A combinatorial approach to role discovery

Albert Arockiasamy Aristides Gionis Nikolaj Tatti
HIIT, Aalto University, Espoo, Finland, firstname.lastname@aalto.fi

***Abstract*—We provide a new formulation for the problem of role discovery in graphs. Our definition is structural: 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.

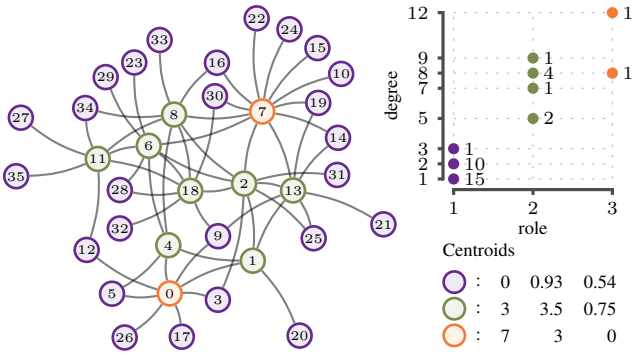

Figure 1: Groom network of Rhesus Macaques [12], Three roles are assigned by GREEDY initialized by DEG. The scatter plot shows the degree of a vertex as a function of its role.

I. INTRODUCTION

Modeling interconnected entities as graphs (or networks) allows us study the global structure and function of a system, instead of looking at single entities in isolation. 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. *Role discovery* has emerged as an important graph-mining task [1]–[7], 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 person in a social network may provide cues for understanding the social behavior of the person in relation to 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 [8]. Rossi and Ahmed [4] 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.

Various 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 rely on identifying structural or automorphic equivalence classes [9], [10], while newer methods represent vertices by feature vectors and assign to the same role vertices with similar feature vectors [3], [4], [7], [11].

In this paper we present a new approach to role discovery, inspired by the definition of *equitable partition* of vertices. As the original definition is too strict to be of use in real-world datasets, we introduce 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 equitability. Given a target number of roles k we ask to find the role assignment that minimizes our objective function.

The proposed objective function is based on creating a *profile* for each 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. 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, not 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 smallest number of roles k can be done efficiently in polynomial time. We further propose two natural heuristic algorithms for minimizing the cost function 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.

II. Related work

Role-discovery methods can be broadly classified into three categories [4]: (i) graph-based, (ii) feature-based, and (iii) hybrid. 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.

In automorphic equivalence (see, for example, [13]), two vertices u and v are equivalent if there is graph automorphism mapping u to v. Furthermore, the vertices are structurally equivalent [10], if the automorphism does not alter the remaining vertices. A more relaxed definition of equivalence is regular equivalence [9], where vertices are equivalent if they have equivalent neighbors, ignoring any multiplicities.

Blockmodelling can be viewed as a role assignment task. Here the idea is to model the edge appearance between two vertices u and v based on their roles. The roles are viewed as latent variables, and are learned using standard statistical optimization techniques [14]. For more details on blockmodels, see the survey of Goldenberg et al. [15].

Feature-based role discovery is a more modern approach that relies on representing each node in the graph by a feature vector and assigning to the same role nodes with similar feature vectors [1], [3], [4], [7], [11]. 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 behavior or node attributes. Once feature vectors are constructed, the assignment 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. Hybrid role discovery methods combine both methods. For example, using learned blockmodels as features [4].

The typical key component in role discovery is how to measure similarity between two vertices. The simplest way to do this is by computing a distance between the neighbors, such as, cosine similarity or Pearson coefficient of common neighbors. As a more intricate example, we can also based vertex similarity on features of neighboring vertices [6] or spectral analysis [16].

We should stress that role discovery has a different goal than community detection: in community detection, we are interested in finding highly connected subsets, whereas in role discovery we want to find vertices that serve similar purpose. Interestingly, Ruan and Parthasarathy [5] proposed mining roles and communities simultaneously.

Finally, a fruitful direction for role discovery is to adapt existing methodology for dynamic settings, where the role may change over time [17], [18]. We leave adapting our approach for dynamic settings as future work.

