Aalto University
School of Science
Degree Programme in Computer, Communication, and Information Science

Albert Merlin Arockiasamy

A combinatorial approach to role discovery

Master's Thesis
Espoo, December 8, 2016

Supervisor: Professor Aristides Gionis, Aalto University
Advisor: Prof. Aristides Gionis
Nikolaj Tatti, D.Sc.

Aalto University
School of Science
Degree Programme in Computer, Communication, and Information Science

ABSTRACT OF MASTER'S THESIS

Author:	Albert Merlin Arockiasamy		
Title:	A combinatorial approach to role discovery		
Date:	December 8, 2016	**Pages:**	v + 48
Major:	Computer Science	**Code:**	SCI3042
Supervisor:	Professor Aristides Gionis		
Advisor:	Prof. Aristides Gionis Nikolaj Tatti, D.Sc.		

We provide a new formulation for the problem of role discovery in graphs. Our definition is structural and recursive: two vertices should be assigned to the same role if the roles of their neighbors, when viewed as multi-sets, are similar enough. An attractive characteristic of our approach is that it is based on optimizing a well-defined objective function, and thus, contrary to previous approaches, the role-discovery task can be studied with the tools of combinatorial optimization.

We demonstrate that, when fixing the number of roles to be used, the proposed role-discovery problem is **NP**-hard, while another (seemingly easier) version of the problem is **NP**-hard to approximate. On the positive side, despite the recursive nature of our objective function, we can show that finding a *perfect* (zero-cost) role assignment with the minimum number of roles can be solved in polynomial time. We do this by connecting the zero-cost role assignment with the notion of equitable partition. For the more practical version of the problem with fixed number of roles we present two natural heuristic methods, and discuss how to make them scalable in large graphs.

Keywords:	role detection, graph mining, social-network analysis, data mining
Language:	English

Acknowledgements

I take this opportunity to show my gratitude to Data Mining group at ICS department. It was a great opportunity to be part of the group and share moments with them. This thesis was made possible only by supportive guidance of Prof. Aristides Gionis, the head of Data Mining group. I am also grateful to Dr. Nikolaj Tatti for his enormous help and patience. He was always encouraging and guiding me with interest. I would like to thank my friends from Fadco program with whom I shared my academic space and learned a lot. Finally, I would like to express my deep gratitude to my parents and family members for their love, support and patience.

Espoo, December 8, 2016

Albert Merlin Arockiasamy

Contents

Chapter 1

Introduction

I recently read a quote attributed to many people as authors: "If you're not networking, you're not working". The quizzical quote aptly defines our daily life. Being social beings, we are always striving to better our social networks, thereby making society a better place. With the advancement in technology, the networks have spanned across continents and 'It's a small world' has become true. The term social network is now attributed to network one has on the internet. Social network analysis, a research area that focuses on analysing the structures and patterns of the social network, has become an important method in modern sociology as well as in consumer tool.

Social network analysis has attracted considerable interest and attention from researchers of social and behavioural science. Many of them believe that the network perspective allows new leverage for answering standard social and behavioural questions by giving precise formal definitions. Network analysis is performed by developing models of the network and analysing them. In order to develop the model without losing any information, graphs are utilized.

Modeling interconnected entities as graphs (or networks) allows us to study the global structure and function of a system, instead of looking at single entities in isolation. Often, the role of each actor is expected to be identified such that the relational ties can be defined or analysed.

The understanding of the structure of a network in a holistic way can be further supported by our ability to understand the *role* of a single vertex with respect to its local neighborhood, or with respect to the whole network. Accordingly, *role discovery* has emerged as an important graph-mining task [11, 15, 17, 37, 39, 44, 45], together with other standard graph-mining problems, such as community detection, link prediction, etc.

Role discovery can be a valuable tool for exploratory graph mining. For instance, identifying the role of a person in a social network may provide

cues for understanding the social behavior of the person in relation to his/her peers. Similarly, identifying the role of a vertex in a technological network may give useful information about the function of the vertex in the network, or it may be used to detect anomalies [35]. In fact, Rossi and Ahmed [37] provide an extensive and well-documented list of graph-mining tasks that can be facilitated by role discovery. The list includes applications such as classification, active learning, graph visualization, transfer learning, graph compression, entity resolution, and more.

Problem statement

Role discovery can be seen as a process that partitions the vertices of a graph into equivalence classes. Equivalence classes are typically aiming at capturing the structural relation between the vertices and their neighbors. In this way, roles can represent structural patterns in the graph, such as star centers, bridges, peripheral vertices, vertices in near-cliques, etc. [17].

Many different approaches have been proposed for defining when two vertices should be considered equivalent and, thus, should be assigned to the same role. Some of the first methods proposed in mathematical sociology rely on identifying structural or automorphic equivalence classes [13, 26], while newer methods represent vertices by feature vectors and assign to the same role vertices with similar feature vectors [17, 37, 38, 45].

An attractive definition for discovering roles in networks is based on the concept of *regular equivalence* [13, 43]: According to regular equivalence, two vertices should be assigned to the same role only if their neighbors have the same roles, ignoring multiplicities. For example, for the collaboration network of a company we may discover that vertices with role A ("project manager") are connected to vertices with the role B ("business analyst") and C ("s/w developer"), while vertices with roles B and C are typically not connected to each other.

In this paper, we present a new approach to role discovery, inspired by the definition of regular equivalence. As the original definition is too strict to be of use in real-world datasets, we provide a *relaxation* that provides robustness and can tolerate noise in the data. In particular, we define an objective function that quantifies the degree to which a given role assignment satisfies regular equivalence. Given a target number of roles k we then ask to find the role assignment that minimizes our objective function. We also take multiplicities into account: a vertex with 100 neighbors having role A is treated differently than a vertex with a single neighbor having role A.

The proposed objective function is based on creating a *profile* for each

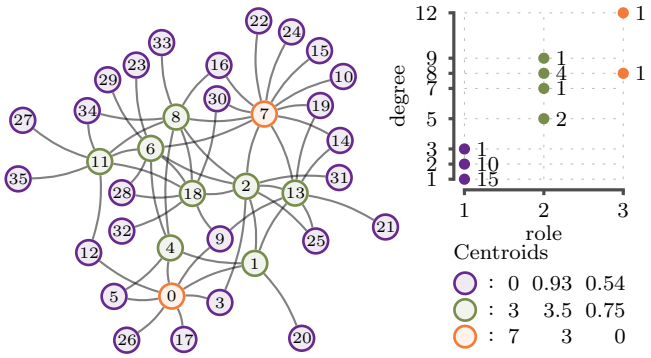


Figure 1.1: Groom network of Rhesus Macaques [3]. 3 roles are assigned using GREEDY initialized by DEG. The scatter plot shows the degree of a vertex as a function of its role.

vertex, which represents the number of neighbor vertices for each other role. Thus, vertices with the same role should have similar profiles. This requirement is expressed as a k-means-type squared-error function. The approach resembles feature-based methods, however, the important difference is the recursive nature of our definition: *roles depend on profiles and profiles depend on roles*.

An example of the roles discovered in a grooming network of monkeys, Rhesus Macaques, is shown in Figure 1.1. In this example, we search for $k = 3$ roles. The role assignment is depicted with different colors, and the profile centroids for each role are shown in the bottom-right subplot. We see that the first role (`purple`) corresponds to relatively isolated individuals, while the other two roles (`green` and `orange`) correspond to more central ones. Observe that the `green` role is indeed different than the `orange` role, as the individuals of the `orange` role are connected to more individuals of the `purple` role, and they are not connected to each other. In the upper-right subplot, we show a scatter-plot of role vs. degree. We see that one of the two vertices with `orange` role has smaller no larger degree than five of the vertices with `green` role, indicating that the role assignment we discover cannot be explained solely by degree.

Our technical contributions are as follows: we formulate the optimization problem and demonstrate that this problem is **NP**-hard. Furthermore, we show that if we fix the profile centroids, the problem still remains **NP**-hard, and cannot be approximated. On the positive side, we show that discovering a *perfect* role assignment, that is, a role assignment with 0 cost, with the smallest number of roles k can be done efficiently in polynomial time. We further propose two natural heuristic algorithms for minimizing the cost func-

tion when k is fixed: (i) the first method is a greedy hill-climbing algorithm, where we optimize a role for a single vertex while keeping the remaining vertex roles constant, (ii) in the second approach, we first fix the profiles, transforming the problem into a standard clustering problem, that we solve using k-means algorithm, and compute the new profiles from the obtained clustering.

We have benchmarked the proposed methods on eight different real-world datasets of varying size. With respect to optimization score the greedy hill-climbing algorithm is found to perform better than the iterative (k-means-like) algorithm. This is consistent with many different initialization strategies, but interestingly enough, the best performance is achieved when greedy is initialized with the solution found by the iterative.

We have also contrasted our methods with RolX [17], a representative feature-based algorithm. Direct comparison is not easy, as there is no available ground truth for the role-discovery problem, and as the RolX algorithm does not provide an optimization criterion. Nevertheless, we find that, when measured with our objective function, RolX obtains high-cost solutions. Compared to a neutral classification task, where the aim is to predict the discovered roles by the corresponding features, the iterative algorithm achieves the highest accuracy.

Structure of the Thesis

The remaining chapters are organized as follows. In Chapter 2, we give a brief overview of roles discovery, focusing on different methods used in discovering roles and the general framework used feature based role discovery. We also present the model RolX which is used for discovering roles in a large graph. We describe RolX since we use it as a benchmark to compare our model RolePro.

In Chapter 3, we discuss some basic graph theory concepts focusing on roles and the formal definition of role discovery problem. The chapter is continued with formal proofs showing that the role discovery process is an NP-hard problem.

In Chapter 4, we provide in detail our model RolePro and algorithms used for role discovery in our model. We initially provide the polynomial algorithm for perfect role assignment and continue by defining a local search algorithm and conclude by defining an iterative algorithm. While discussing each algorithm, their complexities are also discussed.

In Chapter 5, all the algorithms mentioned in Chapter 4 are put into test by experimenting with a number of datasets. Most of the datasets are well-

known to Data Mining community. We also used a collaboration network based on our university research institute, HIIT. The experiments were performed using variations of each algorithm, mainly by having different initial results. Finally, we compare the results of the RolePro model with the results of the RolX model.

In Chapter 6, we review our results and make some concluding remarks.

Additional References

Part of this thesis will appear as a research paper in *International Conference on Data Mining, 2016*. The algorithms were developed jointly with Nikolaj Tatti and Aristides Gionis, while the proof of NP-hardness was given by Nikolaj Tatti.

Chapter 2

Background

We are social beings and society is vital to our survival. In order to have a healthy relationship in the society, every one of us has a social role through which we interact with others. For instance, parents play the role of guardian to their children. Very often the actions performed by us define our roles and vice versa. One example can be the role of a teacher: to help students in gaining knowledge. The teacher will interact with all the students in a class and make sure that they have learned what was intended for the day. If a person looking at the class from outside can easily identify the teacher. Hence our actions define The notion of action and role interaction has been drawn from the field of "Role Theory" [5] in sociology, which states that people behave in predictable ways based on their social roles.

Role discovery is the process of identifying the actions of people and classifying the people using their actions. These days people interact in virtual social networks such as Facebook, Twitter. The same process of role discovery can be applied to such social networks as well to identify roles of each person interacting in the network. In order to perform role discovery, a snapshot of the social network is captured and converted to a graph where nodes are the persons and edges are the interaction between person. People with similar features are grouped. So role discovery is a process that categorizes nodes into different classes of equivalent nodes and each class is labeled with a role.

