A Late-Fusion Approach to Community Detection in Attributed Networks

Chang Liu[1], Christine Largeron[2], Osmar R. Zaïane[1(✉)], and Shiva Zamani Gharaghooshi[1]

[1] Alberta Machine Intelligence Institute, University of Alberta, Edmonton, Canada
{chang6,zaiane,zamanigh}@ualberta.ca
[2] Laboratoire Hubert Curien, Université de Lyon, Saint-Etienne, France
christine.largeron@univ-st-etienne.fr

Abstract. The majority of research on community detection in attributed networks follows an "early fusion" approach, in which the structural and attribute information about the network are integrated together as the guide to community detection. In this paper, we propose an approach called *late-fusion*, which looks at this problem from a different perspective. We first exploit the network structure and node attributes separately to produce two different partitionings. Later on, we combine these two sets of communities via a fusion algorithm, where we introduce a parameter for weighting the importance given to each type of information: node connections and attribute values. Extensive experiments on various real and synthetic networks show that our late-fusion approach can improve detection accuracy from using only network structure. Moreover, our approach runs significantly faster than other attributed community detection algorithms including early fusion ones.

Keywords: Community detection · Attributed networks · Late fusion

1 Introduction

In many modern applications, data is represented in the form of relationships between nodes forming a *network*, or interchangeably a *graph*. A typical characteristic of these real networks is the *community structure*, where network nodes can be grouped into densely connected modules called communities. Community identification is an important issue because it can help to understand the network structure and leads to many substantial applications [6]. While traditional community detection methods focus on the network topology where communities can be defined as sets of nodes densely connected internally, recently, increasing attention has been paid to the attributes associated with the nodes in order to take into account homophily effects, and several works have been devoted to community detection in attributed networks. The aim of such process is to obtain a partitioning of the nodes where vertices belonging to the same subgroup are densely connected and homogeneous in terms of attribute values.

M. R. Berthold et al. (Eds.): IDA 2020, LNCS 12080, pp. 300–312, 2020.
https://doi.org/10.1007/978-3-030-44584-3_24

In this paper, we propose a new method designed for community detection in attributed networks, called *late fusion*. This is a two-step approach where we first identify two sets of communities based on the network topology and node attributes respectively, then we merge them together to produce the final partitioning of the network that exhibits the homophily effect, according to which linked nodes are more likely to share the same attribute values. The communities based upon the network topology are obtained by simply applying an existing algorithm such like Louvain [2]. For graphs whose node attributes are numeric, we utilize existing clustering algorithms to get the communities (i.e., clusters) based on node attributes. We extend to binary-attributed graphs by generating a virtual graph from the attribute similarities between the nodes, and performing traditional community detection on the virtual graph. Albeit being simple, extensive experiments have shown that our late-fusion method can be competitive in terms of both accuracy and efficiency when compared against other algorithms. We summarize our main contributions in this work are:

1. A new late-fusion approach to community detection in attributed networks, which allows the use of traditional methods as well as the integration of personal preference or prior knowledge.
2. A novel method to identify communities that reflect attribute similarity for networks with binary attributes.
3. Extensive experiments to validate the proposed method in terms of accuracy and efficiency.

The rest of the paper is organized as follows: In Sect. 2, we provide a brief review of community detection algorithms suited for attributed networks, next we present our late fusion approach in Sect. 3. Experiments to illustrate the effectiveness of the proposed method are detailed in Sect. 4. Finally, we summarize our work and point out several future directions in Sect. 5.

