SE3352: Algorithm Design December 14, 2021

Notes – Community Detection

Instructor: *Guoqiang Li* Scribes: *Zhemin Huang*, *Xunjie Wang*

Abstract

Community detection is an important and extremely useful tool in both scientific research and data analytics. This note offers a guided tour through the main aspects of community detection. We introduce several widely-used algorithms, ranging from traditional algorithms to state-of-the-art algorithms, which are mostly based on modularity. Besides, we point out the algorithm's application in industries. The challenges faced by community detection algorithms, open problems and future trends related to community detection are also discussed.

Keywords: community detection, network, community structure, modularity, deep learning

1 Introduction

Nowadays, research on complex networks is getting more and more popular. A great variety of systems in nature, society or technology, can be modelled as networks. Community detection, or more specifically, **clustering nodes based on similar features**, helps us understand the inherent patterns and functions of networks.

Researchers have proposed many community detection algorithms with different types and scale of complex networks. However, there are still no universal guidelines on how to assess the performance of different algorithms, or compare them with each other. Traditionally, modularity is preferred by most mainstream algorithms. With the era of deep learning coming, new algorithms and problems are being discussed as well.

This note is composed as follows. We introduce some related concepts of community detection in Section 2. In section 3, we introduce some key properties of the basic concept – modularity. In section 4-7, we introduce several algorithms of different levels, and parallel implementation in industries. In section 8-9, we discuss community detection's real applications, and its future directions.

2 Problem Description

In this section, we first define two simple concepts: network and community.

A network is a special graph that abstracts **complex relationships** in real-world systems, such as Internets, social networks, or biological cell architecture.

Definition 1 (Network). *A weighted network is represented as $G = (V, E, W)$, and an unweighted network is represented as $G = (V, E)$, where V, E denote the set of nodes and edges, respectively, and W denotes the corresponding weights of the connections. In an unweighted network, W can be regarded as 1 and can be removed from G. Besides, a network is either a directed graph, or an undirected graph.*

A community is a type of sub-graph that represents some real social phenomenon. In order words, a community is a group of people or objects, which **share common characteristics**.

Definition 2 (Community). *Communities are the sub-graphs in a network, where nodes share dense connections. Sparsely-connected nodes cripple communities. We use $C = \{C_1, C_2, ..., C_k\}$ to denote a set of k communities divided from a network G. A node v clustered into the community C_i satisfies the condition that the internal degree of each node inside the community exceeds its external degree. For $\forall i, j, C_i \in C, C_j \in C$, they satisfy the following conditions*

$$C_i \cap C_j = \varnothing \tag{1}$$

$$\bigcup_{i=1}^{k} C_i = G \tag{2}$$

Modularity was designed to measure the strength of division of a network into communities. It was proposed by Newman.[21]

Definition 3 (Modularity). *Consider a particular division of a network into k communities. Let us define a $k \times k$ symmetric matrix e whose element e_{ij} is the fraction of all edges in the network that link vertices in community i to vertices in community j. We further define the row sums $a_i = \sum_j e_{ij}$, which represent the fraction of edges that connect to vertices in community i. In a network in which edges fall between vertices who belong to different communities, we would have $e_{ij} = a_i a_j$. Let $||X||$ indicates the sum of the elements of the matrix X. Thus, we define modularity by*

$$Q = \sum_i e_{ii} - {a_i}^2 = \sum_i e_{ii} - \sum_i {a_i}^2 = Trace - ||e^2|| \tag{3}$$

If Q is high, we get strong community structure. In practice, values for such networks typically fall in the range from about 0.3 to 0.7.[21]
In order to raise **a spectral algorithm for community detection**, Newman redefined the concept of modularity.[20]