Two nodes are equivalent, they are interchangeable without loss of information if they interact with other nodes in the network in identical ways. There have been many simpler, relaxed versions of equivalence in recent times, e.g., structural equivalence [43], stochastic equivalence [19]. Based on this relaxation, role discovery can be defined as a process that takes a network graph as input, identifies the equivalent nodes, clusters sets of nodes with similar structural patterns and assigns a role. Two nodes are assigned

to the same role if they have similar structures.

Role-discovery methods can be broadly classified into three categories [37]:

- graph-based role discovery
- feature-based role discovery
- hybrid-based role discovery

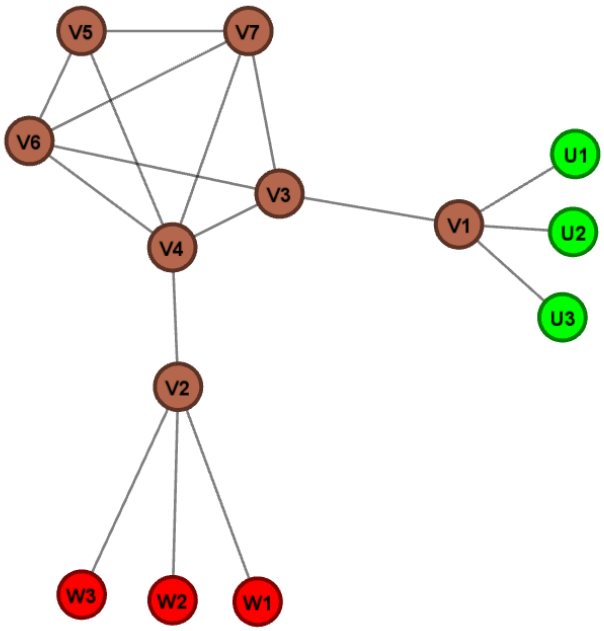


Figure 2.1: Toy example to reveal the differences between the role equivalences.

Graph-based role discovery

Graph-based methods compute roles directly from the graph representation. A number of different definitions have been suggested for quantifying when two nodes are equivalent and should be assigned to the same role. We first look at some of the important equivalences that are used in graph-based role discovery.

Structural equivalence

Structural equivalence states that any two nodes are equivalent nodes if they have same connection pattern to the same set of neighbors. Therefore, two nodes v_i and v_j are structurally equivalent if they have the same neighbors, i.e., $N(v_i) = N(v_j)$. In Figure 2.1, the nodes in red color are structurally equivalent as they have the same (identical) neighbor. There have been many relaxations of structural equivalences [43], [19] that are based on the fact that structurally equivalent nodes are two links away.

Automorphic equivalence

Isomorphism is a mapping from a graph to another graph such that the properties of mapped one are preserved in the mapping one. An automorphism is an isomorphism from one graph to itself thereby preserving symmetries. In formal notation, a node u is automorphically equivalent to node v if there exists an automorphism p such that $u = p(v)$ [19]. Moreover, any set of structural equivalences is also automorphic. In other words, structural equivalences ask whether a node can be exchanged with another node while preserving the relationships of that node, whereas automorphic equivalence focuses on a set of nodes that are exchangeable as subgraphs. In Figure 2.1, the nodes with colors red and green are automorphic and can be clustered into a single class (also known as a role). Similarly, nodes $v1$ and $v2$ form another role.

Regular equivalence

In order to preserve the social concepts, roles definitions are relaxed into regular equivalences. It is based on an idea that nodes play the same role if they are connected to role-equivalent nodes as opposed to structural equivalent nodes where the nodes are to be connected to identical nodes. Intuitively, nodes with same roles are not forced to have a connection to the same neighbors or even the same number of neighbors. From Figure 2.1,nodes in red and green colors are regularly equivalent, while $v1$ and $v2$ form another class of regularly equivalent nodes. Furthermore, $v3$ and $v4$ represent the third role and the last three nodes form the fourth roles.

Stochastic equivalence

Stochastic equivalence was introduced since regular equivalence is strict where each node is assigned to one role, not more. Nodes are assigned to roles such

that the probability of linking a node with other nodes are the same for all the nodes within the role.

Methods for graph-based role discovery

Graph-based roles are composed of the graph, often represented as an adjacency matrix. Though our focus is on feature-based roles, we present a synopsis of some of the known methods for graph-based role discovery.

Blockmodels Blockmodels is one of the most popular role discovery techniques in social network analysis. A network is represented as a role-interaction graph where roles are nodes and interaction between the roles are the edges.

One of the first methods based on blockmodels that emerged was CONCOR (convergence of iterated correlations), proposed by Breiger et al. [8]. The method computes the correlation of the adjacency matrix of the graph. Using the correlation, it computes a correlation matrix and so on. The process is continued until all entries become either 1 or -1.

The main disadvantage of blockmodels is that they are significantly slow for large graphs that are present in social networks. For example, one recent model, mixed-membership stochastic blockmodels for dynamic networks (dMMSB) [2], takes around 24 hours to compute for 1,000 nodes. This is because the model has quadratic complexity.

Row/column similarity of adjacency matrix Besides the block models, there are other methods that utilize variants of similarity [9]. These methods are made of two steps: For each pair of tuples in the adjacency matrix, distance (similarity) is calculated using any of the traditional methods such as euclidean distance or correlation. Using the calculated distance values, the nodes are clustered. Simple clustering methods (e.g., hierarchical clustering or multi-dimensional scaling (MDS) [22]) can be used.

This method can be faster than block models methods if properly configured [7] but the results are harder to interpret. For instance, we get two clusters as final results, and it is difficult to identify one of them as a cluster of teachers and the other as a cluster of students.

Feature-based role discovery

Feature-based role discovery is a more modern approach that relies on representing each node in the graph by a feature vector and assigning them to the

same roles if they have a similar feature vectors [15, 17, 37, 38, 45]. Features can be extracted from graph-based properties of each node, such as, degree, clustering coefficient, centrality, etc., or combined with other information that may be available for the graph nodes, such as dynamic behaviour or node attributes. The set of features derived are based on some transformation of the graph $f(G) = \mathbf{X}$ where G is a graph, $\mathbf{X}$ is the set of features and $f(.)$ is a collection of transformation functions over the graph. A transformation can be any operations on the graph. Some of the operators used in transformation are shown in Table2.1.

Table 2.1: Graph transformation operators that can be used for feature derivation from a graph.

Operators	Examples
Relational Aggreators[30]	Mean, Median, Mode, Count
Set Operators[28]	Union, Intersection, Multiset
Subgraph Pattern[34]	k-star, k-clique, k-motif
Dimensionality Reduction[40]	SVD, NMF, PCA
Similarity[6]	Cosine Similarity, Mutual Information
Paths/walks[25]	random-walks, k-walks
Text Analysis[10][36]	Latent Ditichlet Allocation (LDA), Probabilistic latent semantic analysis

Once feature vectors are constructed, the assignment the problem can be viewed as a traditional clustering problem, and thus can be approached with classic clustering techniques. From technical point of view, our setting differs fundamentally since our features, namely the roles of neighbors depend on the clustering.

Feature-based roles framework

Feature-based roles framework consists of two main parts:

- Feature construction: Transform a given graph to a set of graph features, often represented as feature vectors.
- Role assignment: Assign nodes that have similar feature vectors to same roles.

The basic framework of feature-based role discovery is shown in Figure 2.2. When a graph is provided for role assignment, feature construction operation is performed on the graph to generate features based on the graph structure and attributes. Initially, the graph G and some initial nodes or edge attributes are provided, but along the way of feature construction, additional features may be added to the feature set $\mathbf{X}$. Once the feature set $\mathbf{X}$ is constructed for each node, the node-feature vector is passed as input to role assignment operation. Role assignment operation uses some of the well-known clustering or approximation algorithms to group nodes with similar features and assign them same roles.

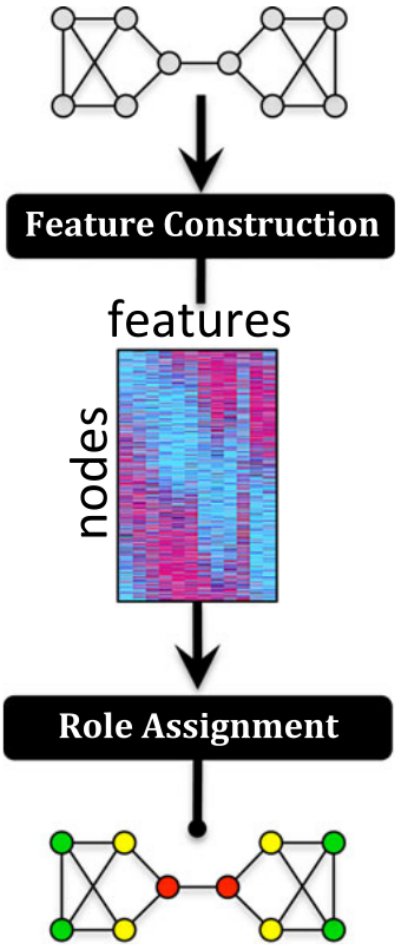


Figure 2.2: The basic framework for feature-based roles discovery. It consists of feature construction and role assignment operations. Graph information is the input and the roles for each node is the output. Node-feature vector is the intermediate output.

Feature construction mainly consists of four steps for constructing node-features vector:

1. *Feature classes selection.* Select the types of features that need to be constructed using the graph data. Some of the feature classes are structural features (degree, betweeness, clustering a coefficient number of 2-star patterns, triangles, etc.), link-value features (paths), node-value features (mode of adjacent node values), non-relational features (attributes).

2. *Feature operators selection.* The operations to be performed on the selected graph values to gain the required features. These operators define the search space of features that is to be used to construct a feature set. Some of the feature operators are already mentioned in Table 2.1. Most of these operators are suitable to be used for computing feature values for nodes and edges of a graph. However, some of these operators cannot be used for constructing structural features as they rely on non-relational data.

3. *Feature search strategy.* Select a strategy to search over the feature space. The strategy may be exhaustive (looking at all features in the feature space), random (using only a fraction of the space by sampling), or guided (using heuristics to select features from the feature space).

4. *Feature selection.* Determine how features are evaluated or scored and pruned in the iteration. All the searched features are not selected for final representation. The goal is set of features that are minimal. Hence correlation and log binning methods are used to remove redundant and noise features.

Once a node-feature vector is constructed, feature-based roles are discovered by assigning nodes with similar features to the same role. There are two main classes of methods for assigning roles to nodes: role clustering methods or low-rank approximation methods. In role clustering methods, clustering algorithms are used to cluster nodes with similar features and assign the same roles to all the nodes within a cluster. The two major clustering types are hierarchical (agglomerative and divisive clustering) and partitioning algorithms (k-means, k-medoids, and self-organizing maps). The above-mentioned algorithms are hard-clustering algorithms: they assign a node to a single role. There are also algorithms that are soft-clustering as they assign multiple roles to a node. Some of them are fuzzy C-means [4] and Gaussian mixture models [32]. Low-rank approximation methods select a relatively small number r

and compute a low-rank matrix that best approximates the original feature matrix **X**. This is done using dimensionality reduction methods such as SVD [16], PCA [1], Kernel-PCA [41], MDS [23], NMF [42].

Hybrid approaches for role discovery

Hybrid approaches utilize both the graph and feature representation to generate roles. The hybrid methods are categorised into two classes based on the method used first. The first class uses a graph-based approach initially, may be a block model to extract roles directly from the graph, and uses the resultant roles as initial attributes in a relational feature based learning system. The refined output features are then used to determine the number of roles to be derived and roles for each node. The main disadvantage of this methods is its scalability since block models are slow for large datasets.