2 Related Work

How to incorporate the node attribute information into the process of network community detection has been studied for a long time. One of the early ideas is to transform attribute similarities into edge weights. For example, [13] proposes *matching coefficient* which is the count of shared attributes between two connected nodes in a network; [15] extends the matching coefficient to networks with numeric node attributes; [4] defines edge weights based on self-organizing maps. A drawback of these methods is that new edge weights are only applicable to edges already existed, hence the attribute information is not fully utilized. To overcome this issue, a different approach is to *augment* the original graph by adding virtual edges and/or nodes based on node attribute values. For instance, [14] generates content edges based on the cosine similarity between node attribute vectors, in graphs where nodes are textual documents and the corresponding attribute vector is the TF-IDF vector describing their content. The kNN-enhance algorithm [9] adds directed virtual edges from a node to one

of its k-nearest neighbors if their attributes are similar. The SA-Clustering [17] adds both virtual nodes and edges to the original graph, where the virtual nodes represent binary-valued attributes, and the virtual edges connect the real nodes to the virtual nodes representing the attributes that the real nodes own.

Another class of methods is inspired by the modularity measure. These methods incorporate attribute information into an optimization objective like the modularity. [5] injects an attribute based similarity measure into the modularity function; [1] combines the gain in the modularity with multiple common users' attributes as an integrated objective; I-Louvain algorithm [3] proposes inertia-based modularity to describe the similarity between nodes with numeric attributes, and adds the inertia-based modularity to the original modularity formula to form the new optimization objective.

With the wide spreading of deep learning, network representation learning and node embedding (e.g. [8]) motivated new solutions. [12] proposes an embedding based community detection algorithm that applies representation learning of graphs to learn a feature representation of a network structure, which is combined with node attributes to form a cost function. Minimizing it, the optimal community membership matrix is obtained.

Probabilistic models can be used to depict the relationship between node connections, attributes, and community membership. The task of community detection is thus converted to inferring the community assignment of the nodes. A representative of this kind is the CESNA algorithm [16], which builds a generative graphical model for inferring the community memberships.

Whereas the majority of the previous methods exploit simultaneously both types of information, we propose the late-fusion approach that combines two sets of communities obtained separately and independently from the network structure and node attributes via a fusion algorithms.

3 The Late-Fusion Method

Given an attributed network $G = (V, E, A)$, with V being the set of m nodes, E the set of n edges, and A an $m \times r$ attribute matrix describing the attribute values of the nodes with r attributes, the goal is to build a partitioning $\mathcal{P} = \{C_1, ..., C_k\}$ of V into k communities such that nodes in the same community are densely connected and similar in terms of attributes, whereas nodes from distinct communities are loosely connected and different in terms of attribute.

For networks with numeric attributes, we can directly apply a community detection algorithm F_s on G to identify a set of communities based on node connections $\mathcal{P}_s = \{C_1, C_2, ..., C_{k_s}\}$, and a clustering algorithms F_a on A to find a set of clusters based on node attributes $\mathcal{P}_a = \{C_1, C_2, ..., C_{k_a}\}$. When it comes to binary attributed networks, traditional clustering algorithms become inaccessible, we instead build a virtual graph G_a that shares the same node set as G, but there is an edge only when the two nodes are similar enough in terms of attributes. Then we apply F_s on G_a and obtain $\mathcal{P}_a$. Note that we omit categorical attributes since categorical values can be easily converted to the binary case.

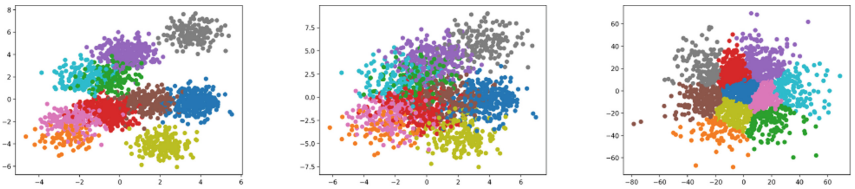
The second step is to combine the partitions $\mathcal{P}_s$ and $\mathcal{P}_a$. We first derive the adjacency matrices D_s and D_a from $\mathcal{P}_s$ and $\mathcal{P}_a$ respectively, where $d_{ij} = 1$ when nodes i and j are in the same community in a partitioning $\mathcal{P}$ and $d_{ij} = 0$ otherwise. Next, an integrated adjacency matrix D is given by $D = \alpha D_s + (1-\alpha)D_a$. Here α is the weighting parameter that leverages the strength between network topology and node attributes. In this way, the information about network topology and node attributes of the original graph G is represented in D. Now G_{int}, derived from the adjacency matrix D, is an integrated, virtual, weighted graph whose edges embody the homophily effect of G. Algorithm 1 shows the steps of our late-fusion approach applied to networks with binary attributes.