Definition 4 (Reformulation of the modularity). *The modularity of a community partition is a scalar ranging from $-\frac{1}{2}$ to 1 that evaluates the density of links inside communities as compared to links between communities. For a given graph $G = (V, E)$. Let A_{ij} be the weight of the edge between i and j, $k_i = \sum_{(i,j)\in E} A_{ij}$ is the sum of the weights of all the edges attached to vertex i, $m = \frac{1}{2}\sum_{ij} A_{ij}$, the quantity*

$\frac{1}{2}(s_i s_j + 1)$ *is 1 if* i *and* j *are in the same group and 0 otherwise,*

$$Q = \frac{1}{4m} \sum_{i \in V, j \in V} (A_{ij} - \frac{k_i k_j}{2m}) \frac{1}{2}(s_i s_j + 1) \tag{4}$$

The leading factor of $\frac{1}{4m}$ seems to be confusing. Actually, it is included for compatibility with the previous definition of modularity.
From the definitions above, we can lead to our goal–community detection.

Definition 5 (Community detection)**.** *Community detection is a method to extract communities from large networks when the modularity is optimal.*

We need to point out that not all methods are based on modularity, such as Infomap[26]. Modularity has its limits[10], but it is still the most commonly used method for evaluating results.

3 Key Properties

In this section, we introduce some key properties of modularity, which helps to understand following algorithms.

Lemma 6. *We denote the set of all possible communities of a graph* G *with* $A(G)$*,* $Q(P)$ *refers to the modularity of* P*. Let* G *be an undirected and unweighted graph and* $P \in A(G)$*. Then* $-\frac{1}{2} \leq Q(P) \leq 1$ *holds.*

Proof. Let $m_i = |E(P)|$ be the number of edges inside community P and $m_e = \sum_{P \neq P' \in P} |E(P, P')|$ be the number of edges having exactly one end-node in P. Then we can calculate the contribution of P to $Q(P)$

$$\frac{m_i}{m} - (\frac{m_i}{m} + \frac{m_e}{2m})^2 = \frac{-4(m_i)^2 + 4m_i(m - m_e) - (m_e)^2}{4m^2} \tag{5}$$

It is obvious that the only maximum point is at $m_i = \frac{m - m_e}{2}$. The contribution of a community is minimized when m_i is zero and m_e is as large as possible. Suppose now $m_i = 0$, the upper bound can only be actually attained in the specific case of a graph with no edges, where coverage is defined to be 1. Besides, any bipartite graph $K_{a,b}$ with simple communities $C = \{C_a, C_b\}$ yields the minimum modularity of $\frac{1}{2}$. This proof the lemma. □

Theorem 7. *Modularity is* ***NP-complete****.[3]*

Proof. We formalize the problem of finding modularity, and prove it by **reduction**.
Problem 1 (Modularity) : Given a graph G and a number K, is there a community $P \in G$, for which $Q(P) \geq K$? $(K \in [-\frac{1}{2}, 1])$
Problem 2 (3-Partition) : Given $3k$ positive integer numbers $a_1, ..., a_{3k}$ such that the sum $\sum_{i=1}^{3k} a_i = kb$,

and $\frac{b}{4} < a_i < \frac{b}{2}$, for an integer b and for all $i = 1, ..., 3k$, is there a partition of these numbers into k sets, such that the sum of the numbers in each set equals to b?

It has been proved that 3-partition problem is a NP-Complete problem.[12]

Then we show that an instance $A = a_1, ..., a_{3k}$ of 3-partition problem can be transformed into an instance $G(A), K(A)$ of modularity problem.

Given an instance A of 3-partition, we can construct a graph $G(A)$ with k cliques $H_1, ..., H_k$ of size $a = \sum_{i=1}^{3k} a_i$ each. For each element $a_i \in A$, we introduce a single element node in graph, and connect it to a_i nodes in each of the k cliques, where each clique member is connected to exactly one element node. Therefore, each clique node has degree a, and the element node corresponding to element $a_i \in A$ has degree ka_i. The number of edges in $G(A)$ is $m = \frac{k}{2}a(a+1)$.