III. Preliminaries and problem definition

We consider a graph $G = (V, E)$ with $|V| = n$ vertices and $|E| = m$ edges. Our goal is to assign roles to the vertices of the graph G. We assume that the total number of roles k is given. 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. 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,

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

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

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

We can now formulate our our role-mining problem.

Problem 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 by a reduction from the 3D-Matching problem. The proof is given in the full version of the paper.

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

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 r can then be easily computed as the centroid of the profiles of all vertices having the role r.

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.

Problem 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* ROLES-FIXEDCENTROIDS *has a zero-cost solution is an **NP**-complete problem.*

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 5. *Unless **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.

IV. SOLVING PERFECT ROLE ASSIGNMENT EXACTLY

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 equitable partition [19], which can be solved exactly.

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 [19].

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;
```

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.

Finally, we analyze the running time of PERFECT.

Proposition 6. PERFECT *runs in $\mathcal{O}(mn \log n)$ time.*

V. 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.

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

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

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.

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

```
initialize role assignment, r ← r_0;
while changes do
    foreach v ∈ V do
        j* ← arg max_j gain(v, j);
        if gain(v, j*) > 0 then reassign ;
```

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.

Proposition 7. *The inner loop of* GREEDY (**foreach** *loop in Algorithm 2*) *requires time* $\mathcal{O}(k^2n + km)$.

VI. 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 we fix one set of parameters while optimizing the other set.

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), computes roles in iterative fashion. CLUSTER is a standard clustering method.

```
r_0 ← CLUSTER(deg(·), k) ; i ← 0;
while cost decreases do
    r_{i+1} ← CLUSTER(p(·, r_i), k) ; 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.

VII. EXPERIMENTAL EVALUATION

In this section we present our experimental evaluation. Our emphasis is to compare the performance of GREEDY and ITERATIVE, as well as to compare the results with ROLX [3].[1] All datasets and software used in the experimental evaluation are publicly available.[2]

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

[2]http://research.ics.aalto.fi/dmg/

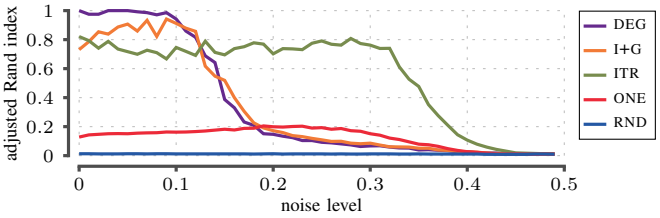

Figure 2: Adjusted Rand index between different methods and the ground truth as a function of noise for the synthetic dataset. Larger numbers indicate stronger agreement.

Experimental setup: We use 10 graphs of different sizes and densities. The first three datasets, *karate*, *dolphins*, and *lesmis* are obtained from UCIrvine Network Data Repository.[3] We create a synthetic dataset, *synth*, with 3 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. We also consider a collaboration network within a research institute;[4] we add an edge between the researchers if they have a joint paper in DBLP. The remaining datasets are obtained from Standford SNAP Repository.[5]

For each dataset, except for *synth* and *collab*, we apply PERFECT, GREEDY, and ITERATIVE. For the 3 smallest graphs, we mine $k = 4$ roles. For the remaining graphs, we set $k = 10$ roles. When applying GREEDY, we need to provide a role assignment as a seed. We consider 4 different variants: (i) ONE, every vertex is assigned the same role, (ii) DEG, vertices are sorted based on degree, and split into equal-size clusters, (iii) RND, roles are assigned randomly, (iv) I+G, where we use the assignment given by ITERATIVE.

To speed-up the computation of GREEDY, we implement the following heuristic: if the role of a vertex has not changed during the 3 last iterations, we no longer test the vertex. 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.

Synthetic data: Our first step is to study how well we can discover the underlying structure in *synth*, the synthetic data. In Figure 2, we plot the adjusted Rand index for each method as a function of noise level. The shown numbers are averages over 100 repetitions. As expected the Rand index 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 outperfoms 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.