Algorithm 1. Late-fusion on networks with binary attributes

Input: $G = (V, E, A), F_s, \alpha$
Output: $\mathcal{P} = \{C_1, C_2, ..., C_k\}$
$\mathcal{P}_s = F_s(G_s)$
$G_a =$ build_virtual_graph (A)
$\mathcal{P}_a = F_s(G_a)$
$D_s =$ get_adjacency_matrix$(\mathcal{P}_s)$, $D_a =$ get_adjacency_matrix$(\mathcal{P}_a)$
$D = \alpha D_s + (1-\alpha)D_a$
$G_{integrated} =$ from_adjacency_matrix (D)
$\mathcal{P} = F_s(G_{integrated})$
return $\mathcal{P}$

Here we address an important detail: how to build the virtual graph G_a from the node-attribute matrix A? We compute the inner product as the similarity measure between each node pair, and if the inner product exceeds a predetermined threshold, we regard the nodes as similar and add a virtual edge between them. The threshold can be determined heuristically based on the distribution of the node similarities. However, the threshold should be chosen properly so that the resulted G_a would be neither too dense nor too sparse, where both cases could harm the quality of the final communities. Under this guidance, we put forward two thresholding approaches:

1. **Median thresholding** (**MT**)**:** Suppose S is the $m \times m$ similarity matrix of all nodes in V, we take all the off-diagonal, upper triangular (or lower triangular) entries of S, find the median of these numbers and set it as the threshold. This approach guarantees that we add virtual edges to half of all node pairs who share a similarity value higher than the other half.
2. **Equal-edge thresholding** (**EET**)**:** We compute $q = 1 - d(G)$ where $d(G)$ is the density of G. Then the q^{th} quantile of the similarity distribution is the chosen threshold. In this approach, we let the original graph G_s be the proxy that decides how we construct the virtual graph G_a.



(a) (b) (c)

Fig. 1. Node attribute distribution for three groups of experiments. (a) Strong attributes, (b) Medium attributes, (c) Weak attributes. Each color represents a unique community (Color figure online)

4 Experiments

Our proposed method has been evaluated through experiments on multiple synthetic and real networks and results are presented in this section. For networks with numeric attributes, we take advantage of existing clustering algorithms to obtain communities based on attributes (i.e., clusters), and for networks with binary attributes, we employ Algorithm 1 to perform community detection. We have also released our code so that readers can reproduce the results[1].

4.1 Synthetic Networks with Numeric Attributes

Data. We use an attributed graph generator [10] to create three attributed graphs with ground-truth communities, denoted as G_{strong}, G_{medium} and G_{weak}, indicating the corresponding ground-truth partitionings are *strong*, *medium*, and *weak* in terms of modularity Q. To examine the effect of attributes on community detection, for each of G_{strong}, G_{medium} and G_{weak}, we assign three different attribute distributions as shown in Fig. 1, where attributes in Fig. 1a and b are generated from a Gaussian mixture model with a shared standard deviation, and Fig. 1c presents the original attributes generated by [10]. By this way, for each graph having a specific community structure (G_{strong}, G_{medium}, G_{weak}) we have also three types of attributes denoted strong attributes, medium attributes and weak attributes leading in fact to 9 datasets.

Evaluation Measures and Baselines. Normalized Mutual Information (NMI) and Adjusted Rand Index (ARI) and running time are used to evaluate algorithm accuracy and efficiency. Louvain [2] and SIWO [7] have been chosen as baseline algorithms that utilize only the links to identify network communities.

[1] https://github.com/changliu94/attributed-community-detection.