The second class of hybrid methods utilizes multiple data sources for role assignment. Initially a feature representation **X** is generated and graph based role assignment is performed based on **X**. Some of the known hybrid approaches are based on tensor factorization methods [12], [14] or collective matrix-tensor factorization (CMTF) methods [27].

RolX: Structural role extraction in large graphs

Henderson et al. have proposed RolX, a non-negative matrix factorization-based (NMF) approach to decompose a node-feature matrix into node-role and role-feature matrices [17]. RolX follows three steps similar to the general framework of feature-based roles: feature extraction, feature grouping and model selection.

In feature extraction, RolX uses a recursive feature extraction method (ReFeX) [18] which is based on constructing node-role matrix using NMF. ReFeX is scalable to large graphs. It extracts local and egonet features based on counts of links adjacent to a node and aggregates egonet-based features in recursion until no informative features can be added. In feature grouping step, RolX groups nodes with similar features using low-ranking approximation method. The node-feature matrix is decomposed into two non-negative node-role and role-feature matrices. The number of roles to be used is derived by model selection method. It is done by using Minimum Description Length criterion [33].

In order to make sense of roles assigned by RolX, two measures were used known as *node sense* and *neighbor sense*. *Node sense* takes a node-role matrix

of RolX and a matrix of node measurements (degree, page rank etc) as input and calculates the contribution of role to node measurements. *Neighbor sense* is similar to *node sense* except that it uses neighbor node-role matrix instead of node measurements. The research shows that RolX is able to find roles with distinct characteristics, and the derived roles are transferrable from one network to another. We will be using RolX to compare the results obtained by our combinatorial profile based method.

Chapter 3

Role Discovery Preliminaries

Graph theory is a powerful tool to study various types of networks such as road networks, telephone networks, social networks and the World Wide Web. In order to discover roles in a social network, we need to understand the preliminaries of graphs. Let us now look at the basic graph theory terms that are useful in role discovery. We will be focusing on terms that are used in this thesis.

Preliminaries and notation

A graph G is a pair (V, E), where V is a set of *vertices* (also called *nodes*) and E is a multiset of unordered pairs of vertices called *edges*. We denote an edge $e \in E$ as a pair u, v, where $u, v \in V$ and u and v are endpoints of e and $|V| = n$, $|E| = m$. We consider both directed and undirected edges. Often vertices and edges can have attributes attached to them. One attribute which is used often in graph theory is the weighted values on edges that define the importance of the edge. We say two edges are *parallel* if they have the same end points. We call a graph *simple* if it does not contain multiple edges.

A vertex is incident on an edge if it one end of the edge. A vertex u is adjacent to vertex v if u, v is an edge in the graph. Neighbors of a vertex v are set of vertices that are adjacent to the vertex. The *degree* $deg(v)$ of a vertex is the number of edges incident to v in the graph.

A *role assignment* $r : V \to [1, k]$ is a function mapping each vertex v to an integer between 1 and k, which is interpreted as a role id and the total number of roles k is given. Given a role assignment r, the *profile* of a vertex v for that role assignment is a k-dimensional vector $\mathbf{p}(v; r) = \mathbf{x}$, where x_i, the i-th coordinate of $\mathbf{p}(v; r)$, is the number of vertices with role i that are

adjacent to v,

$$x_i = |\{(v, w) \in E \mid r(w) = i\}|.$$

Problem formulation and complexity

Our goal is to assign roles to the vertices of the graph G. Our guiding principle for assigning roles to the graph vertices is that *vertices assigned to the same role should have more similar profiles than vertices assigned to different roles.*

As vertex profiles are k-dimensional vectors, we quantify the similarity between them using the Euclidean distance

$$d(\mathbf{x}, \mathbf{y}) = ||\mathbf{x} - \mathbf{y}||_2 = (\sum_{i=1}^{k} |x_i - y_i|^2)^{1/2}$$

where $\mathbf{x}$ and $\mathbf{y}$ are vertex profiles.

Furthermore, each role $i \in [1, k]$ is represented by a k-dimensional vector $\mathbf{c}_i$, which is selected as a representative of the profile vectors of all vertices assigned to role i. We use the Euclidean distance to define the distance between a vertex profile $\mathbf{p}(v; r)$ and a role representative vector $\mathbf{c}$. For simplicity of notation we write

$$d(v, \mathbf{c}; r) = d(\mathbf{p}(v; r), \mathbf{c}) = \|\mathbf{p}(v; r) - \mathbf{c}\|_2 .$$

The role-mining problem we consider can now be formulated as follows.

Proposition 1. (ROLES) *Given a graph $G = (V, E)$ and an integer k, find a role assignment $r : V \to [1, k]$ and k representative role vectors $\mathbf{c}_1, \ldots, \mathbf{c}_k$ that minimize the cost*

$$c(r, \mathbf{c}_1, \ldots, \mathbf{c}_k) = \sum_{v \in V} d\big(v, \mathbf{c}_{r(v)}; r\big)^2 .$$

We can show that the ROLES problem is **NP**-hard. The proof of the following proposition is obtained by a reduction from the 3D-MATCHING problem.

Proposition 2. ROLES *is **NP**-hard.*

Proof. We will prove the hardness from 3D-MATCHING: a **NP**-hard problem where we are given a universe U and family $\mathscr{M}$ of sets of size 3, and we are asked to find a maximum disjoint cover.

Assume that we are given an instance $(U, \mathscr{M})$ of 3D-Matching. Let $n = |U|$ and $m = |\mathscr{M}|$. We can safely assume that n is divisible by 3.

Define $h = \binom{n}{3}$ the number of possible sets of size 3 over U. Let $\ell = \binom{n-1}{2}$ be the number of possible sets of size 3 containing a fixed vertex in U. Let us define $t = 8m$ and $s = 2t$.

Graph construction: The graph consists of two major parts.

The first part entails 7 vertex sets: we start with A_1 and A_2 with $|A_1| = 3$ and $|A_2| = \ell - 1$. A single vertex, say v, is connected to an additional vertex, say w. We write $A_3 = \{v\}$ and $A_4 = \{w\}$. Futhermore, w is connected to a biclique of $A_5 = K(s, s)$. Each vertex $u \in A_1$ and $v \in A_2$ is connected with a *fat* edge: t copies of a path u–x–v, where x is a vertex unique to the path. These vertices form A_6. Similarly, each vertex $u \in A_1$ and $v \in A_3$ is connected with a *fat* edge; the intermediate vertices form A_7. The final graph consists of t copies of the first part. We redefine A_i to be the union of the corresponding groups in each copy.

The second part is very similar to the first. It entails the vertex sets B_1, B_2, and B_3 with $|B_1| = n$, $|B_2| = h - m$ $|B_3| = m$. Here B_1 corresponds to the universe U, and B_2 and B_3 correspond to triplets of elements in U. Each vertex in B_3 is connected to its own vertex. These vertices form B_4. Each vertex in B_4 is also connected to its own dedicated biclique $K(s, s)$. The vertices in bicliques form B_5. A vertex in B_2 is connected to the corresponding vertices in B_1 with a fat edge of size t. The intermediate vertices form B_6. Similarly, a vertex in B_3 is connected to the corresponding vertices in B_1 with a fat edge of size t. The intermediate vertices form B_7.

Role assignment: Let us now define a role assignment, later this will turn out to be the optimal assignment. Let $\mathscr{W}$, not necessarily a subset of $\mathscr{M}$, be a matching covering completely U.

Let C_3 be the vertices in $B_2 \cup B_3$ corresponding to $\mathscr{W}$. Let $C_1 = B_1$, $C_2 = (B_2 \cup B_3) \setminus C_3$ $C_4 = B_4$, $C_5 = B_5$. Let C_6 be the vertices in $B_6 \cup B_7$ adjacent to C_3, and set $C_7 = (B_6 \cup B_7) \setminus C_6$. We define $r(v) = i$, where $v \in A_i \cup C_i$. Let us write $r_{\mathscr{W}} = r$.

Let us compute the cost of r. To that end, write c_{ij} to be the cost incurred by the jth component of the ith role. The only non-zero costs are c_{24}, c_{42}, c_{34}, and c_{43}. Define $n_i = |\{v; r(v) = i\}|$, and let $\alpha = |A_3 \cup (B_3 \cap C_3)|$ and $\beta = |B_3 \cap C_2|$. We can express the cost as

$$c_{24} + c_{34} + c_{42} + c_{43} = f(\beta, n_2) + f(\alpha, n_3) + 2f(\alpha, n_4),$$

where $f(x, y) = x(y - x)/y$. The counts n_i do not depend on $\mathscr{W}$. Since $\alpha + \beta = n_4$, so the cost of depends on $\mathscr{W}$ only via α. We have $2\alpha \geq 2t \geq n_3, n_4$ and $2\beta \leq 2m \leq t \leq n_2$. Since $f(x, y)$ is a parabola peaking at $x = y/2$, the cost decreases as α increases.

Since $f(x, y) \leq \min(x, y/2 - x)$, we can upper-bound the cost by $n_3 + 2n_4 - 3\alpha + \beta < 4m$.

Role r is optimal: Let r^* be the optimal role assignment with a cost of σ. Consider that if instead of selecting optimal centroids for r^*, we select the optimal centroids—we denote them by μ'—among the profiles of the vertices in the cluster. We know that the cost of r^* w.r.t. μ', say τ, is at most 2σ. If $\tau \geq 8m$, then $\sigma \geq 4m$, the cost of r, which violates the optimality of r^*. Consequently, $\tau < 8m = t$.

Note that μ' are all integral. We can safely assume that the copies of the first part of the graph have the same role assignment. This immediately implies that the profiles of r^* should match *exactly* the centroids μ'.

There are 6 groups with distinct degrees in the first part: $\deg(u) = t\ell$ for $u \in A_1$, $\deg(u) = 3t$ for $u \in A_2$, $\deg(u) = 1 + 3t$ for $u \in A_3$, $\deg(u) = 2s + 1$ for $u \in A_4$, $\deg(u) = s + 1$ for $u \in A_5$, $\deg(u) = 2$ for $u \in A_6 \cup A_7$. Since the vertices from different groups have different degrees, r^* must be a refinement of this partition. Moreover, A_6 connects to A_2 and A_7 connects to A_3. Thus they cannot have the same role. This gives us 7 groups, and r^* must be a refinement of this partition. Since we have only 7 roles, this must be the role assignment.

Let us now consider the second part of the graph. Let $u \in B_1$. The only integral centroid that matches $\deg(u)$ is μ'_1, and the remaining centroids differ in degree by at least of t, consequently $d(u, \mu'_i) \geq t$ for $i \neq 3$. This forces, $r^*(u) = 1$. Similarly, $r^*(u) = 4$, for $u \in B_4$, $r^*(u) = 5$, for $u \in B_5$, $r^*(u) = 6, 7$, for $u \in B_6 \cup B_7$.

Let $u \in B_2 \cup B_3$. If $r^*(u) \neq 2, 3$, then each vertex in a fat edge will introduce a cost of at least 1, when compared to the integral centroids μ'. Since there are t of these vertices, we must have $r^*(u) = 2, 3$. Using a similar argument, among the vertices in $B_2 \cup B_3$ adjacent (by a fat edge) to a vertex $v \in B_1$, only one has the role 3, the remaining adjacent vertices have the role 2.