Then we construct $K(A)$. Since graph $G(A)$ has exactly k cliques, it has exactly $(k-1)a$ inter-community edges, so the edge contribution is given by

$$\sum_{C \in P} \frac{|E(C)|}{m} = \frac{m-(k-1)a}{m} = 1 - \frac{2(k-1)a}{ka(a+1)} = 1 - \frac{2k+2}{k(a+1)} \tag{6}$$

Therefore, communities $P = (C_1, ..., C_k)$ with maximum modularity must minimize $d(C_1)^2 + d(C_2)^2 + ... + d(C_k)^2$, where $d(C_k)$ refers to the degree of C_k. Then the sum of degrees per community should be as small as possible. In the optimum case, we can assign to each community element nodes corresponding to elements that sum to $b = \frac{1}{k}a$. In each clique, the sum equals to $k\frac{1}{k}a = a$.

$$d(C_1)^2 + ... + d(C_k)^2 \geq k(a^2+a)^2 = ka^2(a+1)^2 \tag{7}$$

Hence, if there exist communities P with $Q(P)$, then

$$K(A) \geq 1 - \frac{2k-2}{k(a+1)} - \frac{ka^2(a+1)^2}{k^2a^2(a+1)^2} = \frac{(k-1)(a-1)}{k(a+1)} \tag{8}$$

As each element node is contained in exactly one community, this yields a solution for the instance of 3-partition. The instance of 3-partition is satisfiable if the instance of modularity is satisfiable.

Otherwise, suppose the instance for 3-partition is satisfiable. Then there exists a partition into k sets, where the sum over each set is $\frac{1}{k}a$. If we detect communities by joining the element nodes of each set with a different clique, we get communities of modularity $K(A)$. Therefore, the instance of modularity is satisfiable if the instance of 3-partition is satisfiable.

Therefore, the theorem holds. □

4 Girvan-Newman Algorithm

4.1 Introduction

Girvan-Newman algorithm is the first algorithm of the modern age of community detection in graphs.[20] It is a hierarchical divisive method, in which links are iteratively removed based on the value of their betweenness, which expresses the number of shortest paths between pairs of nodes that pass through the link. In the most popular implementation, the procedure of link removal ends when the modularity of the resulting partition reaches a maximum.

We can describe Girvan-Newman algorithm in the following way:

1) Calculate edge betweenness for every edge in the graph.
2) Remove the edge with the highest edge betweenness.
3) Calculate edge betweenness for remaining edges.
4) Repeat steps 2-4 until all edges are removed.

4.2 Edge Betweenness

Definition 8 (Edge Betweenness)**.** *the edge betweenness of* (i, j) *is the number of the shortest paths between pairs of vertices that pass through the edge* (i, j)*.*

For a given graph $G = (V, E)$, Let $d_{s,i}$ be the length of the shortest path between vertex s and vertex i, $w_{s,i}$ be the number of the shortest paths from vertex s to vertex i, $b_{s,i}$ be the number of the shortest paths between vertex s to any vertex in graph that pass through vertex i. Edge betweenness $\sigma_{i,j}$ of edge (i, j) can be computed by:

$$\sigma_{i,j} = \sum_{s \in V} \sigma_{s,i,j} \tag{9}$$

where

$$\sigma_{s,i,j} = \begin{cases} \frac{w_{s,j}}{w_{s,i}} b_{s,i} & d_{s,i} > d_{s,j} \\ \frac{w_{s,i}}{w_{s,j}} b_{s,j} & d_{s,i} < d_{s,j} \\ 0 & d_{s,i} = d_{s,j} \end{cases} \tag{10}$$

4.3 Method

For each source vertex s, we can compute the triple $(d_{s,i}, b_{s,i}, w_{s,i})$ as follows.[20]

The first part of the algorithm for vertex marking:

1) For initial vertex $s \in V$, let $d_{s,s} = 0, w_{s,s} = 1, b_{s,i} = 0$.
2) Let $d_{s,v} = \infty$, $w_{s,v} = 0, b_{s,v} = 1$ for all $v \neq s \in V$.