Table 1. Properties of synthetic networks

	m	n	k	r	Q
G_{strong}	2000	7430	10	2	0.81
G_{medium}	2000	7445	10	2	0.65
G_{weak}	2000	6988	10	2	0.54

Table 2. Properties of Sina Weibo network

m	n	k	r	Q	I
3490	30282	10	10	0.05	0.04

Note that since the attribute distribution does not affect Louvain and SIWO, the results of Louvain and SIWO are only presented in Table 3. We choose Spectral Clustering (SC) and DBSCAN as two representative clustering algorithms as they both can handle non-flat geometry. We treat the number of clusters as a known input parameter of SC, and the neighborhood size of DBSCAN is set to the average node degree. We adopt default values of the remaining parameters from the *scikit-learn* implementation of these two algorithms. Finally, we take the implementation of the I-Louvain algorithm which exploits links and attribute values as our contender. The code of I-Louvain is available online[2]. Given Louvain, SIWO, SC, and DBSCAN, correspondingly we can have four combinations for our late-fusion method. In all experiments, the α parameter in Algorithm 1 is chosen to be 0.5, i.e., the same weight is allocated to structural and attribute information.

Table 3. Results of strong attributes, time is measured in seconds

	G_{strong}			G_{medium}			G_{weak}		
	NMI	ARI	Time	NMI	ARI	Time	NMI	ARI	Time
Louvain	.795	.797	0.41	.695	.686	0.49	.665	.674	0.64
SIWO	.836	**.850**	0.97	.702	**.705**	1.09	.504	.458	0.98
SC	.802	.713	1.15	.777	.677	0.64	**.768**	.669	0.68
DBSCAN	.469	.103	0.06	.434	.083	0.06	.465	.102	0.24
I-Louvain	.515	.150	39.2	.718	.704	30.0	.608	.503	37.6
Louvain + SC	.824	.704	7.34	.784	.618	5.74	.765	.597	7.14
Louvain + DBSCAN	.818	.813	8.64	.730	.702	8.87	.704	**.690**	10.6
SIWO + SC	**.844**	.738	10.3	**.786**	.636	7.33	.723	.508	6.46
SIWO + DBSCAN	.818	.813	11.7	.730	.702	10.2	.704	**.690**	11.6

[2] https://www.dropbox.com/sh/j4aqitujiaifgq4/AAAAH0L3uIPYNWKoLpcAh0TPa.

Table 4. Results of medium attributes, time is measured in seconds

	G_{strong}			G_{medium}			G_{weak}		
	NMI	ARI	Time	NMI	ARI	Time	NMI	ARI	Time
SC	.529	.338	0.83	.522	.322	0.53	.538	.349	0.57
DBSCAN	.096	.012	0.08	.066	.008	0.14	.065	.011	0.09
I-Louvain	.517	.150	36.8	**.707**	**.690**	33.7	.614	.522	33.2
Louvain + SC	.734	.450	5.62	.696	.390	5.96	**.677**	.392	5.66
Louvain + DBSCAN	**.755**	**.726**	9.20	.670	.636	11.9	.641	**.633**	13.6
SIWO + SC	.748	.469	12.7	.699	.402	7.12	.625	.335	7.44
SIWO + DBSCAN	.744	**.726**	8.73	.670	.636	8.98	.641	**.633**	12.4

Results. Table 3, corresponding to strong attributes, shows that late fusion is the best-performing algorithm in terms of NMI on G_{strong} and G_{medium}, and very close to SC on G_{weak} (0.765 against 0.768) whereas it is better in terms of ARI on this last graph. On Tables 4 and 5, corresponding respectively to medium and weak attributes, with the deterioration of the attribute quality, the accuracy of late-fusion degrades, but late fusion still remains at a consistently high level compared to I-Louvain and the clustering algorithms. Moreover, the performance degradation of late-fusion methods is less susceptible to the deterioration of community quality compared to the clustering algorithms, thanks to the complementary structural information. As for the running time, it is expected that classic community detection algorithms Louvain and SIWO are the fastest algorithms, as they do not consider node attributes, but the late-fusion method still outperforms I-Louvain by a remarkable margin.