This shows that $r^* = r_{\mathscr{W}}$, where $\mathscr{W}$ are the sets corresponding to the vertices with role 3. We saw earlier that the cost is minimized when $\alpha = t + |\mathscr{W} \cap \mathscr{M}|$ is maximized. □

Intuitively, problem Roles resembles the k-means clustering problem. However, a careful reader should immediately realize that we are dealing with a much harder problem than k-means clustering. To see this, notice that in the k-means problem we aim to cluster vectors whose coordinates are *fixed*. In the Roles problem, however, we aim to cluster vertex profiles, which are vectors whose coordinates depend on the role assignment. Thus, we are working with a clustering problem in which the data recursively depend

on the output of the clustering problem itself.

To emphasize the difference between ROLES and k-means clustering, consider the standard property of k-means algorithm: *for a fixed cluster membership it is easy to compute optimal representative vectors (centroids), and for fixed centroids, it is easy to compute optimal cluster membership.*

We can show that for the ROLES problem only the first part of the corresponding property holds. Indeed, for a *fixed* role assignment r the profiles of all vertices are also fixed. The representative vector of a role can then be easily computed as the centroid of the profiles of all vertices that are assigned to that role.

On the other hand, when the role centroids $\mathbf{c}_1, \ldots, \mathbf{c}_k$ are fixed it is not easy to compute the optimal role assignment r. We refer to this problem as ROLES-FIXEDCENTROIDS.

Proposition 3. (ROLES-FIXEDCENTROIDS) *Assume a graph $G = (V, E)$ and k centroids $\mathbf{c}_1, \ldots, \mathbf{c}_k$. Find a role assignment $r : V \to [1, k]$ that minimizes the cost function*

$$c(r) = \sum_{v \in V} d\big(v, \mathbf{c}_{r(v)}; r\big)^2 .$$

Proposition 4. *Deciding whether an instance of* ROLES-FIXEDCENTROIDS *has a zero-cost solution is an **NP**-complete problem.*

To prove Proposition 4, we need to introduce the following **NP**-complete problem.

Proposition 5 (TUPLES)**.** *Assume a universe U and a set $\mathscr{S}$ of 5-tuples over U, such that each $u \in U$ occurs in at least 3 tuples. Is there a subset $B \subseteq U$ such that each exactly 2 entries in each $S \in \mathscr{S}$ belong to B.*

Proposition 6. TUPLES *is **NP**-complete.*

Proof. TUPLES is obviously in **NP**. We prove the hardness by reduction from 3-SAT.

Assume that we are given m clauses $C_1, \ldots, C_m$ using n variables, in total. We can safely assume that each clause contains exactly 3 variables by allowing the repetition of a variable.

We define the universe U to contain $2n + 2 + 2m$ variables, which we will denote by t, f, $x_1, \ldots, x_n$, $y_1, \ldots, y_n$, $d_1, \ldots, d_m$, and $e_1, \ldots, e_m$. To simplify notation, we will identify x_i with the ith positive literal and y_i with the ith negative literal.

We introduce the following clauses: *(1)* (t, t, f, f, f), *(2)* $(x_i, \neg x_i, t, f, f)$, where $i = 1, \ldots, n$, *(3)* $(\neg c_1, \neg c_2, \neg c_3, d_j, e_j)$, where $C_j = c_1 \vee c_2 \vee c_3$ is the jth clause.

To make sure that each variable occurs in at least 3 tuples, we copy each tuple 3 times.

Assume there is B solving TUPLES. Set the ith variable to be true if $x_i \in B$, and false otherwise. At most 2 variables $\neg c_i$ are in B, for $C_j = c_1 \vee c_2 \vee c_3$. Assume that $\neq c_1 = \notin B$. Since either $x_i \in B$ or $\neg x_i \in B$, this forces $c_1 \in B$, making C_j satisfied.

Now assume that we can satisfy each clause. First set $B = \{t\}$. Add x_i into B if the ith variable is true, otherwise insert $\neg x_i$. Since a clause $C_j = c_1 \vee c_2 \vee c_3$ is satisfied, we have at most 2 entries in B among $\neg c_i$. Insert d_j and/or e_j to B so that $(\neg c_1, \neg c_2, \neg c_3, d_j, e_j)$ contains exactly 2 entries in B. The resulting B solves TUPLES. □

Proof of Proposition 4. The problem is in **NP**. We will prove the hardness from TUPLES. Assume that we are given an instance $(U, \mathscr{S})$ of TUPLES. Let $n = |U|$ and $m = |\mathscr{S}|$. Let ℓ_u be the total number of occurrences, counting multiplicities, of u in $\mathscr{S}$.

Define the graph as follows: For each $u \in U$, add a cycle of length ℓ_u. Note that the cycle is simple since $\ell_u \geq 3$. Denote this cycle by D_u.

For each tuple C_j add a vertex w_j, and connect it to a vertex in D_u, where $u \in C_j$. The connection can and should be done such that each vertex in each cycle is connected to exactly one w_j.

Set the number of roles $k = 3$, and define the following centroids $\mu_1 = (2, 0, 1)$, $\mu_2 = (0, 2, 1)$, and $\mu_3 = (2, 3, 0)$.

Let r be the optimal role assignment for the defined centroids. The cost of is 0 if and only if *(i)* every vertex in $v \in D_u$ has the same role, $r(v) = 1, 2$, *(ii)* $r(w_j) = 3$, *(iii)* for each w_j there are two adjacent vertices with role 1 and three adjacent vertices with role 2.

Consequently, B in TUPLES corresponds to the vertices with role 1. □

Not only does Proposition 4 imply that ROLES-FIXEDCENTROIDS is an **NP**-hard problem, but it also establishes that ROLES-FIXEDCENTROIDS cannot be approximated to any multiplicative factor, no matter how large.

Corollary 7. *Unless* $\boldsymbol{P = NP}$*, there is no polynomial algorithm that provides an approximation guarantee to the* ROLES-FIXEDCENTROIDS *problem.*

Note that even though intuitively ROLES is a more difficult problem than ROLES-FIXEDCENTROIDS, the hardness result obtained for ROLES-FIXEDCENTROIDS is much stronger than the one obtained for ROLES. This could be

just an artifact of our proof techniques and it may be the case that ROLES is also a hard problem to approximate. As we do not provide an approximation algorithm for ROLES in this paper, its approximability is left as an open problem.

Chapter 4

Algorithms for discovering roles

A polynomial algorithm for perfect role assignment

Before presenting our proposed algorithms for the ROLES problem, we first present a polynomial algorithm for finding a perfect role assignment—a solution with cost zero. We will do this by arguing that the perfect role assignment is equivalent to the notion of equitable partition [29], which can be solved exactly.

In the light of the previous **NP**-hardness results, this polynomial algorithm appears somewhat surprising. Note, however, that this polynomial algorithm guarantees to find the minimum number of roles for which there is a solution of cost zero. This is a different question than the one posed by the ROLES problem, where we aim to find a minimum-cost solution for a given number of roles.

Given a graph $G = (V, E)$, a partition of vertices $V_1, \ldots, V_k$ is said to be *equitable* if the edges respect the partition, that is, $(u_1, v_1) \in E$ if and only $(u_2, v_2) \in E$ for any $u_1, u_2 \in V_i$ and $v_1, v_2 \in V_j$, $i, j = 1, \ldots k$. Note that for such a partition, the cost will always be 0, and vice versa. Naturally, there are many possible partitions but there is only partition with the smallest k, and this partition can be discovered with the algorithm that we will present for the sake of completeness. The proof for this algorithm is given in [29].

The polynomial algorithm, named PERFECT, works by first setting $k = 1$ and assigning all vertices to the same role. Then, it iteratively computes the profile of each vertex, and groups together all vertices with the same profile. For each new group, it then assigns a new role (and increases k), and the iterative process continues as long as new roles are created. Pseudocode for PERFECT is given in Algorithm 1.

Algorithm 1: PERFECT(G), computes a perfect assignment with smallest number of roles.

```
r(v) ← 1 for every v ∈ V;
while number of roles increases do
    compute profiles p(v; r);
    group vertices with the same profiles;
    assign a role to each group;
```

Note that in the first iteration of PERFECT the profile of each vertex v is a scalar (1-dimensional vector) equal to the degree of the vertex. Thus, during this first iteration all vertices with the same degree will be grouped together and will be assigned the same role. In subsequent iterations, the vertices with the same degree will be potentially further subdivided into smaller groups, and vertices within a group are assigned to the same role.

As the number of roles can only increase during the execution of PERFECT, the previous observation implies that PERFECT always returns a solution in which the number of roles k is at least as large as the number of distinct degrees in the graph. To verify this property, notice that it is not possible to obtain a perfect role assignment with a smaller number of roles than the number of distinct degrees.

Finally, note that the PERFECT algorithm always terminates, in the worst case when $k = n$ and each vertex is assigned to a unique role.

The main claim characterizing the optimality of the PERFECT algorithm is formulated in the following proposition.

Finally, we analyze the running time of PERFECT.

Proposition 8. *The computational complexity of* PERFECT *is* $\mathcal{O}(mn \log n)$.

Proof. At each iteration we increase the number of roles. Since the number of roles is at most n, we can have at most $\mathcal{O}(n)$ iterations. Next, we analyze the running time of each iteration.

Let us represent the profiles with sorted sparse lists. The size of a profile of a vertex v is $\mathcal{O}(\deg(v))$. The total time of constructing the profiles is then

$$\mathcal{O}\left(\sum_{v \in V} \deg(v) \log \deg(v)\right) \subseteq \mathcal{O}(m \log n)\,.$$

Grouping based on profiles can be done by sorting the profiles in lexicographical order, and then comparing the consecutive profiles.

We next analyze the time needed to sort the profiles in lexicographical order. Notice that comparing the profiles $\mathbf{p}(v)$ and $\mathbf{p}(w)$ of two vertices v and w cannot be done in constant time; instead time $\mathscr{O}(\min\{\deg(v), \deg(w)\})$ is required.

Assume that merge sort is used for ordering the profiles lexicographically. Consider a counter $\Delta(v)$ defined as follows: initially, $\Delta(v)$ is set to 0. Consider, that we are merging two sorted vertex lists during the merge sort. Assume that we are comparing vertex v from the first list and vertex w from the second list, and assume that we determine $\mathbf{p}(v;r) \leq \mathbf{p}(w;r)$. In such case, we increase $\Delta(v)$ by $\deg(v)$. If $\mathbf{p}(v;r) > \mathbf{p}(w;r)$, then we increase $\Delta(w)$ by $\deg(w)$.

The increase of $\Delta(v)$ or $\Delta(w)$ is an upper bound for the time that is required for each comparison, so it immediately follows that sorting can be done in time $\mathscr{O}(\sum_u \Delta(u))$.

Note whenever we increase the counter, we always advance to the next vertex in the list. That is, each counter $\Delta(v)$ can be increased only once during a single merge, and each vertex participates only in $\mathscr{O}(\log n)$ merges. Consequently, $\Delta(v) \in \mathscr{O}(\deg(v) \log n)$. This implies that sorting the profiles using merge sort requires time $\mathscr{O}(\sum_v \deg(v) \log n) = \mathscr{O}(m \log n)$. □

Hill-climbing algorithm

The algorithm discussed in the previous section returns a perfect (zero-cost) role assignment but it does not put any constraint on the number of roles to be used. In fact, as we will see in the experimental section, in most cases, Perfect is forced to use a large number of roles. This is an expected behavior, as the requirement for a perfect role assignment is too rigid.

