, etc.). This arises due to the fact that role discovery methods attempt to capture or model all the significant structural patterns present in the data and are not restricted or focused on only a single type of pattern. As an aside, the eigenvectors associated to the largest eigenvalues represent one type of role (i.e., usually near-clique or tightly connected nodes).

Nevertheless, there has been some work focused on finding certain "types" of nodes in the graph that have a particular predefined role (i.e., structural pattern) [65], [66], [67]. For example, an early work by Kleinberg *et al.* [65] essentially computes the Singular Value Decomposition (SVD) [68] of a graph's adjacency matrix (i.e., eigenvectors of $\mathbf{A}\mathbf{A}^T$ and $\mathbf{A}^T\mathbf{A}$)[3], then identifies two types of star-center nodes based on incoming or

2. The number of roles k can be specified by the user or learned from the data.

3. $\mathbf{A}\mathbf{A}^T$ and $\mathbf{A}^T\mathbf{A}$ are similarity matrices.

outgoing edges known in that work as authority nodes and hub nodes [69]. These nodes were of interest as they relate to pages on the web that serve as a portal (hub) with many outgoing edges to reputable sites and pages that have many incoming edges (authorities).

2.2.3 Discussion

The main disadvantage of the models in Section 2.2.1 is that they are difficult to compute for large graphs and even more so for the massive graphs found in the real world such as Facebook's friendship graph, Twitter who-tweets-whom, among many others. Most of the previously mentioned work evaluates their model using rather small networks. For example, one recent model, dMMSB, which extends MMSB for dynamic networks, takes around a day to compute for 1,000 nodes, since their method is quadratic in the number of nodes.

This is in contrast to techniques based on similarity of the adjacency matrix (in Section 2.2.2), which if properly configured can be fast to compute [64] . The disadvantage of these methods is that the roles may be less meaningful or harder to interpret, whereas those based on blockmodels may be more accurate or easier to interpret from the implicit role definition implied by the model. Nevertheless, the utility of the roles ultimately depends on the application (what they are used for) and/or domain (where they are used).

2.3 Feature-based Roles

We first define feature-based roles and propose a taxonomy to illustrate the difference between graph-based roles. In Section 2.3.1 we discuss some possibilities of extending node equivalences for feature-based roles, then Section 2.3.3 discusses a few of the past approaches, and finally end with a discussion.

Intuitively, feature-based roles are derived by transforming the graph representation into a feature representation, then assigning roles based on some notion of feature equivalence. This is in contrast to the previous approaches in Section 2.2 that compute roles directly from the graph representation [32], [45], [46]. Roles computed from a feature representation are denoted as feature-based roles. The set of features typically arises based on some transformation(s) over the graph, $f(\mathcal{G}) = \mathbf{X}$ where $\mathcal{G}$ is a graph, $\mathbf{X}$ is the set of features, and $f(\cdot)$ is some collection of transformations over the graph. A transformation may be a set of aggregates or any other feature operator, see Table 4. More generally, the set of features may arise through some transformation(s) over an attributed-graph such that $f(\mathcal{G}, \mathbf{X}) = \tilde{\mathbf{X}}$ where the input features $\mathbf{X}$ and output $\tilde{\mathbf{X}}$ may consist of node features, link features, or non-relational features.

2.3.1 Node Equivalence for Feature-based Roles

The work on graph-based roles adopts some notion of node equivalence with respect to the the graph representation [4], [5], [26], [27], [70]. Since feature-based roles are computed from a different representation, we must move from the notions of node equivalence on the graph representation to node equivalence on a *feature representation*. We view node equivalence on a feature-based representation in two main ways. First, one may develop feature-based role methods that are consistent with one of the previous node equivalences (e.g., regular equivalence). Second, we may simply reinterpret/extend these equivalences for computing roles from a feature representation. Besides extending these definitions for features, one may also relax the node equivalences as done in graph-based roles. In this work, we primarily focus on the second view as this allows for greater flexibility, efficient methods, and may be more useful from a practical point of view.

Definition 2.1 (Node Equivalence on Feature Representation)**:** *Let $f_1, f_2, ..., f_m$ be a collection of structural features (degree, distance, ...), and let u and v be two arbitrary nodes, then a strict notion of node equivalence is defined as,*

$$(\forall i, 1 \leq i \leq m : f_i(u) = f_i(v)) \Rightarrow u \equiv v$$

This is strict in the sense that two nodes share the same role if they have identical feature-vectors. However, a common trend in the past has been to relax the notions of node equivalences while maintaining the properties of interest [26], [27], [32], [46]. For instance, most previous work is based on relaxations of structural equivalence, e.g., regular equivalence, stochastic equivalence, auto-

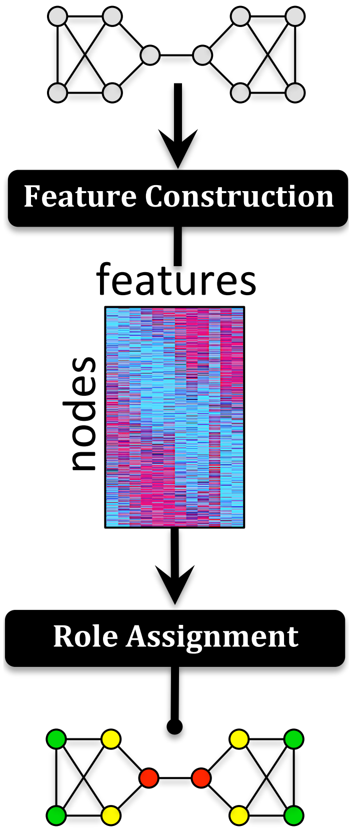


Fig. 4. **General Framework for Feature-based Roles.** The framework consists of *role feature construction* and *role assignment*.

morphic equivalence, among others.

2.3.2 Node Similarity for Feature-based Roles

As such, for feature-based roles, we can also move from the strict notion of node equivalence to the relaxed notion of node similarity (on a feature representation). Intuitively, two nodes u and v share the same role, if they have similar feature-values.

Definition 2.2 (Feature Similarity (Equivalence))**:** *Two nodes u and v are similar, if $\mathsf{S}(\mathbf{x}_u, \mathbf{x}_v) \approx 1$ where $\mathsf{S}(\cdot)$ is a standard similarity (or distance measure: $\mathsf{D}(\cdot) = 0$), and $\mathbf{x}_u$ and $\mathbf{x}_v$ are the feature-vectors of u and v, respectively. The similarity function $\mathsf{S}(\cdot) = 1$ if $\mathbf{x}_u$ and $\mathbf{x}_v$ are identical (s.t. 1 is the maximum similarity).*

Nevertheless, if the features are strictly graph features and representative of the structural properties in $\mathcal{G}$, then this implies u and v are structurally similar. We note that methods for computing a low-rank approximation or clustering (hierarchical) typically use some form of similarity (e.g., euclidean distance, Frobenius norm, ...). Notice that the above definition is independent of the neighbors of nodes and relies only on the features. This independence avoids roles being tied to each other based on cohesion (as is a problem of structural equivalence, see Section 2.1.1). Further, the definition is based on some notion of similarity, which is typically a relaxation of the stricter notion of structural equivalence which we can easily reinterpret for features. In terms of flexibility, these models are capable of expressing a larger class of roles than graph-based methods. This is a result of representing roles from a set of features constructed from a possibly infinite feature space.

2.3.3 Inspiration & Examples

Feature-based roles were perhaps inspired by indirect approaches to blockmodeling [56], [57], [58], latent feature models which are more expressive than traditional blockmodels [71], [72], [73], [74], [75], and other approaches based on the similarity of the rows of the adjacency matrix [62], [64]. We briefly review a few approaches for computing feature-based roles.

The first such method for computing a type of feature-based role was initially explored by Batagelj *et al.* [56]. In this seminal work, they formally define what it means for a collection of structural features (degree, distance, ...) to be consistent with structural equivalence [76]. This indirect approach measures the equivalence of nodes based on the feature-values. Let us note that Burt *et al.* [77] also proposed a dissimilarity measure consistent with structural equivalence. These indirect approaches were originally used in block modeling problems [56], [57], [58] and were usually designed to be consistent with one of the previous node equivalences from Section 2.1.

Some other work has recently focused on a more general type of feature-based role [18], [33], [36], which are not consistent with respect to the previous node equivalences defined by sociologists. In general, these approaches use some form of structural similarity for features. Intuitively, two nodes share the same role, if they have similar features. These approaches construct a large set of local features, that are specially tuned for social networks, then uses Non-negative Matrix Factorization (NMF) [78] to assign roles to the nodes. Some work has even used this particular specification of feature-based roles (i.e., degree/egonet features, NMF, Akaike's information criterion (AIC) [79]) to model and explore dynamic networks [80] and more recently to improve the accuracy of classification [36]. Let us note that this work only explores one type of feature-based role, while there exists many other opportunities that are perhaps more accurate, faster, or require tuning for specific applications. In Section 3, we propose a general framework for discovering feature-based roles, and discuss the issues, decisions required, and more generally the opportunities for using feature-based roles.

2.3.4 Discussion

Traditional graph-based approaches may not be flexible enough to capture complex roles in large networks due to their limitations in expressing such roles. For instance, McDaid [81] identifies two roles for the small karate club network, where the first role corresponds to high degree nodes and the second corresponds to small degree nodes. Alternatively, this article introduces a general framework for computing roles from a feature-based representation. We argue that feature-based roles provide greater flexibility for representing complex roles that are frequently seen in the real world and likely to be important for practical applications. On the other hand, the flexibility provided from a feature-based representation makes it more difficult to identify the features required to capture the roles warranted by the researcher. This part of the process takes some guidance by the expert or more tuning for a specific application.

2.4 Hybrid approaches

At the intersection of graph (Section 2.2) and feature-based roles (Section 2.3) lies hybrid approaches that leverage both the graph and feature representation (typically learned from a relational learning system [30], [31], [82], [83], [84]) in some fashion. We categorize hybrid role discovery methods into two main classes based on if a graph-based approach is used prior to role feature construction or whether the graph structure is leveraged after the learning of the feature representation for roles.

The first class of hybrid role discovery methods use a graph-based approach prior to role feature construction. For instance, we may use a blockmodel to extract roles directly from the graph, which can then be viewed as "initial attributes" for learning more sophisticated or targeted features. Once the initial attributes are learned, we can give them along with the graph as input to a relational feature learning system for which more meaningful features can be learned by incorporating the knowledge from the initial graph-based role discovery method. Using this refined set of features, we can now use a technique to automatically learn the number and assignment of the "hybrid-based" roles (See Section 5). This approach can be seen as a way to loosely constrain the feature-based roles to be similar to the specific parametric form assumed by the statistical blockmodel. The main disadvantage of these methods lies in their scalability due to the limitations of blockmodels (e.g., SBM [46], MMSB [32]), but remain a promising direction in the future as these methods become more scalable (See Section 6). Nevertheless, other more scalable graph-based approaches remain promising for this class of hybrid role discovery methods [65].