Table 5. Results of weak attributes, time is measured in seconds

	G_{strong}			G_{medium}			G_{weak}		
	NMI	ARI	Time	NMI	ARI	Time	NMI	ARI	Time
SC	.483	.270	3.31	.514	.307	2.32	.489	.276	2.45
DBSCAN	.000	.000	0.06	.000	.000	0.06	.000	.000	0.14
I-Louvain	.517	.150	35.1	.707	**.690**	34.3	.614	.522	39.5
Louvain + SC	.770	.670	11.8	.705	.613	10.2	**.689**	.564	9.33
Louvain + DBSCAN	.795	**.797**	11.2	.695	.685	10.4	.667	**.674**	12.9
SIWO + SC	**.797**	.703	13.2	**.709**	.635	12.3	.601	.467	11.0
SIWO + DBSCAN	.795	**.797**	11.6	.695	.685	11.3	.667	**.674**	12.6

4.2 Real Network with Numeric Attributes

Data and Baselines. Sina Weibo[3] is the largest online Chinese micro-blog social networking website. Table 2 shows the corresponding properties of the Sina Weibo network built by [9][4]. It includes within-inertia ratio I, a measure of attribute homogeneity of data points that are assigned to the same subgroup. The lower the within-inertia ratio, the more similar the nodes in the same community are. As DBSCAN algorithm performs poorly on the Sina Weibo network and it is costly to infer a good combination of the hyper-parameters of the algorithm, it has been replaced by k-means as a supplement to spectral clustering. The number of clusters required as an input by k-means and SC is inferred from the 'elbow method', which happens to be 10, the actual number of clusters. Moreover, since we have the prior knowledge that the ground truth communities are based on the topics of the forums from which those users are gathered, we reckon that the formation of communities depends more on the attribute values than the structure and set the parameter α at 0.2.

Results. Table 6 presents the results on Sina Weibo network. The two baseline algorithms Louvain and SIWO and the contending algorithm I-Louvain perform poorly on the Sina Weibo network, whereas the clustering algorithms show a high accuracy. Especially, the k-means algorithm together with our four late-fusion methods with the emphasis on attribute information produce results with the best NMI and ARI. This is because modularity of Sina Weibo network is low (0.05 as indicated in Table 2) and the within-inertia ratio is also low (0.04). The results also validate our assumption that communities in this network are mainly determined by the attributes. We will further explore the effect of α in Sect. 4.4.

Table 6. Experimental results on Sina Weibo network

	NMI	ARI	Time
Louvain	.232	.197	1.98
SIWO	.040	.000	3.26
SC	.612	.520	3.16
k-means	**.649**	**.579**	0.25
I-Louvain	.204	.038	261
Louvain+SC	.611	.519	48.9
Louvain+k-means	**.649**	**.579**	42.1
SIWO+SC	.611	.519	37.9
SIWO+k-means	**.649**	**.579**	50.4

Table 7. Properties of Facebook networks

Network ID	m	n	k	r	Q
0	347	5038	24	224	0.179
107	1045	53498	9	576	0.218
348	227	6384	14	161	0.210
414	159	3386	7	105	0.468
686	170	3312	14	63	0.101
698	66	540	13	48	0.239
1684	792	28048	17	319	0.509
1912	755	60050	46	480	0.339
3437	547	9626	32	262	0.026
3980	59	292	17	42	0.242

[3] http://www.weibo.com.
[4] This dataset is available online https://github.com/smileyan448/Sinanet.

4.3 Real Network with Binary Attributes

Data. Facebook dataset [11] contains 10 egocentric networks with binary attributes corresponding to anonymous information of the user about the name, work, and education and ground-truth communities. This dataset is available online[5] and Table 7 presents the properties of these networks.

We still treat Louvain and SIWO as our baselines. We use the CESNA algorithm [16], able to handle binary attributes in addition to the links, as our contender[6]. To compare the two thresholding strategies proposed in Section 3, we present experimental results of four late-fusion methods: Louvain + equal-edge thresholding (denoted as Louvain-EET), Louvain + median thresholding (denoted as Louvain-MT), SIWO + equal-edge thresholding (denoted as SIWO-EET), and SIWO + median thresholding (denoted as SIWO-MT). We set α to its default value 0.5.