In this section, we return to the Roles problem (defined in Problem 1) and ask to find a minimum-cost role assignment for a given number of roles k. As the Roles problem is **NP**-hard (Proposition 2) and as a simple variant of the problem is **NP**-hard to approximate (Proposition 4), we present a hill-climbing algorithm that iteratively improves the cost of the role-assignment problem, until convergence. The algorithm is presented and analyzed below.

Assume a role assignment r with optimal centroids $\mathbf{c}$. Let v be a vertex, and let j be an integer. Define a new role assignment r' obtained from r by setting $r'(v) = j$. Let $\mathbf{c}'$ be the optimal centroids with respect to r'.

We define the *gain* to be the difference of the value of the objective function for the two role assignments

$$\text{gain}(v, j; r) = \sum_{v \in V} d\big(v, \mathbf{c}_{r(v)}; r\big) - \sum_{v \in V} d\big(v, \mathbf{c}'_{r'(v)}; r'\big)\,.$$

Algorithm 2: GREEDY(G, k, r_0), hill-climbing algorithm.

```
1 initialize role assignment, r ← r_0;
2 while changes do
3     foreach v ∈ V do
4         j* ← arg max_j gain(v, j);
5         if gain(v, j*) > 0 then
6             update r;
7             update profiles and centroids;
```

A positive gain means that r' produces a smaller cost, making it a better role assignment.

The proposed hill-climbing algorithm, named GREEDY, is illustrated as Algorithm 2. The algorithm starts with an initial role assignment r_0. Then it sequentially tries to improve the score by changing the role of each vertex in the graph. For each vertex v, it changes its assignment to the role j that maximizes the gain $\text{gain}(v, j)$. If there is no role that yields a positive gain, the role of v remains unchanged.

For selecting the initial role assignment r_0 we have experimented with a number of different strategies. More details are given in the experimental section.

As GREEDY involves a number of nested loops, a naïve implementation will make it prohibitively expensive for large networks. However, we show that it is possible to implement GREEDY in a much more efficient manner so that the running time of the inner-most loop (**foreach** loop in Algorithm 2) is linear in the size of the graph, and quadratic only in the number of roles k, which in practice can be assumed to be a very small constant.

Proposition 9. *Assume a role assignment r with optimal centroids $\mathbf{c}$. Let v be a vertex and j be an integer. Let $i = r(v)$. Define $n_\ell = |\{u \in V; r(u) = \ell\}|$ to be the number of vertices having role ℓ. Define $d_\ell = |\{(v, w) \in E; r(w) = \ell\}|$ to be the number of neighbors of v having role ℓ.*

Define a new role assignment r' obtained from r by setting $r'(v) = j$. Let $n'_\ell = |\{u \in V; r'(u) = \ell\}|$. Let $\mathbf{c}'$ be the optimal centroids with respect to r'. Let $\mathbf{c}^*$ be the "intermediate" centroids, where we have moved v from i to j

but we have not updated the profiles of neighbors of v. Then

$$\begin{aligned}\operatorname{gain}(v,j) = &- n_i \left\|\mathbf{c}_i\right\|^2 + n_i' \left\|\mathbf{c}_i^*\right\|^2 - n_j \left\|\mathbf{c}_j\right\|^2 + n_j' \left\|\mathbf{c}_j^*\right\|^2 \\ &+ \sum_{(v,w)\in E} 2\mathbf{p}(w,r)_i - 2\mathbf{p}(w,r)_j - 2 \\ &- \sum_{\ell=1}^{k} 2d_\ell \left[(\mathbf{c}_\ell^*)_j - (\mathbf{c}_\ell^*)_i - d_\ell/n_\ell'\right].\end{aligned}$$

Proof. A known identity $\sum(x_i - \mathbf{c})^2 = \sum(x_i^2 - \mathbf{c}^2)$ allows us to rewrite the gain as

$$\begin{aligned}\operatorname{gain}(v,j) = &\sum_{u\in V} \left\|\mathbf{p}(u,r)\right\|^2 - \left\|\mathbf{p}(u,r')\right\|^2 \\ &- \sum_{\ell=1}^{k} n_\ell \left\|\mathbf{c}_\ell\right\|^2 - n_\ell' \left\|\mathbf{c}_\ell'\right\|^2.\end{aligned}$$

Let us define $B(w)$ to be a term in the first sum. Let us abbreviate $\mathbf{p}(w,r)$ as $\mathbf{p}(w)$. Then $B(w) \neq 0$ if and only if $(v,w) \in E$, and in such a case it is equal to

$$\begin{aligned}B(w) &= \mathbf{p}(w)_i^2 - (\mathbf{p}(w)_i - 1)^2 + \mathbf{p}(w)_j^2 - (\mathbf{p}(w)_j + 1)^2 \\ &= 2\mathbf{p}(w)_i - 2\mathbf{p}(w)_j - 2.\end{aligned}$$

We now consider the second sum, which we will rewrite as

$$\sum_{\ell=1}^{k} n_\ell \left\|\mathbf{c}_\ell\right\|^2 - n_\ell' \left\|\mathbf{c}_\ell^*\right\|^2 + \sum_{\ell=1}^{k} n_\ell' \left\|\mathbf{c}_\ell^*\right\|^2 - n_\ell' \left\|\mathbf{c}_\ell'\right\|^2.$$

The first sum, which we will denote by A is equal to

$$A = n_i \left\|\mathbf{c}_i\right\|^2 - n_i' \left\|\mathbf{c}_i^*\right\|^2 + n_j \left\|\mathbf{c}_j\right\|^2 - n_j' \left\|\mathbf{c}_j^*\right\|^2.$$

Let us write $R(\ell)$ to be a term in the second sum. We can express this term as

$$\begin{aligned}R(\ell) &= n_\ell' \left[(\mathbf{c}_\ell^*)_i^2 - ((\mathbf{c}_\ell^*)_i - \gamma_\ell)^2 + (\mathbf{c}_\ell^*)_j^2 - ((\mathbf{c}_\ell^*)_j + \gamma_\ell)^2\right] \\ &= 2d_\ell \left[(\mathbf{c}_\ell^*)_i - (\mathbf{c}_\ell^*)_j - \gamma_\ell\right], \quad \text{where} \quad \gamma_\ell = d_\ell/n_\ell'.\end{aligned}$$

To conclude, we have

$$\operatorname{gain}(v,j) = -A + \sum_{(v,w)\in E} B(w) - \sum_{\ell=1}^{k} R(\ell),$$

which proves the identity. □

Proposition 10. *The inner loop of* Greedy (**foreach** *loop in Algorithm 2*) *requires time* $\mathcal{O}(k^2n + km)$.

Proof. According to Proposition 9, computing the gain can be done in time $\mathcal{O}(k + \deg(v))$. Consequently, computing j^* requires time $\mathcal{O}(k^2 + k\deg(v))$. Updating the profiles, the centroids, and the role assignment r can be done in time $\mathcal{O}(k + \deg(v))$. Summing over all the vertices gives us a total running time of $\mathcal{O}(k^2n + km)$. □

Iterative algorithm

Proposition 4 states that if we fix centroids, then the problem remains intractable, even worse it cannot be approximated. This is a blowback to the standard iterative heuristic, where one set of parameters is fixed while the other set of parameters is optimized.

However, if we were to fix the *profiles*, then the optimization task transforms into a traditional clustering problem. Once, we have discovered a role assignment, we can recompute the profiles, and repeat until we converge into a local minimum. This is exactly what we do in Algorithm 3.

Algorithm 3: Iterative(G, k), an iterative method for computing roles. Cluster is a standard clustering method.

```
1 r_0 ← Cluster(deg(·), k) ;
2 i ← 0;
3 while cost decreases do
4 |   r_{i+1} ← Cluster(p(·, r_i), k) ;
5 |   i ← i + 1;
```

We still have the task to solve the clustering problem. Here, we resort to a standard k-means clustering algorithm. We will denote the resulting algorithm by Iterative.

Note that computing profiles from the role assignment can be done efficiently in $\mathcal{O}(\max\{kn, m\})$ time, where n is the number of vertices and m is the number of edges. The $\mathcal{O}(kn)$ time is needed to initialize n vectors of length k. If the clustering algorithm allows to have sparse representation of the profiles, the processing time can be further reduced to $\mathcal{O}(m)$ time. Thus, the computational bottleneck is the clustering algorithm, as well as how many iterations we typically need to converge.

Chapter 5

Experimental evaluation

Experimental setup:

Any model must be tested with real-world data to see its relevance, so we use several datasets to evaluate our proposed methods: synthetic and real-world social communication networks. We also use some result comparison measures to evaluate the result of the model. We describe the datasets and the measures in detail below.

Synthetic data

In order to study the nature of the proposed methods, we simulate multiple communication networks varying in size and density. *synth1* is a graph with 10 nodes, divided into two groups. *synth2* is graph with three groups of vertices, say V_1, V_2, V_3, with $|V_1| = 40$ and $|V_2| = |V_3| = 30$. We connect V_1 and V_3 to V_2 and fully connect V_3. After this we apply noise by adding or removing an edge with a probability p. We vary p from 0 to 0.5.

Real-world data

We use eight datasets of real-world interaction. The characteristics of these datasets are summarized in Table 5.1.

- *karate*: a social network of friendships between members of karate club at a US university in 1970.
- *dolphins*: a social network of frequent associations between dolphins in a community living off Doubtful Sound in New Zealand.

- *lesmis*: co-appearance of characters in Les Miserables novel by Victor Hugo.
- *facebook*: friends list of Facebook users. The data was collected from survey participants using an application developed by Stanford university researchers. The dataset included node features (profiles), circles, and ego networks. The data has been anonymized by replacing internal ids by a new value.
- *enron*: a well-known dataset that contains email communication of the senior management in a large company. The dataset contains twenty years of emails from 1980.
- *EUall*: an e-mail network from an EU research institution.
- *dblp*: a co-authorship network among computer science researchers.
- *youtube*: Youtube users network.
- *collab*: A collaboration network within a research institute. We add an edge between the researchers if they have a joint paper in DBLP.
- *hen dominance*: A hen dominance network[1]. The dataset contains the network of 32 hens represented as nodes and edge represents dominance of left hen over the right hen.

The first three datasets, *karate*, *dolphins*, and *lesmis* are obtained from UCIrvine Network Data Repository[2]. The remaining datasets are obtained from Standford SNAP Repository[3].

Our emphasis is to compare the performance of Greedy and Iterative, as well as to compare the results with RolX [17][4]. All datasets and software used in the experimental evaluation are publicly available[5].

For each dataset, except for *synth1*, *synth2*, *collab*, and *hen dominance* we apply Perfect, Greedy, and Iterative. For the first 3 smallest graphs, we mine with setting $k = 4$ roles. For datasets *facebook*, *enron*, *EUall*, *dblp*, *youtube*, we set $k = 10$ since the datasets are large and often have numerous types of people interacting in the graphs. When applying

[1] `http://moreno.ss.uci.edu/data.html`

[2] `http://networkdata.ics.uci.edu/index.php`

[3] `http://snap.stanford.edu/data`

[4] We use an implementation by Circulo project, `https://github.com/lab41/circulo`.

[5] `http://research.ics.aalto.fi/dmg/roles.zip`

Table 5.1: Basic characteristics of the datasets. The second last column depicts the number of unique vertex degrees, the last column depicts the average degree.