3) Create queue Q, $Q \leftarrow \{s\}$. Create list L, $L \leftarrow \{s\}$.
4) While Q is not empty:
    a) Dequeue $i \leftarrow Q$.
    b) For each vertex j where $(i, j) \in E$
        i) If $d_{s,j} = \infty$ then $d_{s,j} = d_{s,i} + 1, w_{s,j} = w_{s,i}$. Enqueue $j \rightarrow Q$. Push $j \rightarrow L$.
        ii) If $d_{s,j} \neq \infty$ and $d_{s,j} = d_{s,i} + 1$ then $w_{s,j} = w_{s,j} + w_{s,i}$.
        iii) If $d_{s,j} \neq \infty$ and $d_{s,j} < d_{s,i} + 1$ then do nothing.

For each source vertex s, each edge will be visited twice. Therefore, the time complexity of this part of the algorithm is $O(mn)$.

The second part of the algorithm starts from the vertex that was last marked in the first part of the algorithm and visits vertices in reverse order than they were visited in the first part of the algorithm:

1) While L is not empty:
    a) Pop $i \leftarrow L$.
    b) For each vertex j where $(i, j) \in E$
        i) If $d_{s,i} < d_{s,j}$ then $b_{s,i} = 1 + \sum_j \sigma_{s,i,j}$.
        ii) If $d_{s,i} > d_{s,j}$ then $\sigma_{s,i,j} = \frac{w_{s,j}}{w_{s,i}} b_{s,i}$.

Similar to the first part, the time complexity of this part of the algorithm is also $O(mn)$.

Each edge will be removed only once. Therefore, the time complexity of the whole algorithm is $\boldsymbol{O(m^2n)}$.

4.4 Conclusion

Girvan-Newman algorithm is an intuitive algorithm and easy to implement, which can produce correct classifications of vertices in small networks.

But the time complexity is $\boldsymbol{O(m^2n)}$, or $\boldsymbol{O(n^3)}$ for sparse networks, which limits performance on gigantic networks.[20]

A greedy algorithm, which will be discussed later, can give much smaller complexity than Girvan-Newman algorithm.

5 Louvain Algorithm

5.1 Introduction

Fast unfolding of communities in large networks, also known as Louvain algorithm[2], is a heuristic optimization algorithm of community detection based on the modularity of the given graph. The algorithm performs better in terms of efficiency and effectiveness, and can discover the hierarchical community structures. The goal of optimization is to maximize the modularity of the entire community network.

5.2 Method

The Louvain algorithm is composed of two iteration phases. Initially, each vertex in the network will be considered as an individual community.

In the first phase, for each vertex i, consider the change of modularity ΔQ when removing vertex i from its community and then placing it to the community of its neighbor j. This action will be executed only if ΔQ is positive and is the maximal ΔQ of all adjacent of vertex i. Obviously, the first phase will terminate when Q reaches the local maxima.

This removing-placing action can be decomposed into two similar actions: making vertex i isolated and moving the isolated vertex into a community. The removing action is just the inverse action of placing action. Let $\sum_{in}$ be the sum of the weights of the internal edges of c_j, $\sum_{tot}$ be the sum of the weights of the links incident to vertices in community c_j, $k_{i,in}$ be the sum of the weights of the links from i to vertices in community c_j. When moving an isolated vertex i into a community c_j, the change of modularity can be easily computed by:

$$\Delta Q = \left[\frac{\sum_{in}+k_{i,in}}{2m} - \left(\frac{\sum_{tot}+k_i}{2m}\right)^2\right] - \left[\frac{\sum_{in}}{2m} - \left(\frac{\sum_{tot}}{2m}\right)^2 - \left(\frac{k_i}{2m}\right)^2\right] \tag{11}$$

Fortunately, $\sum_{tot}$, $\sum_{in}$ and $k_{i,in}$ can be easily maintained during the iteration.