The second class of hybrid roles leverage multiple data sources as a way to regularize or influence the role assignment phase. It is assumed that a feature representation $\mathbf{X}$ from the graph was first learned, but there are additional data sources (i.e., graphs and attribute sets) that may be useful for role discovery. These hybrid roles can be learned by adapting tensor factorization methods [85], [86], [87], [88], [89] or collective matrix-tensor factorization (CMTF) methods [90], [91], [92]. Unlike tensor factorization methods, collective factorization methods can learn roles by fusing multiple heterogeneous data sources represented as matrices and/or tensors. For instance, we might have a store-categories matrix, a user-store-item tensor, and a social network matrix. In that context, one might posit that the roles of the users should be influenced by the store as well as the items that were purchased from that store and its category. Note these techniques may also consider time giving rise to dynamic roles (See Section 6.1). More generally, collective matrix-tensor factorization allows for any number of matrices or tensors to be included in the factorization which in turn allows these to directly influence the learned role definitions.

3 Framework for Feature-based Roles

This section introduces a flexible framework for *feature-based role discovery* which offers many benefits over traditional graph-based roles by computing roles from a feature representation (instead of directly from the graph, see Section 2). In particular, the framework consists of the following basic computations:

Role feature construction. Transform graph into a set of graph features

Role assignment. Assign nodes with similar feature vectors to same roles

The general framework, along with the two fundamental steps for discovering feature-based roles are intuitively illustrated in Figure 4. In that illustration, the graph is first transformed into a feature-based representation through some type of feature-construction technique or SRL system (section 4). Afterwards, roles are extracted from the large set of features via some type of low-rank approximation (matrix factorization) or clustering algorithm (section 5).

A detailed overview of the framework is shown in Table 2. Importantly, the feature learning systems used for *role feature construction* are interchangeable. For instance, one may use a simple feature learning system that searches over the space of local 1-hop neighborhood features or one may use an entirely different system that searches over more global features. The choice is entirely dependent on the important structural patterns present in the data and the overall application constraints such as scalability. Likewise, the methods for *role assignment* are also interchangeable as one might choose to use NMF instead of SVD for learning the roles from the set of learned features. This allows the framework to be useful and flexible for application-specific role discovery. Therefore, Table 2 also highlights the main categories of techniques for both feature construction and role assignment.

A key aspect of the feature-based roles is in their flexibility. For instance, unlike graph-based roles which use the graph directly, the feature-based roles (automatically) transform the graph (and any initial attributes) into a feature representation for which the role definitions are learned and extracted. Moreover, the proposed framework can also be used for the novel task of computing roles on the edges of a graph. We also note that application-specific constraints such as sparseness, diversity, locality, among many others may be placed on the feature-based role discovery problem, in either the feature construction or role assignment. The advantages of our general feature-based role framework are summarized below.

- Flexible framework for feature-based roles.
- Roles can be easily tuned for specific applications.
- Complexity and efficiency may be balanced depending on application-specific constraints.
- Able to capture arbitrary structural patterns (i.e., using data-driven and non-parametric approaches).
- Roles generalize since they are defined over features.
- Attributes are easily incorporated, since roles are naturally based on features they can simply be included as additional features before (or after) learning and constructing novel features automatically (see Section 4).

TABLE 2
Overview of the Feature-based Role Framework.

Feature-based Roles	**Role Feature Construction Section 4**	**Section 4.1) Relational Feature Classes**
		Section 4.2) Relational Feature Operators
		Section 4.3) Feature Search
		Section 4.4) Feature Selection
	Role Assignment Section 5	**Section 5.1) Feature Grouping/Clustering**
		Section 5.2) Low-rank Approximation
		Section 5.3) Model Selection

We also note that at the data-level, there are a number of factors that influence the learning and tuning of feature-based roles. Researchers and engineers may design a feature-based role method that is consistent in terms of the knowledge and a priori assumptions. Such a feature-based role method is likely to be much more effective for the specific application at hand. Assumptions and knowledge that may help guide the role discovery process include data and structure and/or future dependencies (e.g., sparsity, size, complexity, patterns), along with application/domain knowledge.

4 Role Feature Construction

In this section, we discuss the process for discovering features for roles. Relational feature construction is the systematic generation of features based on the graph structure or non-relational information. Roles have traditionally been defined strictly from the graph (e.g., graph-based roles from Section 2). In this work, we make only basic assumptions on the input used to discover roles. We assume that we are given a graph G which may have some initial node attributes denoted $\mathbf{X}^{v}$[4] or edge attributes $\mathbf{X}^{e}$. For example, a node attribute might be gender (m/f) and a link attribute might be the text of an email sent between two users. Throughout the process of feature construction, additional features may be added to set of features $\mathbf{X}$.

The goal of feature construction from its traditional view in machine learning is to construct features that are highly correlated with the prediction variable while being uncorrelated with each other. Unfortunately, the goal of feature construction for roles is not as straightforward. Previously, roles were defined mathematically based on structural equivalences [4], which are far too strict for practical purposes. Consequently, these notions were relaxed to get at the notion of structural similarity instead of equivalence [5], [27], [32], [70]. In this work, we use this same idea, but adopt it for computing roles from features, instead of the graph directly. Informally, the goal of a feature construction system should be to generate a set of features that capture the fundamental structures and important patterns in the graph data. We can further relax this definition by allowing the features to capture the interest to a specific domain (technological networks, social networks) or for a specific application (anomaly detection in computer networks). Since this definition depends intrinsically on the domain and application, we focus on surveying and discussing general ways to construct features for role discovery.

Intuitively, there are five main steps for learning a feature representation for role discovery:

– **Relational Feature Classes (Section 4.1).** Select the types of features to construct based on the graph data used in the computation.

TABLE 3
Taxonomy for Role Feature Construction. The proposed taxonomy for role feature construction is simple and intuitive, consisting of only the five main steps below. These steps can be viewed as explicit decisions that need to be learned automatically via the data and machine learning techniques or customized manually by a user (usually for a specific application/task). Note that this taxonomy expresses a very large family of algorithms for which a set of features can be computed for the role discovery problem.

Role Feature Construction Steps & Examples
Section 4.1) Relational Feature Classes
• Graph features (V, E) [33], [93]
• Relational link-value features $(\mathbf{X}^{e}, V, E)$ [94]
• Relational node-value features $(\mathbf{X}^{v}, V, E)$ [30]
• Non-relational features $(\mathbf{X})$ [35], [95]
Section 4.2) Relational Feature Operators
• Aggregates (mode, mean, count, ...) [18], [96]
• Set ops (union, intersection, multiset) [29], [97]
• Subgraph patterns (2-star, 3-star, triangle, etc.) [93], [98]
Section 4.3) Feature Search Strategy
• Exhaustive [30]
• Random [31]
• Guided [82], [83]
Section 4.4) Relational Feature Selection
• Correlation-based [99]
• Log-binning disagreement [100]

– **Relational Feature Operators (Section 4.2).** Determine the operators to use to construct those types of features.

– **Feature Search Strategy (Section 4.3).** Select a strategy for searching over the feature space including exhaustive, randomized, and guided search methods.

– **Relational Feature Selection (Section 4.4).** Determine how features are evaluated/scored and pruned (incrementally) during the learning process.

See Table 3 for an overview. We also present in Algorithm 1 a generic algorithm for discovering features.

4.1 Relational Feature Classes

We define the relational feature space with respect to the relational information used in the feature computation (i.e., edges/nodes only, graph+node/edge attributes, or non-relational information) giving rise to four main feature classes: structural features $(\mathcal{G})$, link-value features $(\mathcal{G}, \mathbf{X}^{e})$, node-value features $(\mathcal{G}, \mathbf{X}^{v})$, and non-relational features (which either uses $\mathbf{X}^{e}$ or $\mathbf{X}^{v}$ in the computation depending on if a non-relational link or node feature is being computed)[5]. The information in parenthesis represents the information used in the feature computation. It should be clear that both link features and node features can have features that are computed from each of the four classes of features. Now, let us define more precisely these four feature construction classes for the links and nodes.

4. $\mathbf{X}$ is used for $\mathbf{X}^{v}$ when meaning is clear

5. Clearly, these four classes of features can be recursively computed.

4.1.1 *Structural features*

Structural features are computed using only the structure of the graph $\mathcal{G}$ (and not features). Examples include traditional features like degree, clustering coefficient, betweenness, etc. In general, any type of subgraph pattern (e.g., number of 2-star patterns, triangles,...) or path/walk-based measures may be used to generate a large number of features. Let us note that these structural features can be computed for the links and nodes by a simple reinterpretation [101]. Roles as defined mathematically in the past [4], [26], [27] are based strictly on structural properties. Hence, these types of structural features are the most important for capturing the traditional notion of roles [5], [70].

4.1.2 *Link-value features*

Link-value features are computed using only the feature values of the links adjacent to the target node (or link)[6]. This process may easily be generalized to be the links ρ-hops away (i.e., paths of length ρ) away. Thus, the only information used in the computation is the graph $\mathcal{G}$ and the link features $\mathbf{X}^e$. As an example, given the feature-values of the adjacent links, one might apply some type of aggregate such as the mode to compute an additional feature. This feature class, like the others, can be used for computing node features or link features. In addition, another more difficult to see link-value (or node-value) features can be constructed via a low-dimensional approximation of $\mathbf{X}^e$ to generate additional features.

4.1.3 *Node-value features*

Similarly, node-value features are computed using only the feature values of the nodes that are adjacent (or a few hops away) from the target node. Thus, a feature is considered a relational node-value feature if the feature values of nodes linked to the target node are used in the construction. For instance, consider a political affiliation node attribute in Facebook, then given a node, we could construct a new node-value feature based on the mode of the values of the adjacent nodes. More generally, we could use the nodes k-hops away to compute a new node-value feature. As an aside, if $k > 1$, then one may decay the weight of that node on the feature calculation with respect to their distance such that nodes that are further away from the target node are given less influence in the feature calculation. Instead of the mode, one may choose to count the number of adjacent nodes of each political affiliation (e.g., number of liberal nodes k-hops away).

4.1.4 *Non-relational features*

A non-relational feature is defined as a feature computed using only the non-relational features (i.e., attributes), ignoring any link-based information [7]. Traditionally, the new feature value of that node is computed using the non-relational features of that node or the entire collection of nodes as done traditionally. The key idea is that the link-based information is ignored. Given a feature vector for an arbitrary node (or link), one might construct additional features by summing together that node's feature values, thresholding a single value, etc [97]. For instance, suppose there are two node features representing "avg time online" and "avg number of Facebook posts", then one may construct a new non-relational feature representing the sum of the two features. If there is textual data, then topic models [102] may also be utilized to construct new non-relational features, e.g., representing the node's main topic. The non-relational features can be used to better tune the role discovery process for a specific application.

6. The target node (or link) is the node for which the feature is being constructed.

7. Let us note that links and nodes may have non-relational features.

TABLE 4
Relational Feature Operators for Roles. Features may be constructed from relational graph data using any of the following relational operators below. As an example, subgraph pattern operators consist of both local graph features such as wedges or triangles and global features such as average shortest path between vertices.