Name	dir.	$\|V\|$	$\|E\|$	# deg	$\overline{\deg}$
karate	no	34	78	11	4.59
dolphins	no	62	159	12	5.13
lesmis	no	77	254	18	6.59
facebook	no	4 039	88 234	227	43.69
enron	no	36 692	183 831	334	10.02
EUall	yes	265 214	420 045	311	3.17
dblp	no	317 080	1 049 866	199	6.62
youtube	no	1 134 890	2 987 624	978	5.27
synth1	no	10	19	3	3.8
synth2	no	100	3270	2	32.70
collab	no	132	237	14	3.59

GREEDY, we need to provide a role assignment as a seed which is then utilized by GREEDY to improve the result. We consider 4 different variants, mainly by setting an initial solution,

1. ONE: every vertex is assigned the same role, namely 0.
2. DEG: vertices are sorted based on degree, and split into k nearly equal-size clusters and assigned a role. If a node is in cluster i, then its role is i.
3. RND: each vertex is assigned a random role between 0 and $k-1$.
4. I+G: The results of ITERATIVE using k-means clustering after one iteration is used as seed.

The experiments conducted on *synth1*, *synth2* and *collab* are described in the next section.

To speed-up the computation of GREEDY, we implement the following heuristic: if the role of a vertex has not changed for a threshold number of iterations, say for 5 times, we no longer test the vertex, assuming that it will not have a better role. However, when we have converged, we start all over by testing every vertex again. We stop, when no gain is possible, even if we consider every vertex.

Evaluation measures

The performance of any models is often evaluated using some standard measures. In our case, we use measures that compare results given by two models and see how correlated the results are. Hence some result comparison measures are used. We will briefly describe them.

Adjusted Rand Index Rand index [31] is a measure in statistics, often used in data clustering to measure the similarity between two clusters. Given a set of n elements $S = \{o_1, o_2, \ldots, o_n\}$ and two paritions of S to compare, $X = \{X_1, X_2, \ldots, X_r\}$, a partition of S into r subsets and $Y = \{Y_1, Y_2, \ldots, Y_s\}$, a partition of S into s subsets, the rand index R is:

$$R = \frac{a+b}{a+b+c+d} = \frac{a+b}{\binom{n}{2}}$$

Here a is the number of pair of elements in S that are in the same set X and in the same set Y, b is the number of pair of elements that are in different sets in X and in different sets in Y, c is the number of pair of elements that are in same set in X and in different sets in Y, and d is the number of pair of elements that are in different sets in X and in same set in Y. In simple form, $a + b$ can be seen as the number of agreements and $c + d$ as the number of disagreements between X and Y. The rand index value lies between 0 and 1. When the partitions agree perfectly, the rand index value is 1. If they totally disagree, the rand index value is 0.

Adjusted rand index [20] is a variant of rand index that adjusts the comparison of clusters to maximize the similarity as much as possible. The formula is

$$AdjustedIndex = \frac{Index - ExpectedIndex}{MaxIndex - ExpectedIndex}$$

Hungarian method Hungarian method [24] is a combinatorial optimization algorithm that is used for solving assignment problem. It takes a non-negative $n * n$ matrix, made by a combination of two assignment results, and returns the best cluster combination results with minimal cost.Once the best possible combination of clusters is calculated, the differences in the result are calculated as hungarian distance value.

Kendall-τ distance The Kendall-τ [21] rank distance is a metric that counts the number of disagreements between two lists. The greater the distance, the more dissimliar the two lists are. Let L_1 and L_2 be two lists, then

Kendall-τ $K(\tau_1, \tau_2)$ is calculated as

$$|\{(i,j) : i < j, (\tau_1(i) < \tau_1(j) \wedge \tau_2(i) > \tau_2(j)) \vee (\tau_1(i) > \tau_1(j) \wedge \tau_2(i) < \tau_2(j))\}|.$$

Here $\tau_1(i)$ and $\tau_2(i)$ are the result at position i in L_1 and L_2 respectively. For example, the Kendall tau distance between 0, 3, 1, 6, 2, 5, 4 and 1, 0, 3, 6, 4, 2, 5 is 4 because the pairs 0-1, 3-1, 2-4, 5-4 are in different order in the two lists, but all other pairs are in the same order. The value is then normalized to be between 0 and 1.

All the above-mentioned measures are used in comparing the results obtained from PERFECT, GREEDY, and ITERATIVE and their variants.

Synthetic data performance

In order to test the effectiveness of PERFECT, we use *synth1*, a perfect graph with distinct roles, and we want to see whether all our methods discover the same roles discovered by PERFECT. PERFECT identifies that there are 3 perfect role assignments which match with a unique degree. When other methods are executed to assign 3 roles, the final score is 0, i.e., perfect assignment. The resultant graph is shown in Figure 5.1.

We also test GREEDY and ITERATIVE on *synth2* to study how well we can discover the underlying structure. In Figure 5.2, we plot the adjusted Rand index and hungarian distance for each method as a function of noise level. The shown numbers are averages over 100 repetitions. As expected the Rand index and hungarian distance generally decreases as the noise level increases: at $p = 0.5$, the graph is completely random and there is no structure left to discover. DEG, I+G, and ITR clearly outperforms ONE and RND, this implies that a good starting point is required for GREEDY. Interestingly, ITR performs worse with small levels of noise but outperforms GREEDY variants once the noise level increase.

In Figure 5.3, we plot the adjusted Rand index for I+G with ITR and Ground truth initialization. We also use different samples such as 1, 10, and 100. From the plot, we can see that ground truth initialization has higher Rand index values. With more samples, the rand index score is better for ground truth initialization. The reason for ITR initialization to perform poorly is because of results provided by ITR. They tend to be far away from the ground truth and reaches the local minima. Hence, with more sampling, ITR performs poorly as opposite to ground truth initialization. To summarize, the methods need a good starting point to achieve better results.

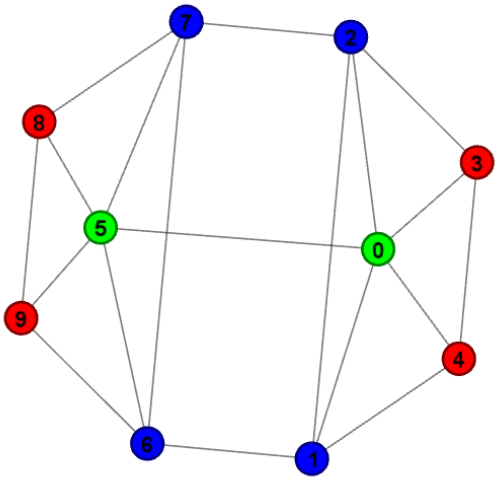


Figure 5.1: Role assignment for *synth1* data. There are 3 distinct roles, denoted by color of the node, that correspond to distinct degrees.

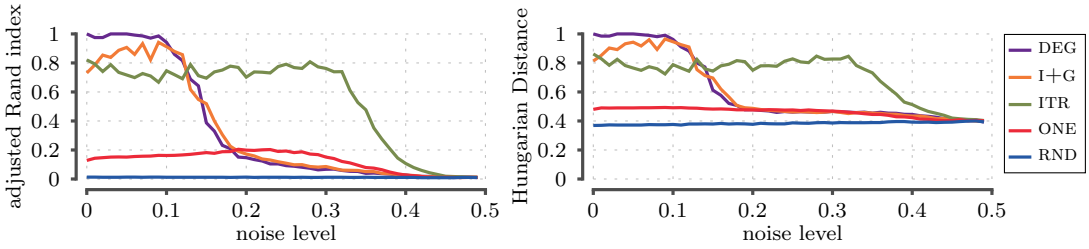


Figure 5.2: Adjusted Rand index and Hungarian Distance between different methods and the ground truth as a function of noise for the synthetic dataset. Larger numbers indicate stronger agreement.

Perfect assignments performance

Next, we consider the assignments given by PERFECT, given in Table 5.2.

We see that the number of roles needed to obtain a perfect solution is typically large: with the exception of *EUall*, we need at least half of the

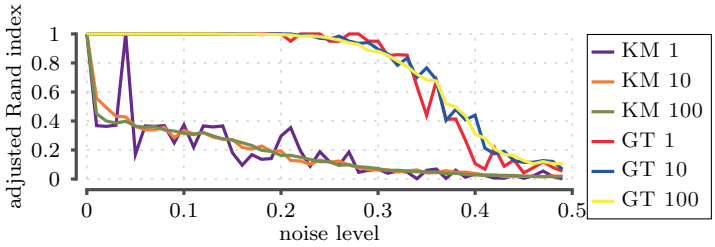


Figure 5.3: Adjusted Rand index between different initializations, samples, and the ground truth as a function of noise for the synthetic dataset. Larger numbers indicate stronger agreement.

Table 5.2: Role assignments discovered by PERFECT. k stands for the number of discovered roles while # deg stands for the number of distinct degrees.

Name	k	iter.	time	$k/\lvert V\rvert$	$k/\#\,\deg$
karate	27	2	1ms	0.79	2.45
dolphins	60	3	2ms	0.96	5
lesmis	56	5	4ms	0.73	3.11
facebook	3 872	5	0.9s	0.96	17.06
enron	20 618	23	11s	0.56	61.73
EUall	20 138	4	9s	0.08	64.76
dblp	233 466	6	1m4s	0.74	1173.2
youtube	684 010	7	3m47s	0.61	699.39

number of vertices. The number of roles is higher than the number of unique degrees, and the ratio increases for large graphs; these graphs have more ways of forcing vertices to have unique roles. The algorithm is practical even for large graphs as the computational complexity $\mathcal{O}(nm \log n)$ given by Proposition 8 is fairly pessimistic: only a few iterations are needed for convergence, and each iteration requires only $\mathcal{O}(m \log n)$ time.

Performance of greedy and iterative algorithms

Our next step is to compare the performance of GREEDY and ITERATIVE. In Table 5.3 we present the costs obtained by each algorithm, normalized by the cost of a trivial role assignment, where each vertex is assigned the same role.

We see that the best scores are obtained by I+G, that is, GREEDY initialized by ITERATIVE. Curiously enough, ITERATIVE alone performs the worst.

Table 5.3: Costs of role assignments, normalized by $c(r')$, where $r'(v) = 0$ for every v. Lower values are better. The parameter k stands for the number of roles. ITR depicts ITERATIVE, while the remaining columns depicts GREEDY with different initializations.

Name	k	ITR	DEG	ONE	RND	I+G
karate	4	0.103	0.097	0.141	0.125	0.089
dolphins	4	0.306	0.253	0.219	0.255	0.213
lesmis	4	0.142	0.124	0.121	0.133	0.118
facebook	10	0.056	0.043	0.043	0.043	0.039
enron	10	0.064	0.021	0.019	0.019	0.019
EUall	10	0.097	0.029	0.024	0.035	0.028
dblp	10	0.178	0.059	0.065	0.059	0.054
youtube	10	0.202	0.029	0.029	0.029	0.029

These results hint that the search space is highly non-trivial, containing a plethora of local minima. This is further supported by Table 5.4, where we report adjusted Rand index.[6] The index implies that while the obtained results all correlate positively they do differ. This puts extra emphasis on a good initialization of GREEDY.

Table 5.4: Adjusted Rand indices between different initializations of GREEDY.

Name	DEG/ONE	DEG/RND	DEG/I+G	ONE/RND	ONE/I+G	RND/I+G
karate	0.104	0.116	0.485	0.107	0.105	0.222
dolphins	0.419	0.134	0.441	0.181	0.374	0.163
lesmis	0.437	0.449	0.453	0.192	0.603	0.283
facebook	0.389	0.385	0.356	0.591	0.521	0.535
enron	0.224	0.135	0.301	0.157	0.232	0.135
EUall	0.421	0.272	0.282	0.305	0.365	0.218
dblp	0.291	0.411	0.427	0.226	0.219	0.307

The left plot in Figure 5.4 shows the cost as a function of k for *facebook* dataset. Naturally, the cost decreases as we increase k. The change is more rapid for small values of k, and evens out for large values.