Table 8. NMI of different community detection results on Facebook network

Network ID	0	107	348	414	686	698	1684	1912	3437	3980	Average
Louvain	.382	.332	.478	**.609**	.284	.281	.047	**.565**	.181	**.729**	.389
SIWO	.390	.363	.375	.586	.215	.259	.053	.557	.174	.605	.358
CESNA	.263	.249	.307	.586	.238	.564	.438	.450	.176	.552	.382
Louvain-EET	**.558**	.355	**.525**	.538	**.463**	**.669**	**.462**	.511	**.310**	.704	**.509**
Louvain-MT	.452	.341	.489	.556	.351	.479	.323	.491	.262	.696	.444
SIWO-EET	.541	**.364**	.452	.531	.406	.630	.460	.509	**.310**	.648	.485
SIWO-MT	.431	.353	.405	.538	.252	.406	.332	.491	.260	.588	.406

Table 9. ARI of different community detection results on Facebook network

Network ID	0	107	348	414	686	698	1684	1912	3437	3980	Average
Louvain	.143	.148	.303	.558	.110	.000	.000	.461	.000	.398	.209
SIWO	.220	.177	.127	.519	.000	.009	.000	.419	.002	.209	.167
CESNA	.073	.097	.156	.480	.001	.202	.310	.361	.014	.067	.176
Louvain-EET	.024	.047	.103	.265	.006	.000	.043	.252	.000	.069	.008
Louvain-MT	.061	.079	.129	.413	.063	.000	.048	.235	.000	.084	.110
SIWO-EET	.043	.045	.124	.252	.003	.000	.057	.235	.000	.095	.009
SIWO-MT	.108	.079	.141	.391	.040	.016	.060	.223	.000	.073	.113

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

[6] The source code of CESNA is available online https://github.com/snap-stanford/snap/tree/master/examples/cesna.

Table 10. Running time of different community detection results on Facebook network, measured in seconds

Network ID	0	107	348	414	686	698	1684	1912	3437	3980	Average
Louvain	0.15	1.83	0.12	0.06	0.09	0.02	0.80	1.28	0.31	0.01	0.47
SIWO	0.34	3.78	0.31	0.16	0.17	0.03	1.46	3.79	0.51	0.02	1.06
CESNA	9.76	103	6.02	2.47	3.12	0.63	38.3	22.9	21.1	0.60	20.8
Louvain-EET	0.72	4.68	0.40	0.25	0.24	0.07	1.95	3.83	0.78	0.03	1.30
Louvain-MT	2.90	20.0	0.82	0.48	0.44	0.08	8.22	9.41	3.28	0.06	4.57
SIWO-EET	1.73	24.4	2.87	0.68	0.76	0.14	5.76	28.5	4.26	0.12	6.92
SIWO-MT	9.45	91.4	5.27	1.73	3.14	0.34	44.9	43.4	13.5	0.17	21.3

Results. Results in terms of NMI, ARI, and running time are respectively presented in Tables 8, 9, and 10. In terms of NMI, results in Table 8 show again that our late-fusion algorithms can significantly improve the community detection accuracy upon Louvain. On average, the late fusion method Louvain+EET outperforms Louvain, SIWO, and CESNA by 30.8%, 42.2%, and 33.2% respectively. The late fusion method Louvain+MT outperforms the three by 14.1%, 24.0%, and 16.2% respectively. However, all of the late-fusion methods perform poorly when evaluated by ARI. This is resulted from the goal of our late-fusion approach. Remember that we aim to find the set of communities such that nodes in the same subgroup are densely connected and similar in terms of attributes, whereas nodes residing in different communities are loosely connected and dissimilar in attributes. This purpose led the late-fusion approach to over-partition communities that are formed by only one of the two sources of information. The over-partitioning greatly hurts the results of ARI. A postprocessing model to resolve the over-partitioning issue with late fusion is left as a future work. The running time results shown in Table 10 again manifests the efficiency advantage of our late-fusion methods over CESNA.