Operators	Examples
Rel. aggr. [96], [103]	MODE, MEAN, COUNT, ...
Set ops [29]	Union, multiset, inters., ...
Subgraph pat. [98]	k-star, k-clique, k-motif, ...
Dim. redu. [104], [105]	SVD, PMF, NMF, ICA, PCA, ...
Similarity [106], [107]	Cosine sim, mutual info, ...
Paths/walks [108]	random-walks, k-walks, ...
Text analy. [109], [110]	LDA, Link-LDA/PLSA, ...

4.1.5 *Discussion*

The feature-based roles can naturally incorporate any of the above types of features. The only requirement is that for any set of link features, one must first use some type of feature operator (e.g., aggregate) to construct features on the node. We discuss opportunities for discovering link roles in Section 6, that is assigning roles to individual links, instead of nodes. However, any of the node features, including the non-relational features (i.e., attributes) can be used directly in the role computation by simply adding them to the node-feature matrix $\mathbf{X}$.

4.2 Relational Feature Operators for Roles

At the heart of feature construction are the actual feature operators that will be used in the underlying search process. These feature operators define the space of features that can potentially be searched over to construct a feature set, which in turn will ultimately be used to define the roles. The main classes of feature operators are categorized in Table 4, along with examples of each. Clearly, there is an infinite number of features that can be computed from these classes of feature operators.

Many of the feature operators can naturally be used to compute feature values for links and nodes. In addition, the majority of feature operators can compute features using most of the previous classes of inputs from Section 4.1. However, some of the operators that rely on non-relational information (i.e., text analysis), are obviously not applicable for constructing structural features, but can be applied for constructing link/node-value features and additional non-relational features (see Section 4.1). In addition, some of these relational operators can be applied recursively (e.g, aggregates, set ops, clustering, dimensionality reduction) and in an iterative fashion (to compute additional recursive features). For instance, we might compute the number of two-star patterns for a given node, then use the max relational aggregate operator, which would find the neighbor with the maximum number of two-star patterns and use this value as an additional feature-value. Let us note that Section 4.1 and 4.2 defines the overall feature space while Section 4.3 defines how this space is explored. See [101] for additional details.

4.2.1 Discussion

To construct meaningful, accurate and useful roles, it may be necessary to define a small subset of relational operators based on domain specific knowledge or assumptions. For instance, for social networks, operators that construct more local features (e.g., subgraph patterns) may be more useful whereas for technological networks other operators that construct global features may be more appropriate. On the other hand, if the evaluation criteria for searching this space of features is tuned for a specific application or domain, then the feature search strategy should select the appropriate features (and providing a small subset of operators is not necessary).

4.3 Feature Search Strategy

We previously defined the possible role feature space by specifying the raw feature inputs from Section 4.1 (e.g., structural features, node/link-value, and non-relational features) and the relational operators to consider (Section 4.2). The next step is to select an appropriate relational search strategy, which are usually either exhaustive, random, or guided search. An exhaustive strategy considers all features that are possible given the specified inputs and operators [30], while a random strategy will consider only a fraction of this space through some sampling strategy (See [31] for an example). A guided strategy uses a heuristic to identify the features that should be considered [82], [83]. In all three cases, each feature that is considered is subjected to some evaluation strategy that assesses its utility for representing roles (See Section 4.4 for more details). In Table 5, we provide a summary of relational learning systems that may be refined for feature-based roles. Since the majority of systems in Table 5 use guided search (instead of exhaustive or random), we list the name of the technique utilized. Note the guided feature learning also includes methods that learn weights incrementally for the feature subspaces. Usually the weights depend on the ability of learning novel and useful features from that subspace, e.g., all feature subspaces may be sampled uniformly in the first iteration, then biased in the subsequent iterations by the number of novel non-redundant features discovered. Hence, the role feature learning is guided by the weight (or bias) assigned to each feature subspace and updated after every subsequent iteration.

As mentioned previously, roles are largely domain and application dependent, and thus the evaluation strategy that assesses features should usually be appropriately tuned. Nevertheless, we can state some basic properties for constructing features that are suitable for computing generalized roles that match the previous definitions from Section 2. Any arbitrary feature set constructed from a feature search technique should be:

- ▷ **Representative.** The set of features should be representative of the main structural patterns in the graph (or those of interest to a specific domain or application).
- ▷ **Minimal.** The set of features should be minimal. There should not be redundant features.

However, as shown later in Section 5, the importance of these properties also depend on the technique used for assigning roles using the learned feature representation. For instance, some low-dimensional approximation techniques should automatically handle redundant features as they would simply be represented together in a much lower-dimensionality, while the other non-representative features would be regarded as “noise” and essentially removed. The most important issue is that the main structural properties/patterns of the graph are captured by the constructed graph features.

One may construct features either manually or automatically. Manual feature construction is essentially an exhaustive method that does not score or select features iteratively, and thus one needs to only select the types of features to construct (Section 4.1) and the feature operators (Section 4.2) to use over those types of features. In other words, the set of features are predefined based on expert knowledge of the problem, application and/or requirements. One key advantage of manually choosing features is that the resulting roles will be easier to interpret since they are based on a set of finely tuned features. Also, in cases where the problem/application is well-defined and there is plenty of domain knowledge, which is unlikely to change (thus not requiring strong generalization properties), then manual specification may result in stronger application-specific roles. The disadvantages of course are that an expert might miss a feature that is important to model or the known assumptions may change over time. But more importantly, the time and monetary costs to actually perform this tuning may make it impractical for most tasks.

Alternatively, we may construct features automatically in a non-parametric fashion using a system [30], [31], [82], [83], [84], which is the primary focus of this article. For instance, Algorithm 1 essentially searches a space of features automatically until no further novel features emerge. In that example, the feature space is defined by the set of primitive feature operators used initially (e.g., degree/egonet/k-core or other variants) and the set of recursive relational operators (e.g., sum, mode, etc.) selected. At each iteration, new features are constructed, redundant ones are discarded, and this process repeats until there are no more novel/useful features being generated. Other variants of the above are also possible. This automatic approach is more appropriate for large-scale analysis where roles must generalize across networks and/or roles may not be well understood (limited assumptions). The roles generated from the automatic feature construction are typically more difficult to interpret, but have the advantage of capturing arbitrary structural patterns. The ability to capture novel structural patterns may be important for applications such as anomaly detection. A more reasonable approach might be to have an expert manually tune a set of features and then use a system for automatically constructing additional features. This way, the roles are guaranteed to capture the properties warranted by the application (and expert) and also capture the main features of the graph data.

4.4 Relational Feature Selection: Scoring & Pruning

The previous section discussed strategies for searching over the space of relational features to generate a candidate set of features that capture the fundamental node-centric structural patterns (for assigning roles). Now we must decide how to evaluate the role-based features, which consists of (i) scoring the candidate features and then (ii) selecting/pruning them using these scores and any additional knowledge. The two fundamental goals of *unsupervised feature selection* are reduction and denoising. Reduction seeks to decorrelate and remove dependencies among features, whereas denoising attempts to find and reduce noisy features, therefore providing features that are more discriminative.

Let us start by defining the general notion of a similarity matrix[8]. Given a similarity function S and a set or matrix of features $\mathbf{X}$, we define a similarity matrix as $\mathbf{S} = \forall (i,j) \in F$, $\mathsf{S}(\mathbf{x}_i, \mathbf{x}_j)$. There are various similarity measures that can be used to evaluate role-based features such as Pearson correlation (and Spearman rank correlation) [111], information gain [112], gain ratio [113], gini-index [114], and Kearns-Mansour criteria [115], among others [101], [111], [114]. We also note that logarithmic binning is also useful for many sparse real-world networks with skewed node-level distributions (degree, number of triangles, etc) such as social and information

8. The similarity matrix may be viewed as a weighted similarity graph between the features.

TABLE 5
Systems for Searching and Selecting Features

Proposed System	Search method	Feature evaluation
RPT [30]	Exhaustive	χ^2 statistic/p-value
RDN-Boost. [122], [123]	Exhaustive	Weighted variance
ReFeX [124]	Exhaustive	Log-binning disagreem.
Spatiotemp. RPT [31]	Random	χ^2 statistic/p-value
SAYU [82]	Aleph	AUC-PR
nFOIL [83]	FOIL	Conditional LL
SAYU-VISTA [125]	Aleph	AUC-PR
Discri. MLN [126]	Aleph++	m-estimate
ProbFOIL [127]	FOIL	m-estimate
kFOIL [128]	FOIL	Kernel target-alignment
PRM stru. learn. [129]	Greedy hill-clim.	Bayesian model selection
TSDL [130]	Beam search	WPLL
BUSL [131]	Template-based	WPLL
PBN Learn & Join [132]	Level-wise search	Pseudo-likelihood

networks [100]. Other metrics that could be used include maximal information coefficient (MIC) [116], Mallows C_p [117], Bayesian information criterion (BIC) [118], [119] and many others [120], [121]. In addition, Table 5 summarizes the feature evaluation used in a variety of relational learning systems found in the literature.

Thus far, we have mostly presented unsupervised approaches that are based on the notion of a representative set of features that are non-redundant and minimal. However, the feature learning process may be guided by the underlying assumptions and knowledge about the application-specific objective of the roles being constructed. As an example, a candidate feature for roles may be evaluated by classification in an iterative fashion; if accuracy improves on a holdout set, then the feature may be added [82], [125]. In other cases, features may be scored at each successive iteration, and then only the feature with the largest score may be retained [131].

Given the similarity matrix $\mathbf{S}$ containing similarity scores/weights between all such pairs of features, how should we decide on the relational features to select? This problem may also be viewed as a relational role-based feature pruning problem. For feature-based roles, the feature selection method is driven by the goal of obtaining a representative set of features that is also minimal (e.g., redundant and noisy features are removed). This also has the additional benefit of improving the space-efficiency of the feature representation, which is important for large real-world networks. The pruning is generally performed automatically using a threshold (e.g., 0.5), which defines the level at which two features are labeled as similar. However, the pruning and similarity may be tuned for specific applications as well.