[6]Value 1 corresponds to a complete agreement, while 0 implies that the assignments are independent.

Let us now study how well the role assignment correlates with vertex degree. Since the roles are symbolic, we sort the roles based on average degree, and compared the roles and degrees using Kendall-τ.[7] We see from the results given in Table 5.5 that there is a significant correlation between the degrees and the role assignment. Naturally, this is partly due to how we sort the roles. However, there are some subtle differences. The coefficients depend on the dataset, for example, *EUall* obtains one of the lowest values. There is a clear difference between ITERATIVE and GREEDY: the former producing ranks with weak or almost no correlation with the degree. This advances further the notion that ITERATIVE gets stuck in a local minimum, and should not be used alone.

Table 5.5: Kendall-τ statistics between role assignments and vertex degree. Roles are sorted based on their average degree. ITR depicts ITERATIVE, while the remaining columns depict GREEDY with different initializations.

Name	ITR	DEG	ONE	RND	I+G
karate	0.449	0.794	0.501	0.636	0.636
dolphins	0.123	0.661	0.672	0.598	0.744
lesmis	0.228	0.711	0.729	0.679	0.734
facebook	0.052	0.707	0.714	0.741	0.739
enron	0.007	0.467	0.474	0.459	0.467
EUall	0.003	0.327	0.203	0.275	0.352
dblp	0.008	0.596	0.559	0.564	0.536
youtube	0.004	0.467	0.454	0.404	0.465

Running time Let us now consider the computational complexity of the algorithms. We present the number of iterations needed for convergence in Table 5.6, and the running times in Table 5.7. The number of required iterations is modest, especially when compared to the size of the input graph. The running times are manageable: we should point out that we implemented GREEDY using Python, an implementation with a more efficient programming platform should make the algorithm more user-friendly, especially for large graphs. As expected, the running times increase as we increase the number of roles; see the right plot in Figure 5.4.

[7] We use the b variant to accommodate the ties.

Table 5.6: Total number of iterations required for convergence. ITR depicts ITERATIVE, while the remaining columns depicts GREEDY with different initializations.

Name	ITR	DEG	ONE	RND	I+G
karate	2	2	5	15	5
dolphins	2	3	5	12	5
lesmis	2	14	16	13	5
facebook	2	85	92	131	73
enron	3	180	220	215	124
EUall	3	719	404	2921	122
dblp	3	17	41	56	53
youtube	4	462	413	471	552

Table 5.7: Evaluation time of algorithms. ITR depicts ITERATIVE, while the remaining columns depict GREEDY with different initializations.

Name	ITR	DEG	ONE	RND	I+G
karate	2ms	5ms	14ms	19ms	14ms
dolphins	4ms	14ms	22ms	32ms	29ms
lesmis	5ms	46ms	48ms	29ms	36ms
facebook	0.4s	39s	41s	53s	39s
enron	1.9s	7m6s	9m4s	9m5s	7m40s
EUall	21s	49m3s	21m4s	1h45m	10m22s
dblp	21s	1h21m	22m3s	41m6s	29m57s
youtube	87s	12h10m	7h30m	16h10m	14h52m

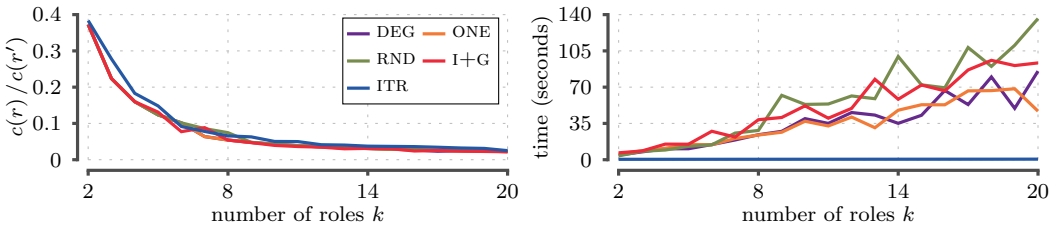


Figure 5.4: The normalized cost and the execution time for *facebook* dataset. The cost is normalized by $c(r')$ such that $r'(v) = 0$, for every v, that is, r' assigns the same role to every vertex.

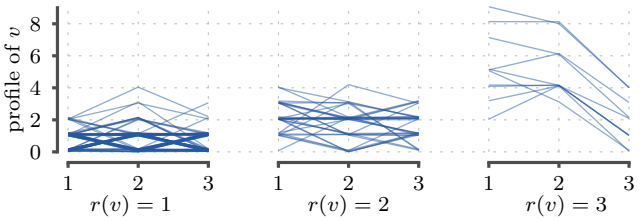


Figure 5.5: Visualization of profiles for *collab* network, obtained using DEG with $k = 3$. Each line represents the number of neighbors of a vertex v as a function of the neighbor role. The three groupings are based on the role of v.

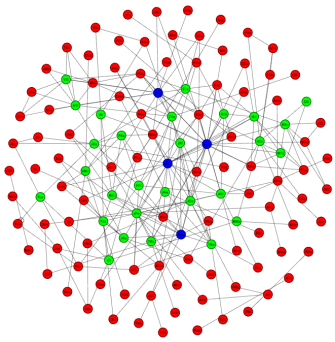
Case study of the collaboration network

Next, we consider collaboration network *collab*. Here we apply DEG with $k = 3$. The role assignments of the network, as well as ego networks of some authors, is shown in Figure 5.6. The blue colored researchers have collaboration with many researchers and the red colored researchers have few collaborators. Finally, the green colored researchers have a higher number of researchers than red colored ones but lower than blue colored researchers.

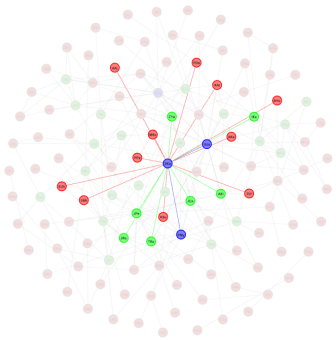
The obtained profiles are visualized in Figure 5.5. The obtained centroids are $\mathbf{c}_1 = (0.65, 0.76, 0.58)$, $\mathbf{c}_2 = (2.06, 1.7, 1.55)$, $\mathbf{c}_3 = (5.2, 5.1, 1.8)$, that is, the researchers with the 3rd role have many co-authors while the researchers with the 1st role have with limited number of co-authors. Finally, the researchers with the middle role are somewhere between the two classes. To assess the results obtained on this dataset, we compare the roles discovered by our algorithm with a partitioning obtained by the job title of the researchers. We use three classes: professors, senior researchers and staff, and Ph.D. students and junior post-docs. The adjusted Rand index between the two partitions is 0.387.[8] We should point that the seniority of a researcher is not always reflected in the collaboration graph: there are many senior researchers with few collaborators.

For understanding purpose, the snapshot of the network and egonets of some the authors is shown in Figure 5.6. As mentioned earlier, the nodes with blue colors are collaborating with many other authors. It is also visible in egonet of Samuel Kaski. Though Nikolaj has fewer collaborators than Aris, he is still assigned the role of Aris as their profiles are closer to each other.

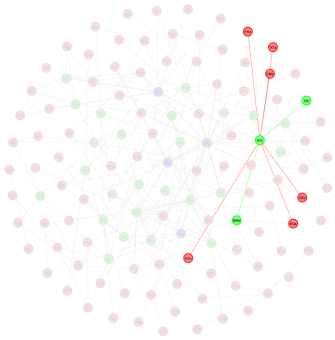
[8]Value 1 means complete agreement, while 0 means that the partitions are independent.



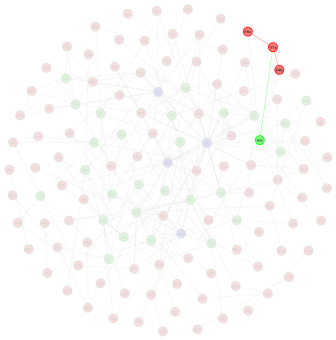
(a) HIIT network with roles



(b) Egonet of Samuel Kaski



(c) Egonet of Aristides Gionis



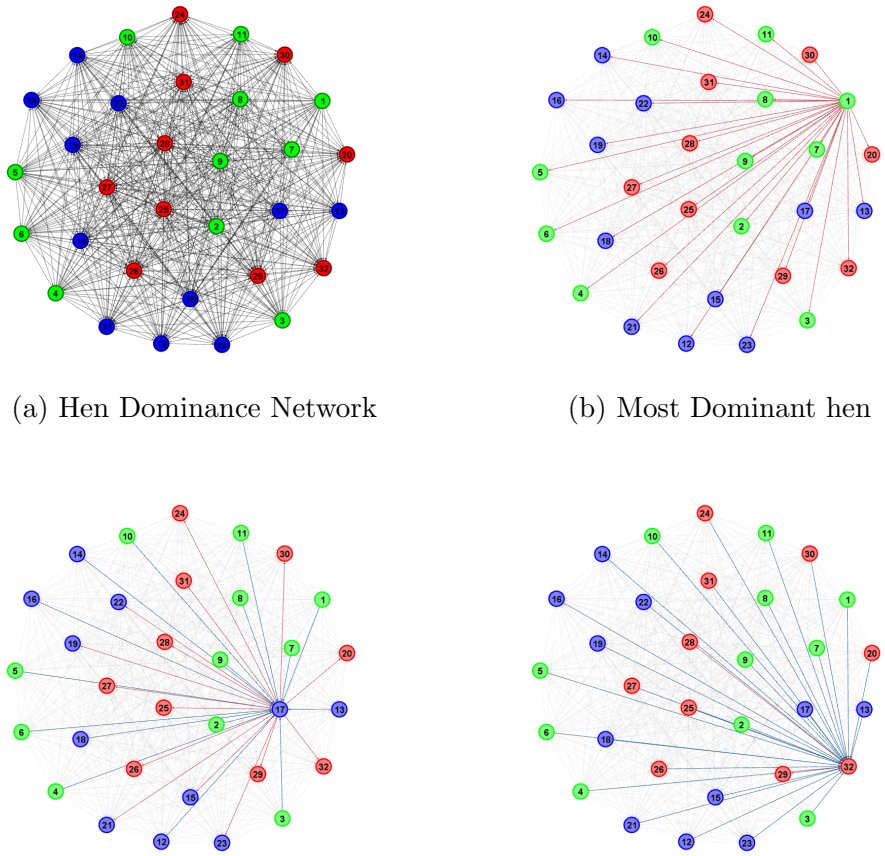
(d) Egonet of Nikolaj Tatti

Figure 5.6: HIIT Network and some highlighted authors' network. The selected author is the center of the egonet. Egonet of Samuel Kaski contains authors from all three roles whereas that of Aris contains mostly mid and low level researchers and Nikolaj's egonet contains low level researchers as collaborators. The profile plays vital role in role assignment, and Nikolaj has less degree than Aris but is assigned same role.

Case study of hen dominance network

We are also interested in testing our model in other real world networks that are interesting and unique. Hence, we use *hen dominance* for testing. Upon executing RolePro on the dataset, the roles assigned were analysed. The snapshots of the dominance network and some of the highlighted hens'

dominance is shown in Figure 5.7.



(a) Hen Dominance Network

(b) Most Dominant hen

(c) Middle level hen

(d) Least Dominant hen

Figure 5.7: Roles in Hen Dominance Network. Three roles are assigned to the hens. Most dominant hens are assigned green color, least dominant ones are assigned red ones, and middle level hens are assigned blue color. Hen numbered 1 is most dominant in the group and hen numbered 32 is the least dominant in the group.