4.4 Effect of Parameter α

In the Sina Weibo experiment, we see the advantage of having a weighting parameter to accordingly leverage the strength of the two sources of information. In this section, we dive deeper into the effect of α on the community detection results. To do so, we devise an experiment where we use the G_{strong} and G_{weak} introduced in Table 1. In reverse, we assign **weak** attributes to G_{strong} and **strong** attributes to G_{weak}. Then we perform our late fusion algorithm on these two graphs with varying α values. In our experiment, we choose SIWO as F_s and k-means as F_a.

Table 11 presents the NMI and ARI of the late fusion with SIWO and k-means when α varies. G_{strong} has communities with a strong structure but weak attributes, so the accuracy score for NMI and ARI goes up as we put more weight on the structure; On the contrary, G_{weak} has weak structural communities but

Table 11. Effect of α

	$\alpha = 0.0$		$\alpha = 0.2$		$\alpha = 0.5$		$\alpha = 0.8$		$\alpha = 1.0$	
	NMI	ARI	NMI	ARI	NMI	ARI	NMI	ARI	NMI	ARI
G_{strong}	0.530	0.359	0.530	0.359	0.756	0.513	0.836	0.850	0.836	0.850
G_{weak}	0.867	0.834	0.867	0.834	0.762	0.470	0.526	0.364	0.526	0.364

strong attributes, hence the accuracy score decreases as α increases. One can also notice that when α is sufficiently high or low, late fusion becomes equivalent to using community detection or clustering only, which is in accordance with our observation done on the Sina Weibo experiment.

In practice, when network communities are mainly determined by the links, α should be greater than 0.5; $\alpha < 0.5$ is recommended if attributes play a more important role in forming the communities; When prior knowledge about network communities is unavailable or both sources of information contribute equally, α should be 0.5.

4.5 Complexity of Late Fusion

It is a known drawback of attributed community detection algorithms that they are very time-consuming due to the need to consider node attributes. Our late-fusion method tries to circumvent this problem by taking advantage of the existing community detection and clustering algorithms that are efficiently optimized, and combining their results by a simple approach. To further show the computational efficiency of our late-fusion method, we compute the running time of the late-fusion method and compare it with other methods.

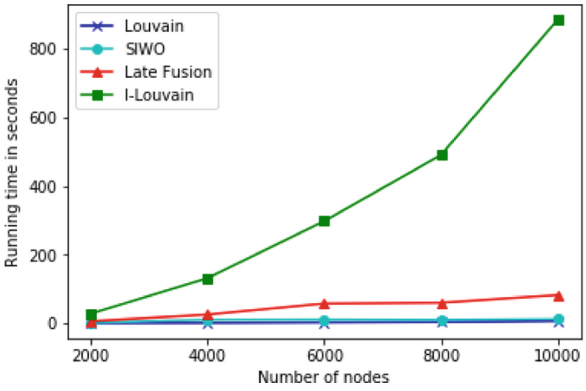


Fig. 2. Running time of Louvain, SIWO, late fusion and I-Louvain on networks of different sizes

We test the running time of four different community detection methods on five graphs with the number of nodes varying from 2000, 4000, 6000, 8000, and 10000. These graphs are also generated by the attributed graph generator [10]. We control the modularity of each graph at the range of 0.64−0.66 and keep other hyperparameters the same. For each size, we randomly sample 10 graphs from the graph generator and plot the average running time of each method. As we can see in Fig. 2, it is expected that our late-fusion method is inevitably slower than the two community detection methods that only utilize node connections. However, our algorithm runs way faster than the I-Louvain algorithm, albeit both being approximately linear in the growth of network sizes.