4.5 Generalized Role Feature Learning Template

Section 4.1 and 4.2 defined the space of relational features for roles whereas Section 4.3 and 4.4 determined how this space is explored and evaluated. The underlying decisions essentially determine the utility of the roles and therefore should be guided by knowledge and

Algorithm 1 Role Feature Learning Template

```
1  Input: $G = (V, E, \mathbf{X}^{\text{attr}})$ – Initial graph and attributes, $P$ – Set of
   primitive operators, $\Phi$ – Set of relational iterative operators, $\mathsf{S}(\cdot)$ –
   Score function, maxiter – Maximum number of iterations allowed,
   $\lambda$ – Threshold for searching
2  $F_0 \leftarrow$ PRIMITIVES($G$,$P$)
3  Let $\mathbf{X}$ be the feature data computed from the primitives
4  for $t \leftarrow 1$ to maxiter do
5      $F_t, \mathbf{X} \leftarrow$FEATURESEARCH($F_{t-1}, \mathbf{X}, \Phi$)
6      $F_t \leftarrow F_t \cup F_{t-1}$
7      $\mathcal{G}_F \leftarrow$CREATEFEATUREGRAPH($F_t, \mathbf{X}, \mathsf{S}, \lambda$)
8      $\mathcal{C} \leftarrow$ Partition the feature graph $\mathcal{G}_F$ (e.g., conn. components)
9      for each $\mathcal{C}_k \in |\mathcal{C}|$ do                         ▷ Prune features
10         Find the earliest (or min corr.) feature $f_i$ s.t. $\forall f_j \in \mathcal{C}_k : i < j$.
11         $F_t \leftarrow (F_t \setminus \mathcal{C}_k) \cup \{f_i\}$
12     Remove features from $\mathbf{X}$ that were pruned (not in $F_t$)
13     if $F_t = F_{t-1}$ then terminate search                 ▷ no new features
14 return $\mathbf{X}$ and $F_t$               ▷ feature matrix $\mathbf{X} \in \mathbb{R}^{n \times f}$ and list $F_t$

15 procedure CREATEFEATUREGRAPH($F_{t-1}, \mathbf{X}, \mathsf{S}$), $\lambda$)
16     Set $\mathcal{G}_F = (V_F, E_F)$ – the initial feature-graph
17     Set $V_F$ to be the set of features from $F$ and $E_F = \emptyset$
18     for each pair of features $(f_i, f_j) \in F_{t-1}$ do
19         if $\mathsf{S}(f_i, f_j) \geq \lambda$ then Add edge $(i, j)$ to $E_F$
```

assumptions known prior about the specific-application in which the learned roles will be used.

We present in Algorithm 1 a generalized (iterative) algorithm template for constructing a feature representation for role discovery from relational graph data. Given a large graph and attributes, how do we learn a feature representation that captures the relational dependencies between attributes in the graph and the structural properties and patterns present in the graph data? We propose an iterative graph feature discovery algorithm that learns a feature representation from a graph and any attributes. In particular, the learned features succinctly capture the main structural patterns/properties more compactly (non-redundant) while also revealing novel features.

The overall idea of Algorithm 1 is to start from the relational data $G = (V, E, \mathbf{X}^{\text{attr}})$ given as input, including graph data and attributes. Using a set of primitive relational operators, the algorithm constructs new features that are added to the initial feature set F_0 (Line 2). After the construction and pruning of the primitive features, the algorithm proceeds iteratively. At each iteration, new features are constructed using a set of relational iterative operators Φ (See Line 5). The previous set of features F_{t-1} are added to the new set of features F_t, thus $F_t \leftarrow F_t \cup F_{t-1}$ shown in Line 6. Now, we compute scores between all pairs of features and prune edges between features that are *not* significantly correlated:

$$E_F = \{(f_i, f_j) \mid \forall (f_i, f_j) \in F \times F \text{ s.t. } \mathsf{S}(f_i, f_j) > \lambda\}$$

This process results in a weighted feature graph where large edge weights represent dependencies between two features (Line 7). Now, we use the weighted feature graph $\mathcal{G}_F$ to prune all redundant and noisy features from F_t. This is done by first partitioning the feature graph (Line 8). For partitioning we use connected components, though there are numerous other possibilities (e.g., largest clique). Intuitively, each connected component represents a set of redundant features since edges represent dependencies. Therefore, for each component, one may prune all but the earliest (or least correlated) feature. Observe that the features learned at each iteration are guaranteed to be preserved in further subsequent iterations and therefore, $|F_1| \leq \cdots \leq |F_{t-1}| \leq |F_t|$. At the end of each iteration, we ensure that the feature matrix $\mathbf{X}$ reflects the set of features F_t (Line 12). Hence, pruned features are removed from $\mathbf{X}$. Finally, the iterative algorithm converges if no new features were constructed, hence $|F_t| = |F_{t-1}|$. Otherwise, the algorithm proceeds iteratively, repeating the previous steps. As an aside, Algorithm 1 is well-suited for directed/undirected graphs, which may be weighted, timestamped, multi-typed, and contain an arbitrary number of node and edge attributes. Many components of the algorithmic template are interchangeable and thus may be replaced/adapted for use with other techniques (e.g., backward feature selection, and others [114]). Furthermore Algorithm 1 is easily modified for extracting a list of previously learned features F on another input graph (useful for transfer learning, etc).

5 ROLE ASSIGNMENT

Section 4 proposed a general role-based feature learning framework for automatically transforming the the graph data into a representative set of features that capture the fundamental notion of roles. This section focuses on the second critical phase for discovering feature-based roles: how to assign nodes with similar feature vectors to the same role? In particular, we survey two main categories of methods for assigning roles using the graph-based feature representation: role clustering methods (Section 5.1) or low-rank approximation techniques (Section 5.2). Techniques to automatically select the best number of roles are discussed in Section 5.3. For each category of methods, there are also both soft role assignment and hard role assignment techniques (Section 5.4). We also note that some role assignment methods learn role definitions and thus generalize in the sense that roles learned on one network may be extracted on another arbitrary network (See Table 1 for potential applications). Other useful properties are discussed below whenever appropriate.

5.1 Role Clustering

There are many clustering algorithms that can be used to assign nodes to their corresponding roles using the graph features $\mathbf{X}$. The two primary types are hierarchical clustering algorithms (e.g., agglomerative or divisive clustering) [133] and partitioning algorithms such as k-means, k-medoids [134], [135], and self-organizing maps [136]. Many of the traditional clustering methods such as k-means are hard-clustering techniques. This is in contrast to soft-clustering methods which allow nodes to be in multiple clusters. A few classical methods are fuzzy C-means [137] or types of Gaussian Mixture Models [138], among others [60]

5.2 Low-rank Approximation

The other way to compute roles from a large feature matrix $\mathbf{X} \in \mathbb{R}^{n\times f}$ is to select a relatively small number r where $r \leq f$ (usually automatically, see 5.3) and compute a low rank-r matrix $\hat{\mathbf{X}}_r$ that best approximates the original feature matrix with respect to any standard matrix norm. There are many dimensionality reduction methods that can be used for this purpose. A few of the most common methods are SVD [68], Principal Component Analysis (PCA) [139], Kernel-PCA [140], MDS [63], spectral decomposition, NMF [78], [141], CUR [142], Factor Analysis, Probabilistic Matrix Factorization (PMF) [143], Independent Component Analysis (ICA) [144], [145], among many others. The majority of low-rank approximation techniques have a matrix $\mathbf{U}$ in SVD (and $\mathbf{W}$ in NMF) where each row represents the role-memberships for each node. While not all of the clustering algorithms allow nodes to be in multiple roles, nearly all the low-rank approximation techniques can be thought of as assigning nodes a role-membership. We discuss and provide two examples below.

Singular-value Decomposition (SVD). Let $\mathbf{X} \in \mathbb{R}^{n\times f}$ be the node by feature matrix, then we decompose $\mathbf{X}$ into three matrices using the SVD, $\mathbf{X} = \mathbf{USV}^T$ where $\mathbf{U} \in \mathbb{R}^{n\times f}$, $\mathbf{S} \in \mathbb{R}^{f\times f}$, $\mathbf{U} \in \mathbb{R}^{f\times f}$. The matrix $\mathbf{S}$ contains the singular values located in the $(i,i)_{1,..,f}$ cells in decreasing order of magnitude and all other cells contain zero. The eigenvectors of $\mathbf{XX}^T$ make up the columns of $\mathbf{U}$ and the eigenvectors of $\mathbf{X}^T\mathbf{X}$ make up the columns of $\mathbf{V}$. The matrices $\mathbf{U}$ and $\mathbf{V}$ are orthogonal, unitary and span vector spaces of dimension n and f, respectively. The columns of $\mathbf{U}$ are the principal directions of the features and the rows of $\mathbf{V}^T$ are the principal directions of the nodes. The principal directions are ordered according to the singular values and therefore according to the importance of their contribution to $\mathbf{X}$.

The SVD is used by setting the insignificant $f - r$ singular values to zero, which implies that we approximate the matrix $\mathbf{X}$ by a matrix $\mathbf{X}_r = \mathbf{U}_r\mathbf{S}_r\mathbf{V}_r^T$. A fundamental theorem by Eckart and Young [146] states that $\mathbf{X}_r$ is the closest rank-r least squares approximation of $\mathbf{X}$. This theorem allows us to set the insignificant singular values to zero and keep only the few influential singular values. The columns of $\mathbf{U}_r \in \mathbb{R}^{n\times r}$ represent the most significant roles while each row of $\mathbf{U}_k$ represents a node's membership in each of the roles. The singular values in $\mathbf{S}$ provide contribution scores for each of the roles in $\mathbf{U}_r$ and $\mathbf{V}_r^T$. The columns of V_r represent how roles contribute to the features. Essentially, the underlying roles were hidden in the large set of features, but when we reduce the dimensionality, the latent roles become apparent as similar graph features are combined. There are also many other methods similar to SVD that could be easier to interpret [142].

Non-negative Matrix Factorization (NMF). We also provide an example of computing roles using NMF. Given a node-feature matrix, we generate a rank-r approximation $\mathbf{WH} \approx \mathbf{X}$ where each row of $\mathbf{W} \in \mathbb{R}^{n\times r}$ represents a node's membership in each role and each column of $\mathbf{H} \in \mathbb{R}^{r\times f}$ represents how membership of a specific role contributes to estimated feature values. More formally, given a nonnegative matrix $\mathbf{X} \in \mathbb{R}^{n\times f}$ and a positive integer $r < \min(n, f)$, find nonnegative matrices $\mathbf{W} \in \mathbb{R}^{n\times r}$ and $\mathbf{H} \in \mathbb{R}^{r\times f}$ that minimizes the functional, $f(\mathbf{W},\mathbf{H}) = \frac{1}{2}||\mathbf{X} - \mathbf{WH}||^2$ Let us note that NMF is often "easier" to interpret than SVD, due to the non-negativity constraint. As an aside, role clustering methods and low-rank approximation methods may sometimes overlap, e.g., there has been a lot of work showing the equivalence between NMF and k-means clustering [147], [148].

Discussion: Depending upon the application, expected data characteristics, and computational requirements (i.e., memory, runtime, or scalability constraints), the role-based low-rank approximation methods in the framework are flexible for the following: (i) similarity/objective function (e.g., Frobenius norm, KL-divergence), (ii) regularization terms if warranted (e.g., sparsity constraints, L2, etc), and (iii) solver (e.g., Multiplicative update, SGD, ALS). For instance, suppose roles are assigned via an NMF-based approach using KL-divergence with L2 regularization terms and a CCD-based solver. One may also add sparsity and other constraints [149], [150], [151] to these approaches to better adapt the roles for specific applications [152]. Furthermore depending on the input data and application, many techniques exist for improving the semantics of roles and their interpretation (e.g., pre/postprocessing techniques may help avoid/interpret negative coordinates). We also note that many of these techniques may also be used for learning roles over a time series of graphs, see [18]. For dynamic networks, one may also represent the time series of graphs as a tensor where roles can now be learned using a tensor factorization method [85], [87], [88], [89], [153].