Two colors are used on the edges for highlighting the dominance nature. Red color denotes the selected center node dominates the adjacent nodes. Blue color denotes the selected center is dominated by its adjacent nodes. Moreover, nodes of dominant hens are assigned green color, and nodes of least dominant hens are assigned red color and middle-level ones are assigned a blue color. Hen numbered 1 is the most dominant hen as it dominates all the other hens and hen numbered 32 is dominated by all hens. Middle-level

hens dominate some hens and are dominated by other hens. After assigning roles to hens, one can save the least dominant hens from being targeted by other hens by isolation.

Comparison with RolX

As a final step, we compare the obtained roles with RolX. In Table 5.8, we present normalized costs of RolX and R+G: Greedy initialized with RolX. We also present adjusted Rand indices of RolX versus the greedy variants. The implementation we use for RolX is not able to process *youtube* dataset due to memory consumption.

Table 5.8: Comparison of RolX and Greedy. Columns 2–3 depict obtained costs, normalized by $c(r')$, where $r'(v) = 0$ for every v. The remaining columns depict adjusted Rand index of RolX versus Greedy with different initializations.

	$c(r)\,/\,c(r')$		Adj. Rand vs. RolX			
Name	RolX	R+G	DEG	ONE	RND	I+G
karate	0.217	0.124	0.292	0.068	0.129	0.259
dolphins	0.457	0.302	0.187	0.207	0.236	0.273
lesmis	0.305	0.161	0.298	0.358	0.135	0.336
facebook	0.285	0.056	0.079	0.108	0.111	0.073
enron	0.467	0.019	0.058	0.055	0.053	0.054
EUall	0.438	0.029	0.251	0.259	0.237	0.281
dblp	0.509	0.061	0.655	0.258	0.349	0.359

We first observe that role assignments returned by RolX have a high cost. This is a natural result since RolX does not optimize our objective. However, when we use RolX as a seed for Greedy, the obtained rankings have a low score for large graphs. Positive Rand indices imply that the assignments of RolX and Greedy do correlate, but also differ.

As a sanity check, we construct a classifier predicting the role of a vertex based on its features. For Iterative and Greedy we use the profiles as features, and for RolX we use the feature vectors. We use decision tree classifier with 10-fold cross-validation, and the accuracy results are given in Table 5.9. We see that all methods perform well, note that 10 classes for large datasets and 4 classes for small datasets. We observe that the accuracy is generally higher for Iterative and RolX than for Greedy.

Table 5.9: Classification accuracy statistics of RolX, Iterative, and Greedy. itr depicts Iterative, while the remaining columns depicts Greedy with different initializations.

Name	RolX	itr	deg	one	rnd	i+g
karate	0.735	0.912	0.647	0.676	0.412	0.823
dolphins	0.613	0.903	0.662	0.661	0.565	0.662
lesmis	0.714	0.948	0.779	0.741	0.831	0.741
facebook	0.887	0.884	0.656	0.691	0.673	0.675
enron	0.847	0.876	0.589	0.584	0.579	0.567
EUall	0.896	0.875	0.669	0.833	0.689	0.791
dblp	0.881	0.812	0.906	0.623	0.667	0.658

Chapter 6

Conclusions

In this thesis, we propose a new type of role discovery optimization problem. We propose that the vertices should have the same role if their profiles, role counts of neighbors, are similar.

From a technical point, our method is different than feature-based techniques because our features are in fact roles of neighbors. Hence, a two-step approach—(*i*) construct features and (*ii*) cluster roles from features—does not work. This intricate dependency makes the optimization problem difficult: we show that the problem is **NP**-hard, and cannot be even approximated if we fix the centroids for the roles.

On the positive side, we show that we can discover the perfect, zero-cost, solution with a minimal number of roles efficiently in polynomial time. When the number of roles is fixed, we propose two simple natural heuristics: iterative optimization and a hill-climbing algorithm.

Interestingly enough, we do not directly use any network-based feature when comparing vertices. Instead, we are only interested in role counts. Our logic is that fundamentally different ego-networks for vertices say, u and v, should result in different role counts which should imply that u and v are different. Nevertheless, combining our approach with other feature-based role discovery methods provides a potentially fruitful direction for future work.

Bibliography

[1] Abdi, H., and Williams, L. J. Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics 2*, 4 (2010), 433–459.

[2] Airoldi, E. M., Blei, D. M., Fienberg, S. E., and Xing, E. P. Mixed membership stochastic blockmodels. *Journal of Machine Learning Research 9*, Sep (2008), 1981–2014.

[3] Beisner, B. A., Jackson, M. E., Cameron, A. N., and McCowan, B. Detecting instability in animal social networks: Genetic fragmentation is associated with social instability in rhesus macaques. *PLoS ONE 6*, 1 (2011), e16365.

[4] Bezdek, J. C., Ehrlich, R., and Full, W. Fcm: The fuzzy c-means clustering algorithm. *Computers & Geosciences 10*, 2-3 (1984), 191–203.

[5] Biddle, B. J. Recent development in role theory. *Annual review of sociology* (1986), 67–92.

[6] Boriah, S., Chandola, V., and Kumar, V. Similarity measures for categorical data: A comparative evaluation. *red 30*, 2 (2008), 3.

[7] Brandes, U., and Lerner, J. Structural similarity: Spectral methods for relaxed blockmodeling. *Journal of classification 27*, 3 (2010), 279–306.

[8] Breiger, R. L., Boorman, S. A., and Arabie, P. An algorithm for clustering relational data with applications to social network analysis and comparison with multidimensional scaling. *Journal of mathematical psychology 12*, 3 (1975), 328–383.

[9] Burt, R. S. Positions in networks. *Social forces 55*, 1 (1976), 93–122.

[10] Chang, J., and Blei, D. M. Relational topic models for document networks. In *AIStats* (2009), vol. 9, pp. 81–88.

[11] Danilevsky, M., Wang, C., Desai, N., and Han, J. Entity role discovery in hierarchical topical communities. In *ACM SIGKDD International Workshop on Mining Data Semantics and Heterogeneous Information Networks* (2013), pp. 1–8.

[12] De Lathauwer, L. A survey of tensor methods. In *2009 IEEE International Symposium on Circuits and Systems* (2009), IEEE, pp. 2773–2776.

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

[14] Friedlander, M. P., and Hatz, K. Computing non-negative tensor factorizations. *Optimisation Methods and Software 23*, 4 (2008), 631–647.

[15] Gilpin, S., Eliassi-Rad, T., and Davidson, I. Guided learning for role discovery (GLRD): framework, algorithms, and applications. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining* (2013), ACM, pp. 113–121.

[16] Golub, G. H., and Reinsch, C. Singular value decomposition and least squares solutions. *Numerische mathematik 14*, 5 (1970), 403–420.

[17] Henderson, K., Gallagher, B., Eliassi-Rad, T., Tong, H., Basu, S., Akoglu, L., Koutra, D., Faloutsos, C., and Li, L. Rolx: structural role extraction & mining in large graphs. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining* (2012), pp. 1231–1239.

[18] Henderson, K., Gallagher, B., Li, L., Akoglu, L., Eliassi-Rad, T., Tong, H., and Faloutsos, C. It's who you know: graph mining using recursive structural features. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining* (2011), ACM, pp. 663–671.

[19] Holland, P. W., and Leinhardt, S. An exponential family of probability distributions for directed graphs. *Journal of the american Statistical association 76*, 373 (1981), 33–50.

[20] Hubert, L., and Arabie, P. Comparing partitions. *Journal of classification 2*, 1 (1985), 193–218.

[21] Kendall, M. G. A new measure of rank correlation. *Biometrika 30*, 1/2 (1938), 81–93.

[22] Kruskal, J. B. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika 29*, 1 (1964), 1–27.

[23] Kruskal, J. B. Nonmetric multidimensional scaling: a numerical method. *Psychometrika 29*, 2 (1964), 115–129.

[24] Kuhn, H. W. The hungarian method for the assignment problem. *Naval research logistics quarterly 2*, 1-2 (1955), 83–97.

[25] Lichtenwalter, R. N., Lussier, J. T., and Chawla, N. V. New perspectives and methods in link prediction. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining* (2010), ACM, pp. 243–252.

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

[27] Ma, H., Yang, H., Lyu, M. R., and King, I. Sorec: social recommendation using probabilistic matrix factorization. In *Proceedings of the 17th ACM conference on Information and knowledge management* (2008), ACM, pp. 931–940.

[28] McDowell, L. K., Gupta, K. M., and Aha, D. W. Cautious collective classification. *Journal of Machine Learning Research 10*, Dec (2009), 2777–2836.

[29] McKay, B. D. In *Combinatorial Mathematics: Proceedings of the International Conference on Combinatorial Theory* (1978), D. A. Holton and J. Seberry, Eds., pp. 223–232.

[30] Neville, J., and Jensen, D. Iterative classification in relational data. In *Proc. AAAI-2000 Workshop on Learning Statistical Models from Relational Data* (2000), pp. 13–20.

[31] Rand, W. M. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association 66*, 336 (1971), 846–850.

[32] Rasmussen, C. E. The infinite gaussian mixture model. In *NIPS* (1999), vol. 12, pp. 554–560.

[33] Rissanen, J. Modeling by shortest data description. *Automatica 14*, 5 (1978), 465–471.

[34] Robins, G., Pattison, P., Kalish, Y., and Lusher, D. An introduction to exponential random graph (p*) models for social networks. *Social networks 29*, 2 (2007), 173–191.

[35] Rossi, R., Fahmy, S., and Talukder, N. A multi-level approach for evaluating internet topology generators. In *IFIP Networking Conference, 2013* (2013), IEEE, pp. 1–9.

[36] Rossi, R., and Neville, J. Modeling the evolution of discussion topics and communication to improve relational classification. In *Proceedings of the First Workshop on Social Media Analytics* (2010), ACM, pp. 89–97.

[37] Rossi, R. A., and Ahmed, N. K. Role discovery in networks. *Knowledge and Data Engineering, IEEE Transactions on 27*, 4 (2015), 1112–1131.

[38] Rossi, R. A., Gallagher, B., Neville, J., and Henderson, K. Modeling dynamic behavior in large evolving graphs. In *Proceedings of the 6th ACM International Conference on Web Search and Data Mining (WSDM)* (2013), pp. 667–676.

[39] Ruan, Y., and Parthasarathy, S. Simultaneous detection of communities and roles from large networks. In *Proceedings of the second ACM conference on Online social networks* (2014), ACM, pp. 203–214.

[40] Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. Application of dimensionality reduction in recommender system-a case study. Tech. rep., DTIC Document, 2000.

[41] Schölkopf, B., Smola, A., and Müller, K.-R. Kernel principal component analysis. In *International Conference on Artificial Neural Networks* (1997), Springer, pp. 583–588.

[42] Wang, Y.-X., and Zhang, Y.-J. Nonnegative matrix factorization: A comprehensive review. *IEEE Transactions on Knowledge and Data Engineering 25*, 6 (2013), 1336–1353.

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

[44] Yang, Y., and Pei, J. In-network neighborhood-based node similarity measure: A unified parametric model. *arXiv preprint arXiv:1510.03814* (2015).

[45] Zhao, Y., Wang, G., Yu, P. S., Liu, S., and Zhang, S. Inferring social roles and statuses in social networks. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining* (2013), ACM, pp. 695–703.