Perfect assignments: Next, we consider the assignments given by PERFECT, given in Table I.

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

[4]Helsinki Institute of Information Technology

[5]http://snap.stanford.edu/data

Table I: 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

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 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 6 is fairly pessimistic: only 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 II 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.

Table II: Columns 3–7 show costs of role assignments, normalized by $c(r')$, where $r'(v) = 0$ for every v. Lower values are better. Columns 8–12 show Kendall-τ statistics between role assignments and vertex degree. The parameter k stands for the number of roles. ITR depicts ITERATIVE, while the remaining columns depicts GREEDY with different initializations.

		Costs of role assignments					Kentall-τ vs. vertex degree				
Name	k	ITR	DEG	ONE	RND	I+G	ITR	DEG	ONE	RND	I+G
karate	4	.103	.097	.141	.125	.089	.449	.794	.501	.636	.636
dolphins	4	.306	.253	.219	.255	.213	.123	.661	.672	.598	.744
lesmis	4	.142	.124	.121	.133	.118	.228	.711	.729	.679	.734
facebook	10	.056	.043	.043	.043	.039	.052	.707	.714	.741	.739
enron	10	.064	.021	.019	.019	.019	.007	.467	.474	.459	.467
EUall	10	.097	.029	.024	.035	.028	.003	.327	.203	.275	.352
dblp	10	.178	.059	.065	.059	.054	.008	.596	.559	.564	.536
youtube	10	.202	.029	.029	.029	.029	.004	.467	.454	.404	.465

We see that the best scores are obtained by I+G, that is, GREEDY initialized by ITERATIVE. Curiously enough, ITERATIVE alone performs the worst. These results hint that the search space is highly non-trivial, containing a plethora of local minima. This is further supported by Table III, 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.

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

Table III: 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

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 II that there is 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 local minimum, and should not be used alone.

Running time: Let us now consider the computational complexity of the algorithms. We present the number of iterations needed for convergence and the running times in Table IV. 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.

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

	Number of iterations					Evaluation time				
Name	ITR	DEG	ONE	RND	I+G	ITR	DEG	ONE	RND	I+G
karate	2	2	5	15	5	2ms	5ms	14ms	19ms	14ms
dolphins	2	3	5	12	5	4ms	14ms	22ms	32ms	29ms
lesmis	2	14	16	13	5	5ms	46ms	48ms	29ms	36ms
facebook	2	85	92	131	73	0.4s	39s	41s	53s	39s
enron	3	180	220	215	124	1.9s	7m6s	9m4s	9m5s	7m40s
EUall	3	719	404	2921	122	21s	49m3s	21m4s	1h45m	10m22s
dblp	3	17	41	56	53	21s	1h21m	22m3s	41m6s	29m57s
youtube	4	462	413	471	552	87s	12h10m	7h30m	16h10m	14h52m

Case study of the collaboration network: Next we consider collaboration network *collab*. Here we apply DEG with $k = 3$. 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. The researchers with the middle role are somewhere between the

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

two classes. To assess the obtained results, we compared 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 PhD students and junior postdocs. The adjusted Rand index between the two partitionings is 0.387. We should point that the seniority of a researcher is not always reflected in the collaboration graph: there are many senior reseachers with few collaborators.

Comparison to ROLX: As a final step, we compare the obtained roles with ROLX. In Table V, 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 V: Comparison of ROLX and GREEDY. Columns 2–3 depict obtained costs, normalized by $c(r')$, where $r'(v) = 0$ for every v. Columns 5–8 depict adjusted Rand index of ROLX versus GREEDY with different initializations. The remaining columns show classification accuracy used roles as features.