In the second phase, the whole network will shrink to a new network whose vertices are the communities generated by the first phase. The weights of edges between two new vertices are determined by sum of the weights of the edges between vertices in the corresponding two communities. The internal edges of a community leads to a self-loop for the corresponding vertex in the new network. The second phase will generate a new network that is possible to reapply the algorithm in the first phase, and then the algorithm will go to next iteration.

Let us denote by "pass" a combination of these two phases. The passes will keep being iterated until convergence of the modularity Q. Figure 1 shows the visualization of the Louvain algorithm with a simple graph. As discussed before, the Louvain algorithm can discover the hierarchical structures of communities of the network. The height of hierarchical community structure is equal to the number of passes, which is empirically a small number in common cases.

5.3 Conclusion

The Louvain algorithm is an intuitive heuristic algorithm, which is a multistep technique based on a local optimization of Newman-Girvan modularity in the neighborhood of each node. Due to the easy computation of ΔQ and swift convergence of Q, the whole algorithm runs extremely fast. Many experiments suggest that its complexity is linear on typical and sparse data.

Algorithm 1 Pseudocode of the Louvain algorithm[1]

Require: the initial network $G = (V, E)$
Ensure: final community assignment C, and modularity Q

```
repeat
  Put each node of G in its community
  Q' ← Q
  while some nodes are moved do
    for v ∈ G do
      Place v in its neighboring community including its own, which maximizes the modularity gain
    end for
  end while
  Calculate new modularity Q
  if Q > Q' then
    C ← the network between communities of G
  else
    return C, Q
  end if
until true
```