5.3 Model Selection: Choosing the number of roles

Many techniques have been proposed for selecting the appropriate number of roles. Some of these techniques are heuristic while others have a more fundamental basis in statistics (e.g., AIC [79]) and information theory, e.g., Minimum Description Length (MDL) [154], [155], [156]. In this section, we survey a few of these approaches and discuss details of each and how they could be used in the context of role discovery.

Selecting number of clusters. There has been a substantial amount of research on selecting the number of clusters automatically [157], [158], [159], [160], [161]. The classical clustering methods such as hierarchical clustering [162]

Algorithm 2 Role Model Selection

```
1  Set mincost = $\infty$, failed = 0, trials = $\tau$
2  Set $\mathbf{W}_0$ and $\mathbf{H}_0$ to be random matrices (normal dist.)
3  Scale $\mathbf{W}_0$ and $\mathbf{H}_0$ by max value in $\mathbf{X}$
4  for $r = 1$ to $\min(n, f)$ do
5      Learn model $D(\mathbf{X}|\mathbf{W}_0[:, r], \mathbf{H}_0[r, :])$
6      Compute cost of model via criterion (e.g., MDL, AIC)
7      if cost < mincost then                ▷ model improves likelihood
8          Set mincost = cost, reset failed to 0, and save model
9      else   set failed = failed + 1
10     if failed $\geq$ trials then
11         terminate search and return model
```

and k-means [158] have been adapted to select the number of roles automatically (using various criteria).

Selecting the number of dimensions for low-rank approximation. Many methods have been developed for this purpose [163], [164], [165], [166]. For example, using SVD or other related techniques, it is common in information retrieval to select 50 or 100 eigenvectors as their importance typically follows a geometric distribution (in most types of data) [167]. In general, the number of significant eigenvalues from the feature matrix will be much less for feature-based roles. Empirically, the number of roles found in practice is usually quite small, between 2 and 15 roles are usually found for a variety of network types [80].

We provide a general role model selection template in Algorithm 2 for automatically selecting the number of roles for low-rank approximation techniques. Intuitively, Algorithm 2 greedily increases the number of roles as long as the cost of the role model decreases (or likelihood improves using that number of roles). Note that $\mathbf{W}_0$ and $\mathbf{H}_0$ are the initial randomized matrices whereas $\mathbf{W}_0[:, r]$ and $\mathbf{H}_0[r, :]$ in line 5 are the first r columns and rows from the initial $\mathbf{W}_0$ and $\mathbf{H}_0$, respectively. One may also use MDL to automatically determine the number of structural roles. Intuitively, learning more roles increases model complexity but decreases the error (and vice versa). In this way, MDL selects the number of roles r (representing structural patterns) such that the model complexity (number of bits) and model errors are balanced. Note the number of bits is typically $\log_2(\text{number of parameters})$. Naturally, the best model minimizes, $number\ of\ bits + errors$. Note that $\tau = 5$ is a heuristic used in some optimization schemes to represent a finite number of forward greedy steps [168]. This approach provides statistical guarantees for the estimated model.

Importantly, Algorithm 2 may also serve as a basis for a more guided or supervised approach to selecting the number of roles (i.e., driven by an application-specific objective function). For instance, if we were interested in classification, then one may learn the best model that maximizes the cross-validated AUC.

5.4 Hard and Soft Role Assignments

Hard role assignment refers to a vertex (or edge) being assigned to only a single role [8], [10], [46], whereas soft role assignment allows for vertices to play multiple roles (mixed-membership) [28], [32], [33], [49]. See Figure 5 for an illustration. Depending on the situation and constraints, one may make more sense than the other. For instance, soft role assignment is typically more expensive in terms of storage requirements. For a dynamic graph where each snapshot graph G_t represents 1 minute (timescale), then it may be impractical to allow a vertex to play multiple roles. As an example, if the vertices represent individuals, then it is unlikely that any individual would be playing more than a single role, since the time scale is so small. However, if the snapshot graph G_t represents 1 day of activity, then individuals are likely to play multiple roles. As an example, an individual is likely to spend a good portion of the day in the "work role", then after work they may transition into the wife or mother role.

- **Soft Role Assignment Methods**. Matrix factorization techniques allow for soft role assignments. These include the typical methods of SVD, NMF, PMF, or any other factorization. Gaussian mixture models also assign soft roles to vertices (or edges).
- **Hard Role Assignment Methods**. The classical k-means and the variants are hard role assignment methods as these typically assign a data point to a single centroid (role).

6 Discussion and Challenges

This section discusses additional issues in role discovery and highlights important directions and challenges for future work. A few of the tasks for which roles may be helpful are summarized in Table 1.

6.1 Dynamic and streaming graphs

Most role discovery methods introduced in this article are for static networks. However, networks are not static, but are naturally dynamic and continuously streaming over time and thus role discovery methods for dynamic and streaming networks are of fundamental practical and theoretical importance. Modeling "dynamic" roles are important for many practical applications (e.g., recommendation) due to the fact that user preferences and behavior are fundamentally tied to time. Intuitively, methods for learning roles in dynamic networks must leverage the notion that recent user preferences are more predictive than preferences in the distant past.

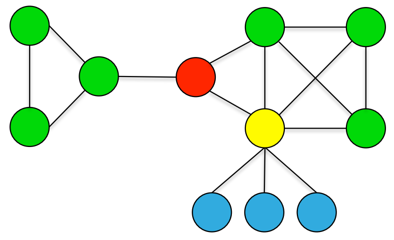
There have only been a few approaches for dynamic feature-based roles. More specifically, Role-Dynamics learns features from the time series of graph data, then assigns roles using those features. Using the learned feature and role definitions, they now extract feature-based roles in a streaming fashion for detecting graph-based anomalies [33]. This approach uses simple local primitives (i.e., degree/egonet-based features) with $\{mode, sum\}$ operators and used NMF+MDL for assigning roles. More recently, DBMM extended that approach by modeling the global network role transitions as well as learning local role transition models for each node [18]. There remains many challenges and opportunities for dynamic feature-based roles. For instance, one may instead represent the sequence of node-feature matrices $\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_t$ as a tensor and use tensor factorization techniques to learn the role definitions [85], [87], [88], [89], [153]. Moreover, we may also utilize non-negative tensor factorization (NTF) techniques [86], [89] for grouping the features and thus providing a time-series of role mixed-memberships. There are also regularized non-negative tensor factorization approaches [169] and constraint-based approaches (including sparsity and other constraints) [149], [150], [151] that can be used to better adapt the roles for specific applications.

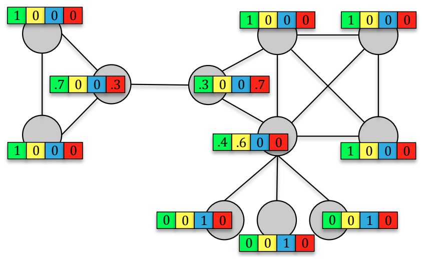
In terms of graph-based roles, Fu *et al.* proposed an approach based on MMSB for dynamic networks called dMMSB [28], [49] that essentially characterizes how the roles in MMSB change over time. However, dMMSB does not scale to large networks found in the real-world [28] (See Section 6.3). The other disadvantage is in the assumption of a specific parametric form, whereas feature-based roles are more flexible in the types of roles expressed and that roles are data-driven/non-parametric and thus can easily adapt based on stream/dynamic characteristics.

Online role discovery methods that assign and update role memberships as new data arrives remains an open area of research [170]. These methods are not only required to update role memberships, but must incrementally update the underlying role model and definitions. In feature-based roles, the set of features may also become stale and new representative features may need to be learned in an incremental fashion. Additional work may investigate techniques to automatically learn the appropriate time scale, lag of the time series to use, and parameters for exponential smoothing.

6.2 Evaluation and Guidance

Roles have traditionally been used as a descriptive modeling tool to understand the "roles" played by actors in the network and has been limited to relatively small networks. This makes it difficult to understand and evaluate the benefits of the various methods. The previous work discovers roles, then uses them for applications (e.g., link prediction, anomaly detection). However, there has yet to be any work that "guides" the roles *during learning* by the end goal or application. Ideally, the learned roles should be guided by the final goal and should adapt based on the application-specific goal. For instance, one might adapt an approach that discovers roles that maximize classification accuracy on a hold-out data set. In that case, classification accuracy is used as a surrogate evaluation measure that is carried out during the role discovery process (e.g., a role and its features are retained if classification accuracy increases). These types of approaches are not only important for learning more goal-oriented roles, but may serve as a fundamental basis for evaluating role discovery methods.


(a) Hard Assignment



(b) Soft Assignment

Fig. 5. **Example of Hard and Soft Role Assignments**. Fig. 5(a) demonstrates hard role assignments to the nodes. In particular, nodes are assigned the roles of clique, bridge, star-center, and peripheral role. In Fig 5(b), each node is assigned a distribution of roles s.t. $\sum r_i = 1$. In some cases, this may be beneficial. For example, the vertex in Fig. 5(a) that is assigned the bridge role also participates in a clique of size three. In another example, the vertex in Fig. 5(a) that is assigned the star-center role is also participating in a clique of size four. In both these cases, a hard assignment may not make sense, instead a distribution of roles should be assigned to accurately represent the node roles.

There has yet to be any attempt to develop an evaluation framework with the aim at understanding the algorithmic tradeoffs including accuracy (e.g., approx. error, evaluation measure, interpretation, etc) and efficiency. Accuracy and efficiency need to be carefully balanced by the user depending on the restrictions/knowledge of the domain and the end goal (application) of role discovery.

6.3 Scaling up role discovery methods

The majority of traditional graph-based role methods were only suitable for relatively small networks [28], [49]. Recently, there has been work on scaling up generalized block models [171] as well as stochastic block-model variants using downsampling and a fast stochastic natural gradient ascent algorithm for variational inference [172], [173]. Though, there still remains a lot of work to be done in this area, for instance, scaling up the methods further for graphs of billions of edges, adapting other methods using these fast inference procedures, and evaluating and comparing the accuracy of the approximation methods and their scalability in terms of parallel speedup.

Despite the fact that feature-based role methods are in general much more efficient and scalable than graph-based roles, there has yet to be any parallel feature-based role methods. Though, a systematic investigation into these methods and relative parallel speedups would be extremely useful. We note that role features in Algorithm 1 may be computed independently for each node in parallel while role definitions may also be learned in parallel [174], [175]. Downsampling (or network sampling) is another promising direction for feature-based role methods [39], for instance, feature definitions may be learned on a much smaller sampled network, then roles may be assigned to the sampled nodes based on this feature representation.