	$c(r)\,/\,c(r')$		Rand vs. ROLX				Classification accuracy					
Name	ROLX	R+G	DEG	ONE	RND	I+G	ROLX	ITR	DEG	ONE	RND	I+G
karate	.217	.124	.292	.068	.129	.259	.735	.912	.647	.676	.412	.823
dolphins	.457	.302	.187	.207	.236	.273	.613	.903	.662	.661	.565	.662
lesmis	.305	.161	.298	.358	.135	.336	.714	.948	.779	.741	.831	.741
facebook	.285	.056	.079	.108	.111	.073	.887	.884	.656	.691	.673	.675
enron	.467	.019	.058	.055	.053	.054	.847	.876	.589	.584	.579	.567
EUall	.438	.029	.251	.259	.237	.281	.896	.875	.669	.833	.689	.791
dblp	.509	.061	.655	.258	.349	.359	.881	.812	.906	.623	.667	.658

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 V. 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.

VIII. CONCLUDING REMARKS

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

From technical point, our method is different than feature-based techniques because our features are in fact roles of neighbors. This 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 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.

REFERENCES

[1] S. Gilpin, T. Eliassi-Rad, and I. Davidson, "Guided learning for role discovery (GLRD): framework, algorithms, and applications," in *KDD*. ACM, 2013, pp. 113–121.

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

[3] K. Henderson, B. Gallagher, T. Eliassi-Rad, H. Tong, S. Basu, L. Akoglu, D. Koutra, C. Faloutsos, and L. Li, "RolX: structural role extraction & mining in large graphs," in *KDD*, 2012, pp. 1231–1239.

[4] R. A. Rossi and N. K. Ahmed, "Role discovery in networks," *TKDE*, vol. 27, no. 4, pp. 1112–1131, 2015.

[5] Y. Ruan and S. Parthasarathy, "Simultaneous detection of communities and roles from large networks," in *COSN*, 2014, pp. 203–214.

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

[7] Y. Zhao, G. Wang, P. S. Yu, S. Liu, and S. Zhang, "Inferring social roles and statuses in social networks," in *KDD*. ACM, 2013, pp. 695–703.

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

[9] M. G. Everett and S. P. Borgatti, "Regular equivalence: General theory," *Journal of Mathematical Sociology*, vol. 19, no. 1, pp. 29–52, 1994.

[10] F. Lorrain and H. C. White, "Structural equivalence of individuals in social networks," *The Journal of Mathematical Sociology*, vol. 1, no. 1, pp. 49–80, 1971.

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

[12] B. A. Beisner, M. E. Jackson, A. N. Cameron, and B. McCowan, "Detecting instability in animal social networks: Genetic fragmentation is associated with social instability in rhesus macaques," *PLoS ONE*, vol. 6, no. 1, p. e16365, 2011.

[13] R. A. Hanneman and M. Riddle, *Introduction to social network methods*. University of California, 2005, http://www.faculty.ucr.edu/~hanneman/.

[14] A. T. Snijders and K. Nowicki, "Estimation and prediction for stochastic blockmodels for graphs with latent block structure," *Journal of Classification*, vol. 14, no. 1, pp. 75–100, 1997.

[15] A. Goldenberg, A. X. Zheng, S. E. Fienberg, and E. M. Airoldi, "A survey of statistical network models," *Found. Trends Mach. Learn.*, vol. 2, no. 2, pp. 129–233, 2010.

[16] C. E. Tsourakakis, "Toward quantifying vertex similarity in networks," *Internet Mathematics*, vol. 10, no. 3-4, pp. 263–286, 2014.

[17] A. Abnar, M. Takaffoli, R. Rabbany, and O. R. Zaïane, "SSRM: structural social role mining for dynamic social networks," *Social Network Analysis and Mining*, vol. 5, no. 1, pp. 1–18, 2015.

[18] K. Li, S. Guo, N. Du, J. Gao, and A. Zhang, "Learning, analyzing and predicting object roles on dynamic networks," in *ICDM*, 2013, pp. 428–437.

[19] B. D. McKay, "Computing automorphisms and canonical labellings of graphs," in *Combinatorial Mathematics: Proceedings of the International Conference on Combinatorial Theory*, 1978, pp. 223–232.