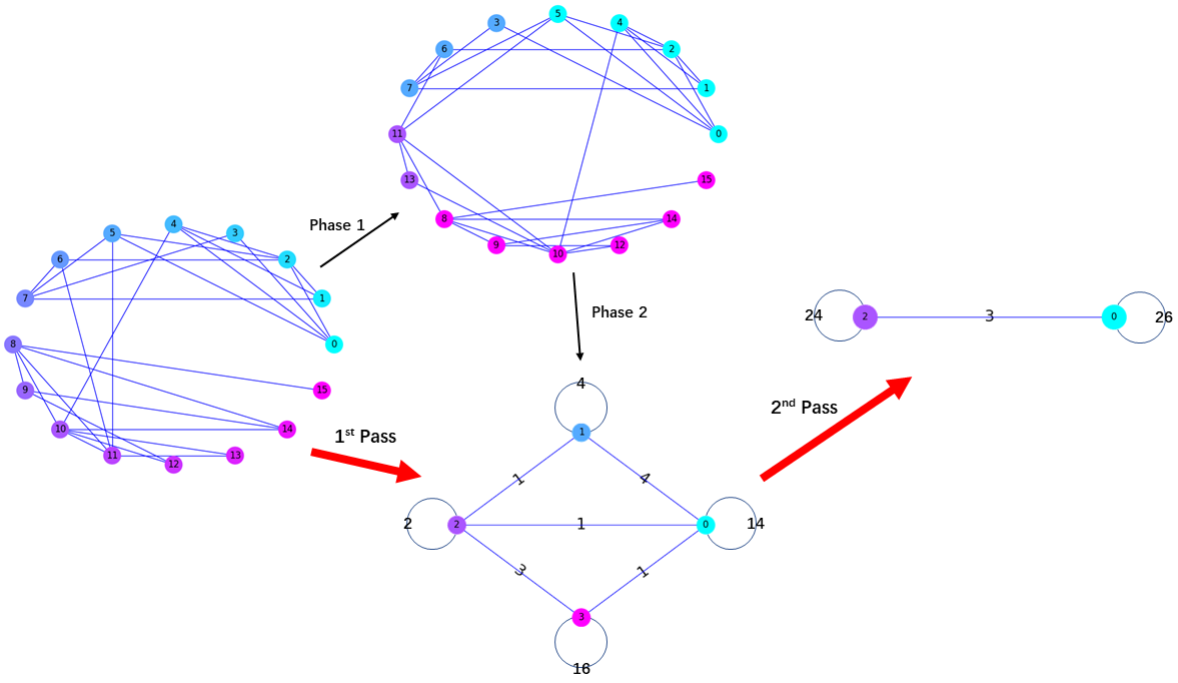

Figure 1: Visualization of the steps of Louvain algorithm. The first phase is to optimize the modularity using a heuristic approach. The second phase is to aggregate vertices in one community into a super vertex, in order to generate a new network possibly for reapplying the first phase. When the network can not gain modularity anymore, the whole algorithm terminates.

6 Parallelization of Louvain algorithm

In the original serial algorithm, each vertex examines the communities of its neighbors and makes a choice to chooses a new community based on a function to maximize the calculated change in modularity. In the distributed version all vertices make this choice simultaneously rather than in serial order, updating the graph state after each change. Because choices are made in parallel some choice will be incorrect and will not maximize modularity values, however after repeated iterations community choices become more stable and can give an approximate result closely mirror the serial algorithm.

6.1 Implementation

Practically, It is a good choice to implement a parallel algorithm on Spark.[34] Spark is a cluster computing framework supporting reusing a working set of data across multiple parallel operations while retaining the scalability and fault tolerance of MapReduce.[7] The implementation of parallelization of Louvain on Spark is described as:

1) Consider each vertex in the graph as an independent community.
2) For each vertex i, remove the vertex i from the original community and assign it to one of the communities of its adjacent vertices, calculate the modularity increment after vertex i is assigned to the new community, and record the adjacent vertex with the maximal modularity increment. If the modularity increment is positive, add vertex i to the community where the adjacent vertex corresponding to the maximal modularity increment Otherwise it remains unchanged.
3) Repeat 2) in parallel until the structure of communities no longer changes.
4) Merge vertices in the same community into new vertices and update weights of edges in parallel.
5) Repeat 1) in parallel until the modularity of the whole network no longer changes.

To describe using MapReduce:

1) Get information of adjacent vertices.
   a) Map: produce information of adjacent vertices VertexData;
   b) Reduce: get information of adjacent vertices (Id, Array[VertexData]).
2) Get the new community of every vertices (Id, getBestCommunity(Array[VertexData]))
3) Update the information in the network and merge vertices to go to next iteration.

6.2 Conclusion

Parallelization can effectively improve the efficiency of Louvain algorithm. This is because it makes full use of the excellent performance of parallel computing in iterative computing when computing information attached to vertices, combined with the distributed computing model, which greatly reduce computing time.

7 Improvement of Louvain Algorithm

7.1 Drawbacks of Louvain Algorithm

The Louvain algorithm is very simple and elegant, however, it has some **crucial drawbacks**.
In the Louvain algorithm, a node may be moved to a different community, while it may have acted as a "bridge" between different communities. Removing such a node disconnects the nodes in the old community.
An expected result is that other nodes in the old community will then be assigned to other communities. However, this is not necessarily the case. Figure 2 has elucidated such situation.