5 Conclusion and Future Direction

In this paper, we proposed a new approach to the problem of community detection in attributed networks that follows a late-fusion strategy. We showed with extensive experiments that most often, our late-fusion method is not only able to improve the detection accuracy provided by traditional community detection algorithms, but it can also outperform the chosen contenders in terms of both accuracy and efficiency. We learned that combining node connections with attributes to detect communities of a network is not always the best solution, especially when one side of the network properties is strong while the other is weak, using only the best information available can lead to better detection results. It is part of our future work to understand when and how we should use the extra attribute information to help community detection. ARI suffers greatly from over-partitioning issue with our late fusion when applied to networks with binary attributes. A postprocessing model to resolve this issue is desired. We also hope to expand the late-fusion approach to networks with a hybrid of binary and numeric attributes as well as networks with overlapping communities.

References

1. Asim, Y., Ghazal, R., Naeem, W., Majeed, A., Raza, B., Malik, A.K.: Community detection in networks using node attributes and modularity. Int. J. Adv. Comput. Sci. Appl. **8**(1), 382–388 (2017)
2. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. J. Stat. Mech: Theory Exp. **2008**(10), P10008 (2008)
3. Combe, D., Largeron, C., Géry, M., Egyed-Zsigmond, E.: I-Louvain: an attributed graph clustering method. In: Fromont, E., De Bie, T., van Leeuwen, M. (eds.) IDA 2015. LNCS, vol. 9385, pp. 181–192. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24465-5_16
4. Cruz, J.D., Bothorel, C., Poulet, F.: Semantic clustering of social networks using points of view. In: CORIA, pp. 175–182 (2011)
5. Dang, T., Viennet, E.: Community detection based on structural and attribute similarities. In: International Conference on Digital Society (ICDS), pp. 7–14 (2012)

6. Fortunato, S., Hric, D.: Community detection in networks: a user guide. Phys. Rep. **659**, 1–44 (2016)
7. Gharaghooshi, S.Z., Zaïane, O., Largeron, C., Zafarmand, M., Liu, C.: Addressing the resolution limit and the field of view limit in community mining. In: Berthold, M.R., et al. (eds.) Symposium on Intelligent Data Analysis, IDA 2020. LNCS, vol. 12080, pp. 1–12. Springer, Cham (2020)
8. Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD Conference, pp. 855–864. ACM (2016)
9. Jia, C., Li, Y., Carson, M.B., Wang, X., Yu, J.: Node attribute-enhanced community detection in complex networks. Sci. Rep. **7**, 1–15 (2017)
10. Largeron, C., Mougel, P., Benyahia, O., Zaïane, O.R.: DANCer: dynamic attributed networks with community structure generation. Knowl. Inf. Syst. **53**(1), 109–151 (2017)
11. Leskovec, J., Mcauley, J.J.: Learning to discover social circles in ego networks. In: Advances in Neural Information Processing Systems, pp. 539–547 (2012)
12. Li, Y., Sha, C., Huang, X., Zhang, Y.: Community detection in attributed graphs: An embedding approach. In: AAAI (2018)
13. Neville, J., Adler, M., Jensen, D.: Clustering relational data using attribute and link information. In: Proceedings of the Text Mining and Link Analysis Workshop, 18th International Joint Conference on Artificial Intelligence, pp. 9–15. Morgan Kaufmann Publishers, San Francisco (2003)
14. Ruan, Y., Fuhry, D., Parthasarathy, S.: Efficient community detection in large networks using content and links. In: Proceedings of the 22nd International Conference on World Wide Web, pp. 1089–1098. ACM (2013)
15. Steinhaeuser, K., Chawla, N.V.: Community detection in a large real-world social network. In: Liu, H., Salerno, J.J., Young, M.J. (eds.) Social Computing, Behavioral Modeling, and Prediction, pp. 168–175. Springer, Boston (2008). https://doi.org/10.1007/978-0-387-77672-9_19
16. Yang, J., McAuley, J., Leskovec, J.: Community detection in networks with node attributes. In: ICDM Conference, pp. 1151–1156. IEEE (2013)
17. Zhou, Y., Cheng, H., Yu, J.X.: Graph clustering based on structural/attribute similarities. Proc. VLDB Endow. **2**(1), 718–729 (2009)