Additionally, methods for learning a set of space-efficient features remains a challenge. In some cases, the features learned in past work required more memory to store than the sparse graph itself and thus impractical for massive graphs. A promising direction would be to design methods for learning sparse features or better pruning methods to reduce this cost. Further, there has yet to be any methods for learning (graph or feature-based) roles in a distributed setting (using MPI/MapReduce) as well as role discovery methods suitable for GPU parallelization (i.e., massively multicore architectures with limited memory), yet both are critical for practical settings in the age of big data [176].

6.4 Additional Directions

Recently, there has been some work on incorporating graph and textual data into roles [177], [178]. Additional work is needed to systematically understand the impact and utility of such approaches. Similarly, no one has included the latent topics as attributes in a relational feature learning system to generate additional novel features to improve the learned roles.

Incorporating initial attributes into the feature learning system (instead of only the graph) for learning roles is another promising area of research. It is widely known that feature engineering is the most important factor contributing to the success of machine learning algorithms in practice [179]. Naturally, the same is true for feature-based roles and therefore using the attributes (or external interest) as a means to guide the feature learning system provides a unique opportunity to mesh the graph structure with external attributes allowing for a more guided approach to role learning.

There has not been any work on role discovery methods for heterogeneous networks. Nevertheless, many of the methods introduced in this article may be adapted for multiple node or edge types, though there has yet to be any investigation or evaluation of such methods for practical applications.

Feature learning systems for graph data depend heavily on the "correctness" of the observed links [101]. In many cases, the links in the graph may be difficult to observe, noisy, and may be ambiguous to the end goal of role discovery [180]. Thus, preprocessing the graph data (e.g., weighting the links) may substantially improve features learning via the role-based feature learning systems. In addition, previous work has regarded all relational feature operators uniformly, though in practice some operators may be more useful than others. Weighting the features via the relational operators that constructed them is likely to be useful for fine-tuning role discovery for certain applications and has yet to be explored.

Feature-based roles also has connections to the increasing important problem of deep learning [181], [182]. Though most work has focused on image data, we may view the feature-based roles as a generalized graph-based deep learning approach. In particular, this article introduced a framework for learning a two-layered representation based on the graph structure. Recently, additional layers were learned on top of the feature-based roles to model the "role dynamics" [18]. In particular, they learned a global layer representing how the roles change over time as well as local multi-level layers for each node representing how the roles of a node change over time.

Semi-supervised role discovery methods is another promising direction that has yet to be addressed. For instance, one may compute strict properties of the nodes in the graph and use these to label examples of roles in the graph for which other roles can be learned and extracted. These could now be used in a semi-supervised fashion to help moderate the role discovery algorithm leading to roles that are more interpretable and useful.

Finally, the problem of computing link roles has yet to be addressed. However, many of the previous techniques for feature-based roles may be adapted for computing link-based roles.

Acknowledgments

The authors would like to thank the reviewers for helpful comments. This work is supported by the NSF GRFP Fellowship under Grant No. DGE-1333468.

References

[1] T. Parsons, "Illness and the role of the physician: A sociological perspective*," *Amer. J. of Ortho.*, vol. 21, no. 3, pp. 452–460, 1951.
[2] R. Merton, *Social theory and social structure*. Simon & Sch., 1968.
[3] S. Borgatti, M. Everett, and J. Johnson, *Analyzing Social Networks*. Sage Publications, 2013.
[4] F. Lorrain and H. White, "Structural equivalence of individuals in social networks," *J. Math. Soc.*, vol. 1, no. 1, pp. 49–80, 1971.
[5] P. HollandKathryn Blackmond and S. Leinhardt, "Stochastic blockmodels: First steps," *Soc. networks*, vol. 5, pp. 109–137, 1983.
[6] P. Arabie, S. Boorman, and P. Levitt, "Constructing blockmodels: How and why," *J. Math. Psych.*, vol. 17, no. 1, pp. 21–63, 1978.
[7] C. Anderson, S. Wasserman, and K. Faust, "Building stochastic blockmodels," *Social Networks*, vol. 14, no. 1, pp. 137–161, 1992.
[8] V. Batagelj, A. Mrvar, A. Ferligoj, and P. Doreian, "Generalized blockmodeling with pajek," *Metodoloski zvezki*, pp. 455–467, 2004.
[9] P. Doreian, V. Batagelj, and A. Ferligoj, *Generalized blockmodeling*. Cambridge Univ Pr, 2005, vol. 25.
[10] K. Nowicki and T. Snijders, "Estimation and prediction for stochastic blockstructures," *J. Amer. Stat. Assoc.*, vol. 96, no. 455, pp. 1077–1087, 2001.
[11] J. Scripps, P. Tan, and A. Esfahanian, "Node roles and community structure in networks," in *WebKDD*, 2007, pp. 26–35.
[12] P. Mahadevan *et al.*, "The Internet AS-level Topology: Three Data Sources and One Definitive Metric," *SIGCOMM*, vol. 36, no. 1, pp. 17–26, 2006.
[13] R. A. Rossi, S. Fahmy, and N. Talukder, "A multi-level approach for evaluating internet topology generators," in *IFIP Networking*, 2013, pp. 1–9.

[14] A. Varki, "Biological roles of oligosaccharides: all of the theories are correct," *Glycobiology*, vol. 3, no. 2, pp. 97–130, 1993.

[15] J. Luczkovich, S. Borgatti, J. Johnson, and M. Everett, "Defining and measuring trophic role similarity in food webs using regular equivalence," *J. Theo. Bio.*, vol. 220, no. 3, pp. 303–321, 2003.

[16] H. Ma, I. King, and M. R. Lyu, "Mining web graphs for recommendations," *TKDE*, vol. 24, no. 6, pp. 1051–1064, 2012.

[17] S. Golder and J. Donath, "Social roles in electronic communities," *Internet Research*, vol. 5, pp. 19–22, 2004.

[18] R. A. Rossi, B. Gallagher, J. Neville, and K. Henderson, "Modeling dynamic behavior in large evolving graphs," in *WSDM*, 2013, pp. 667–676.

[19] A. Farahat, N. K. Ahmed, and U. Dholakia, "Does a daily deal promotion signal a distressed business? an empirical investigation of small business survival," in *EWSSN*, 2013, pp. 1–8.

[20] A. Clauset, M. E. Newman, and C. Moore, "Finding community structure in very large networks," *Phy. rev. E*, vol. 70, no. 6, 2004.

[21] J. Chen and Y. Saad, "Dense subgraph extraction with application to community detection," *TKDE*, vol. 24, no. 7, pp. 1216–1230, 2012.

[22] L. Backstrom, D. Huttenlocher, J. Kleinberg, and X. Lan, "Group formation in large social networks: membership, growth, and evolution," in *KDD*, 2006.

[23] D. Chakrabarti, R. Kumar, and A. Tomkins, "Evolutionary clustering," in *KDD*, 2006.

[24] B. Yang, J. Liu, and J. Feng, "On the spectral characterization and scalable mining of network communities," *TKDE*, vol. 24, no. 2, pp. 326–337, 2012.

[25] M. Newman, "Fast algorithm for detecting community structure in networks," *Physical Review E*, vol. 69, no. 6, p. 066133, 2004.

[26] D. White and K. Reitz, "Graph and semigroup homomorphisms on networks of relations," *Social Networks*, vol. 5, no. 2, pp. 193–234, 1983.

[27] P. Holland and S. Leinhardt, "An exponential family of probability distributions for directed graphs," *J. Amer. Stat. Assoc.*, pp. 33–50, 1981.

[28] E. Xing, W. Fu, and L. Song, "A state-space mixed membership blockmodel for dynamic network tomography," *Ann. Appl. Stat.*, vol. 4, no. 2, pp. 535–566, 2010.

[29] L. McDowell, K. Gupta, and D. Aha, "Cautious collective classification," *JMLR*, vol. 10, pp. 2777–2836, 2009.

[30] J. Neville, D. Jensen, L. Friedland, and M. Hay, "Learning relational probability trees," in *SIGKDD*, 2003, pp. 625–630.

[31] A. McGovern, N. Collier, I. Matthew Gagne, D. Brown, and A. Rodger, "Spatiotemporal Relational Probability Trees: An Introduction," in *ICDM*, 2008, pp. 935–940.

[32] E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing, "Mixed membership stochastic blockmodels," *JMLR*, vol. 9, pp. 1981–2014, 2008.

[33] R. Rossi, B. Gallagher, J. Neville, and K. Henderson, "Role-dynamics: Fast mining of large dynamic networks," in *WWW LSNA*, 2012, pp. 997–1006.

[34] I. Bhattacharya and L. Getoor, "Collective entity resolution in relational data," *TKDD*, vol. 1, no. 1, pp. 1–36, 2007.

[35] S. J. Pan and Q. Yang, "A survey on transfer learning," *TKDE*, vol. 22, no. 10, p. 1345, 2010.

[36] K. Henderson, B. Gallagher, T. Eliassi-Rad, H. Tong, S. Basu, L. Akoglu, D. Koutra, and L. Li, "Rolx: Structural role extraction & mining in large graphs," in *SIGKDD*, 2012, pp. 1231–1239.

[37] M. Bilgic, L. Mihalkova, and L. Getoor, "Active learning for networked data," *ICML*, 2010.

[38] T. Tassa and D. J. Cohen, "Anonymization of centralized and distributed social networks by sequential clustering," *TKDE*, vol. 25, no. 2, pp. 311–324, 2013.

[39] N. K. Ahmed, J. Neville, and R. Kompella, "Network sampling: From static to streaming graphs," *TKDD*, pp. 1–45, 2013.

[40] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *TEVC*, vol. 1, no. 1, pp. 67–82, 1997.

[41] D. H. Wolpert, "The lack of a priori distinctions between learning algorithms," *Neural Comp.*, vol. 8, no. 7, pp. 1341–1390, 1996.

[42] ——, "The supervised learning no-free-lunch theorems," in *Soft Computing and Industry*. Springer London, 2002, pp. 25–42.

[43] H. Xu, C. Caramanis, and S. Mannor, "Sparse algorithms are not stable: A no-free-lunch theorem," *TPAMI*, vol. 34, no. 1, pp. 187–193, 2012.

[44] C. Goutte, "Note on free lunches and cross-validation," *Neural Computation*, vol. 9, no. 6, pp. 1245–1249, 1997.

[45] A. Goldenberg, A. Zheng, and S. Fienberg, *A survey of statistical network models*. Now Publishers, 2010.

[46] K. Nowicki and T. Snijders, "Estimation and prediction for stochastic blockstructures," *J. Amer. Stat. Assoc.*, vol. 96, pp. 1077–1087, 2001.

[47] M. Richardson and P. Domingos, "Markov logic networks," *Mach. Learn.*, vol. 62, no. 1-2, pp. 107–136, 2006.

[48] S. Riedel and I. Meza-Ruiz, "Collective semantic role labelling with markov logic," in *CoNLL*, 2008, pp. 193–197.

[49] W. Fu, L. Song, and E. Xing, "Dynamic mixed membership blockmodel for evolving networks," in *ICML*, 2009, pp. 329–336.

[50] M. Everett, "Role similarity and complexity in social networks," *Social Networks*, vol. 7, no. 4, pp. 353–359, 1985.

[51] L. Sailer, "Structural equivalence: Meaning and definition, computation and application," *Soc. Net.*, vol. 1, no. 1, pp. 73–90, 1979.

[52] M. Everett, J. Boyd, and S. Borgatti, "Ego-centered and local roles: A graph theoretic approach," *J. Math. Soc.*, vol. 15, no. 3-4, pp. 163–172, 1990.

[53] J. Boyd and M. Everett, "Relations, residuals, regular interiors, and relative regular equivalence," *Social Networks*, vol. 21, no. 2, pp. 147–165, 1999.

[54] P. Doreian, V. Batagelj, and A. Ferligoj, "Generalized blockmodeling of two-mode network data," *Social Networks*, vol. 26, no. 1, pp. 29–53, 2004.

[55] R. Breiger, S. Boorman, and P. Arabie, "An algorithm for clustering relational data with applications to social network analysis and comparison with multidimensional scaling," *Journal of Mathematical Psychology*, vol. 12, no. 3, pp. 328–383, 1975.

[56] V. Batagelj, A. Ferligoj, and P. Doreian, "Direct and indirect methods for structural equivalence," *Social networks*, vol. 14, no. 1-2, pp. 63–90, 1992.

[57] P. Doreian, V. Batagelj, and A. Ferligoj, "Generalized blockmodeling of two-mode network data," *Social networks*, vol. 26, no. 1, pp. 29–53, 2004.

[58] V. Batagelj, A. Ferligoj, and P. Doreian, "Indirect blockmodeling of 3-way networks," *Selected Contributions in Data Analysis and Classification*, pp. 151–159, 2007.

[59] P. Lazarsfeld and N. Henry, *Latent structure analysis*. Houghton, Mifflin, 1968.

[60] E. Erosheva and S. Fienberg, "Bayesian mixed membership models for soft clustering and classification," *Classificationthe ubiquitous challenge*, pp. 11–26, 2005.

[61] P. Hoff, A. Raftery, and M. Handcock, "Latent space approaches to social network analysis," *J. Amer. Stat. Assoc.*, vol. 97, no. 460, pp. 1090–1098, 2002.

[62] R. Burt, "Positions in networks," *Social forces*, vol. 55, no. 1, pp. 93–122, 1976.

[63] J. B. Kruskal, "Nonmetric multidimensional scaling: a numerical method," *Psychometrika*, vol. 29, no. 2, pp. 115–129, 1964.

[64] U. Brandes and J. Lerner, "Structural similarity: spectral methods for relaxed blockmodeling," *J. Class.*, vol. 27, pp. 279–306, 2010.

[65] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," *JACM*, vol. 46, no. 5, pp. 604–632, 1999.

[66] H. Tong, S. Papadimitriou, C. Faloutsos, S. Y. Philip, and T. Eliassi-Rad, "Gateway finder in large graphs: problem definitions and fast solutions," *Information Retrieval*, vol. 15, no. 3-4, pp. 391–411, 2012.

[67] D. Jiang and J. Pei, "Mining frequent cross-graph quasi-cliques," *TKDD*, vol. 2, no. 4, p. 16, 2009.

[68] G. Golub and C. Reinsch, "Singular value decomposition and least squares solutions," *Numerische Mathematik*, vol. 14, no. 5, pp. 403–420, 1970.

[69] F. Chung, *Spectral graph theory*. Amer. Math. Soc., 1997, no. 92.

[70] M. Everett and S. Borgatti, "Regular equivalence: General theory," *J. Math. Soc.*, vol. 19, no. 1, pp. 29–52, 1994.

[71] K. Miller, T. Griffiths, and M. Jordan, "Nonparametric latent feature models for link prediction," *NIPS*, pp. 1276–1284, 2009.

[72] T. L. Griffiths and Z. Ghahramani, "Infinite latent feature models and the indian buffet process," in *NIPS*, 2005, pp. 475–482.

[73] D. Navarro and T. Griffiths, "Latent features in similarity judgments: A nonparametric bayesian approach," *Neural computation*, vol. 20, no. 11, pp. 2597–2628, 2008.

[74] Z. Ghahramani, T. Griffiths, and P. Sollich, "Bayesian nonparametric latent feature models," *Bayesian Stat.*, pp. 201–225, 2007.

[75] F. Doshi-Velez and Z. Ghahramani, "Correlated non-parametric latent feature models," in *UAI*. AUAI Press, 2009, pp. 143–150.

[76] V. Batagelj, "Similarity measures between structured objects," *Studies in physical and theoretical chemistry*, vol. 63, pp. 25–39, 1989.

[77] R. Burt, M. Minor *et al.*, *Applied network analysis: A methodological introduction*. Sage Publications Beverly Hills, CA, 1983.

[78] D. Lee, H. Seung *et al.*, "Learning the parts of objects by non-negative matrix factorization," *Nat.*, vol. 401, pp. 788–791, 1999.

[79] H. Akaike, "A new look at the statistical model identification," *TAC*, vol. 19, no. 6, pp. 716–723, 1974.

[80] R. Rossi, B. Gallagher, J. Neville, and K. Henderson, "Modeling temporal behavior in large networks: A dynamic mixed-membership model," in *LLNL-TR-514271*, 2011.

[81] A. McDaid, B. Murphy, N. Friel, and N. Hurley, "Model-based clustering in networks with stochastic community finding," *Arxiv preprint arXiv:1205.1997*, 2012.

[82] J. Davis, E. Burnside, I. Castro Dutra, D. Page, and V. Costa, "An integrated approach to learning Bayesian networks of rules," in *ECML*, 2005, pp. 84–95.

[83] N. Landwehr, K. Kersting, and L. De Raedt, "nFOIL: Integrating naıve bayes and FOIL," in *AAAI*, 2005, pp. 275–282.

[84] R. Rossi and J. Neville, "Time-evolving relational classification and ensemble methods," in *PAKDD*. Springer, 2012, pp. 1–13.

[85] L. De Lathauwer, "A survey of tensor methods," in *ISCAS*, 2009, pp. 2773–2776.

[86] M. P. Friedlander and K. Hatz, "Computing non-negative tensor factorizations," *Opt. Meth. & Soft.*, vol. 23, pp. 631–647, 2008.

[87] Y. K. Yılmaz and A. T. Cemgil, "Probabilistic latent tensor factorization," in *Latent Variable Analysis and Signal Separation*, 2010, pp. 346–353.

[88] A. Cichocki, M. Mørup, P. Smaragdis, W. Wang, and R. Zdunek, "Advances in nonnegative matrix and tensor factorization," *Computational intelligence and neuroscience*, 2008.

[89] Y.-D. Kim and S. Choi, "Nonnegative tucker decomposition," in *CVPR*, 2007, pp. 1–8.

[90] H. Ma, H. Yang, M. R. Lyu, and I. King, "Sorec: social recommendation using probabilistic matrix factorization," in *CIKM*, 2008, pp. 931–940.

[91] G. Bouchard, D. Yin, and S. Guo, "Convex collective matrix factorization," in *AISTATS*, 2013, pp. 144–152.

[92] A. P. Singh and G. J. Gordon, "Relational learning via collective matrix factorization," in *SIGKDD*, 2008, pp. 650–658.

[93] M. Rahman, M. Bhuiyan, and M. A. Hasan, "Graft: an approximate graphlet counting algorithm for large graph analysis," in *CIKM*, 2012, pp. 1467–1471.

[94] R. Rossi and J. Neville, "Time-evolving relational classification and ensemble methods," in *PAKDD*, 2012.

[95] L. A. Breslow and D. W. Aha, "Simplifying decision trees: A survey," *KER*, vol. 12, no. 01, pp. 1–40, 1997.

[96] J. Neville and D. Jensen, "Iterative classification in relational data," in *SRL Workshop*, 2000, pp. 42–49.

[97] R. Kohavi and G. John, "Wrappers for feature subset selection," *Artificial intelligence*, vol. 97, no. 1-2, pp. 273–324, 1997.

[98] G. Robins, P. Pattison, Y. Kalish, and D. Lusher, "An introduction to exponential random graph (p*) models for social networks," *Social Networks*, vol. 29, pp. 173–191, 2007.

[99] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *JMLR*, vol. 3, pp. 1157–1182, 2003.

[100] S. Milojević, "Power law distributions in information science: Making the case for logarithmic binning," *JASIST*, vol. 61, no. 12, pp. 2417–2425, 2010.

[101] R. A. Rossi, L. K. McDowell, D. W. Aha, and J. Neville, "Transforming graph data for statistical relational learning," *JAIR*, vol. 45, pp. 363–441, 2012.

[102] M. Steyvers and T. Griffiths, "Probabilistic topic models," *Handbook of latent semantic analysis*, vol. 427, no. 7, pp. 424–440, 2007.

[103] Q. Lu and L. Getoor, "Link-based classification," in *ICML*. AAAI Press, 2003, pp. 496–503.

[104] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Application of dimensionality reduction in recommender system–a case study," in *WebKDD*, 2000.

[105] I. Fodor, "A Survey of Dimension Reduction Techniques," *US DOE Office of Scientific and Technical Information*, vol. 18, 2002.

[106] S. Boriah, V. Chandola, and V. Kumar, "Similarity measures for categorical data: A comparative evaluation," in *Proceedings of 2008 SIAM Data Mining Conference*, 2008, pp. 243–254.

[107] D. Lin, "An information-theoretic definition of similarity," in *ICML*. Morgan Kaufmann, 1998, pp. 296–304.

[108] R. Lichtenwalter, J. Lussier, and N. Chawla, "New Perspectives and Methods in Link Prediction," in *SIGKDD*. ACM, 2010.

[109] J. Chang and D. Blei, "Relational topic models for document networks," in *AISTATS*, 2009.

[110] R. Rossi and J. Neville, "Modeling the evolution of discussion topics and communication to improve relational classification," in *SIGKDD SOMA*, 2010, pp. 1–10.

[111] S.-H. Cha, "Comprehensive survey on distance/similarity measures between probability density functions," *M3AS*, vol. 1, 2007.

[112] Y. Yao, "Information-theoretic measures for knowledge discovery and data mining," in *Entropy Measures, Maximum Entropy Principle & Emerging Applications*, 2003, pp. 115–136.

[113] C. Aggarwal, Y. Zhao, and P. Yu, "On the use of side information for mining text data," *TKDE*, 2012.

[114] I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh, "Feature extraction," *Foundations and applications*, 2006.

[115] M. Kearns, Y. Mansour, and A. Y. Ng, "An information-theoretic analysis of hard and soft assignment methods for clustering," in *Learning in graphical models*, 1998, pp. 495–520.

[116] D. N. Reshef, Y. A. Reshef, H. K. Finucane, S. R. Grossman, G. McVean, P. J. Turnbaugh, E. S. Lander, M. Mitzenmacher, and P. C. Sabeti, "Detecting novel associations in large data sets," *science*, vol. 334, no. 6062, pp. 1518–1524, 2011.

[117] C. Mallows, "Some comments on C p," *Technometrics*, vol. 42, no. 1, pp. 87–94, 1973.

[118] E. Hannan and B. Quinn, "The determination of the order of an autoregression," *J. Royal Stat. Soc.: Ser. B*, pp. 190–195, 1979.

[119] G. Schwarz, "Estimating the dimension of a model," *The annals of statistics*, vol. 6, no. 2, pp. 461–464, 1978.

[120] J. Shao, "Bootstrap model selection," *J. Amer. Stat. Assoc.*, vol. 91, no. 434, pp. 655–665, 1996.

[121] E. George and R. McCulloch, "Variable selection via Gibbs sampling," *J. Amer. Stat. Assoc.*, pp. 881–889, 1993.

[122] S. Natarajan, T. Khot, K. Kersting, B. Gutmann, and J. Shavlik, "Gradient-based boosting for statistical relational learning: The relational dependency network case," *Machine Learning*, vol. 86, pp. 25–56, 2012.

[123] T. Khot, S. Natarajan, K. Kersting, and J. Shavlik, "Learning markov logic networks via functional gradient boosting," in *ICDM*. IEEE, 2011, pp. 320–329.

[124] K. Henderson, B. Gallagher, L. Li, L. Akoglu, T. Eliassi-Rad, H. Tong, and C. Faloutsos, "It's Who You Know: Graph Mining Using Recursive Structural Features," in *SIGKDD*, 2011, pp. 1–10.

[125] J. Davis, I. Ong, J. Struyf, E. Burnside, D. Page, and V. S. Costa, "Change of representation for statistical relational learning," in *IJCAI*, 2007, pp. 2719–2725.

[126] T. Huynh and R. Mooney, "Discriminative structure and parameter learning for markov logic networks," in *ICML*, 2008.

[127] L. De Raedt and I. Thon, "Probabilistic rule learning," *Inductive Logic Programming*, vol. 6489, pp. 47–58, 2010.

[128] N. Landwehr, A. Passerini, L. De Raedt, and P. Frasconi, "Fast learning of relational kernels," *Machine learning*, vol. 78, no. 3, pp. 305–342, 2010.

[129] L. Getoor, N. Friedman, D. Koller, and B. Taskar, "Learning probabilistic models of relational structure," in *ICML*, 2001, pp. 170–177.

[130] S. Kok and P. Domingos, "Learning the structure of Markov logic networks," in *ICML*, 2005, pp. 441–448.

[131] L. Mihalkova and R. Mooney, "Bottom-up learning of Markov logic network structure," in *ICML*, 2007, pp. 625–632.

[132] H. Khosravi, O. Tong Man, X. Xu, and B. Bina, "Structure learning for markov logic networks with many descriptive attributes," in *AAAI*, 2010, pp. 487–493.

[133] F. Murtagh and P. Contreras, "Algorithms for hierarchical clustering: an overview," *DMKD*, vol. 2, no. 1, pp. 86–97, 2012.

[134] P. Berkhin, "Survey of clustering data mining techniques," *Recent Advances in Clustering*, vol. 10, pp. 25–71, 2006.

[135] X. Zhu, "Semi-supervised learning literature survey," *Computer Science, University of Wisconsin-Madison*, 2006.

[136] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.

[137] J. Bezdek, R. Ehrlich *et al.*, "Fcm: The fuzzy c-means clustering algorithm," *Comp. & Geosci.*, vol. 10, no. 2-3, pp. 191–203, 1984.

[138] C. Rasmussen, "The infinite gaussian mixture model," *NIPS*, vol. 12, no. 5.2, p. 2, 2000.

[139] H. Abdi and L. J. Williams, "Principal component analysis," *Computational Statistics*, vol. 2, no. 4, pp. 433–459, 2010.

[140] B. Schölkopf, A. Smola, and K. Müller, "Kernel principal component analysis," *ICANN*, pp. 583–588, 1997.

[141] Y.-X. Wang and Y.-J. Zhang, "Nonnegative matrix factorization: A comprehensive review," *TKDE*, vol. 25, no. 6, pp. 1336–1353, 2013.

[142] M. Mahoney and P. Drineas, "Cur matrix decompositions for improved data analysis," *PNAS*, vol. 106, pp. 697–702, 2009.

[143] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," *NIPS*, vol. 20, pp. 1257–1264, 2008.

[144] K.-L. Du and M. Swamy, "Independent component analysis," in *Neural Networks and Statistical Learning*, 2014, pp. 419–450.

[145] P. Comon, "Independent component analysis, a new concept?" *Signal processing*, vol. 36, no. 3, pp. 287–314, 1994.

[146] C. Eckart and G. Young, "The approximation of one matrix by another of lower rank," *Psychometrika*, pp. 211–218, 1936.

[147] C. H. Ding, X. He, and H. D. Simon, "On the equivalence of nonnegative matrix factorization and spectral clustering." in *SDM*, vol. 5, 2005, pp. 606–610.

[148] C. Ding, T. Li, and W. Peng, "On the equivalence between non-negative matrix factorization and probabilistic latent semantic indexing," *CSDA*, vol. 52, no. 8, pp. 3913–3927, 2008.

[149] M. Heiler and C. Schnörr, "Controlling sparseness in non-negative tensor factorization," in *ECCV*, 2006, pp. 56–67.

[150] A. Cichocki, A. H. Phan, and C. Caiafa, "Flexible hals algorithms for sparse non-negative matrix/tensor factorization," in *MLSP*. IEEE, 2008, pp. 73–78.

[151] A. Cichocki, R. Zdunek, S. Choi, R. Plemmons, and S.-i. Amari, "Novel multi-layer non-negative tensor factorization with sparsity constraints," in *Adap. & Nat. Comp. Alg.*, 2007, pp. 271–280.

[152] S. Gilpin, T. Eliassi-Rad, and I. Davidson, "Guided learning for role discovery (glrd): Framework, algorithms, and applications," in *SIGKDD*, 2013, pp. 1–9.

[153] L. De Lathauwer, B. De Moor, and J. Vandewalle, "On the best rank-1 and rank-n approximation of higher-order tensors," *SIAM. J. Matrix Anal. & Appl.*, vol. 21, no. 4, pp. 1324–1342, 2000.

[154] D. J. Cook, L. B. Holder, and G. M. Youngblood, "Graph-based analysis of human transfer learning using a game testbed," *TKDE*, vol. 19, no. 11, pp. 1465–1478, 2007.

[155] P. D. Grünwald, *The minimum description length principle*. MIT press, 2007.

[156] J. Rissanen, "Modeling by shortest data description," *Automatica*, vol. 14, no. 5, pp. 465–471, 1978.

[157] R. Tibshirani, G. Walther, and T. Hastie, "Estimating the number of clusters in a data set via the gap statistic," *J. Royal Stat. Soc.: Ser. B (Stat. Meth.)*, vol. 63, no. 2, pp. 411–423, 2001.

[158] D. Pelleg and A. Moore, "X-means: Extending k-means with efficient estimation of the number of clusters," in *ICML*. San Francisco, 2000, pp. 727–734.

[159] S. Dudoit and J. Fridlyand, "A prediction-based resampling method for estimating the number of clusters in a dataset," *Genome biology*, vol. 3, no. 7, 2002.

[160] G. Celeux and G. Soromenho, "An entropy criterion for assessing the number of clusters in a mixture model," *J. Class.*, vol. 13, no. 2, pp. 195–212, 1996.

[161] C. Sugar and G. James, "Finding the number of clusters in a dataset," *J. Amer. Stat. Assoc.*, vol. 98, no. 463, pp. 750–763, 2003.

[162] S. Salvador and P. Chan, "Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms," in *ICTAI*. IEEE, 2004, pp. 576–584.

[163] S. Dray, "On the number of principal components: A test of dimensionality based on measurements of similarity between matrices," *Comp. Stat. & Data Anal.*, vol. 52, pp. 2228–2237, 2008.

[164] H. Eastment and W. Krzanowski, "Cross-validatory choice of the number of components from a principal component analysis," *Technometrics*, pp. 73–77, 1982.

[165] D. Jackson, "Stopping rules in principal components analysis: a comparison of heuristical and statistical approaches," *Ecology*, pp. 2204–2214, 1993.

[166] S. Wold, "Cross-validatory estimation of the number of components in factor and principal components models," *Technometrics*, pp. 397–405, 1978.

[167] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval*. Cambridge University Press, 2008, vol. 1.

[168] A. Jalali, C. C. Johnson, and P. K. Ravikumar, "On learning discrete graphical models using greedy methods," in *NIPS*, 2011, pp. 1935–1943.

[169] A. Cichocki and R. Zdunek, "Regularized alternating least squares algorithms for non-negative matrix/tensor factorization," in *Advances in Neural Networks*, 2007, pp. 793–802.

[170] S. Vijayakumar, A. D'souza, and S. Schaal, "Incremental online learning in high dimensions," *Neural computation*, vol. 17, no. 12, pp. 2602–2634, 2005.

[171] J. Chan, S. Lam, and C. Hayes, "Increasing the scalability of the fitting of generalised block models for social networks," in *IJCAI*, 2011, pp. 1218–1224.

[172] J. Yin, Q. Ho, and E. Xing, "A scalable approach to probabilistic latent space inference of large-scale networks," in *NIPS*, 2013, pp. 422–430.

[173] R. Ranganath, C. Wang, B. David, and E. Xing, "An adaptive learning rate for stochastic variational inference," in *ICML*, 2013, pp. 298–306.

[174] N. Lopes and B. Ribeiro, "Non-negative matrix factorization implementation using graphic processing units," in *IDEAL*. Springer, 2010, pp. 275–283.

[175] H.-F. Yu, C.-J. Hsieh, S. Si, and I. S. Dhillon, "Scalable coordinate descent approaches to parallel matrix factorization for recommender systems." in *ICDM*, 2012, pp. 765–774.

[176] O. Trelles, P. Prins, M. Snir, and R. C. Jansen, "Big data, but are we ready?" *Nature Rev. Gen.*, vol. 12, no. 3, pp. 224–224, 2011.

[177] A. McCallum, X. Wang, and A. Corrada-Emmanuel, "Topic and role discovery in social networks with experiments on enron and academic email," in *JAIR*, 2007, pp. 249–272.

[178] M. Danilevsky, C. Wang, N. Desai, and J. Han, "Entity role discovery in hierarchical topical communities," 2013.

[179] P. Domingos, "A few useful things to know about machine learning," *Comm. of the ACM*, vol. 55, no. 10, pp. 78–87, 2012.

[180] R. Caceres, K. Carter, and J. Kun, "A boosting approach to learning graph representations," *preprint arXiv:1401.3258*, 2014.

[181] Y. MarcAurelio Ranzato, L. Boureau, and Y. LeCun, "Sparse feature learning for deep belief networks," *NIPS*, vol. 20, pp. 1185–1192, 2007.

[182] Y. Bengio, "Learning deep architectures for ai," *Foundations and Trends in ML*, vol. 2, no. 1, pp. 1–127, 2009.