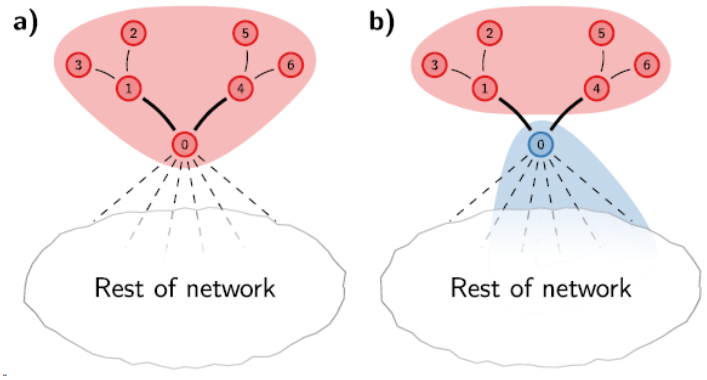

Figure 2: Disconnected community. When node 0 in (a) is moved to a different community, the red community becomes internally disconnected, as shown in (b). However, node 1-6 will still stay in the red community, which is not a satisfying result.[29]

7.2 Leiden Algorithm

The Leiden algorithm was named after the university's name, and was designed to solve the problems of the Louvain algorithm.[29] The Leiden algorithm is more complex than the Louvain algorithm. It consists of three phases:

1) Local moving of nodes.
2) Refinement of the partition.
3) Aggregation of the network based on the refined partition, use the non-refined partition to create an initial partition for the aggregate network.

From figure 3, we can have an intuitive understanding of the whole process.
The algorithm has been widely accepted and used. For example, **scanpy** is a well-known python package for analyzing single-cell gene expression data.[31] Developers have replaced the default community detection method with Leiden algorithm, which has better performance on large-scale data.

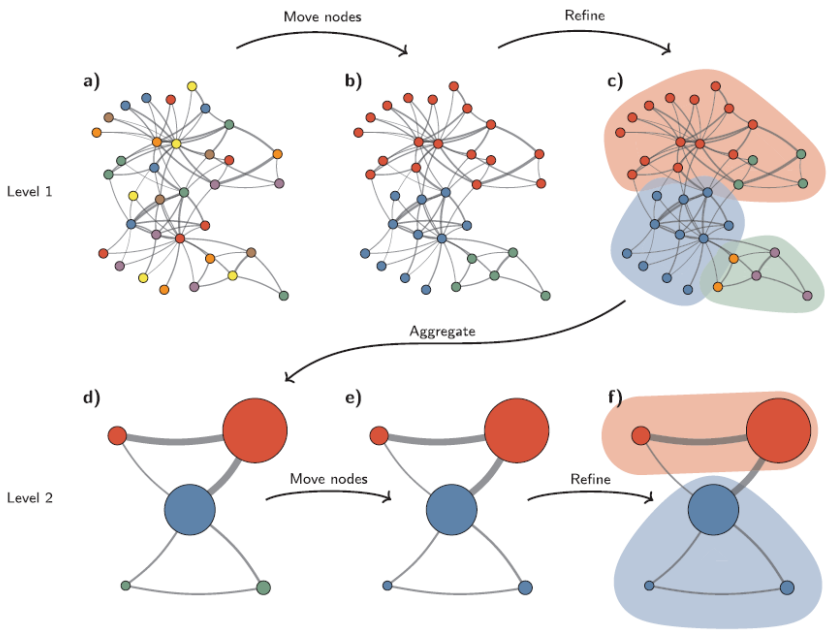

Figure 3: Leiden algorithm. The Leiden algorithm starts from a singleton partition (a). The algorithm moves individual nodes from one community to another to find a partition (b), which is then refined (c). An aggregate network (d) is created based on the refined partition, using the non-refined partition to create an initial partition for the aggregate network. For example, the red community in (b) is refined into two subcommunities in (c), which after aggregation become two separate nodes in (d), both belonging to the same community. The algorithm then moves individual nodes in the aggregate network (e). In this case, refinement does not change the partition (f). These steps are repeated until no further improvements can be made.[29]

7.3 Conclusion

The Leiden algorithm is an improved version of the Louvain algorithm, which is also the **state-of-the-art** algorithm. It is more complicated than Louvain algorithm, but has better compatibility and efficiency. Table 1 shows the comparison of the Louvain and Leiden algorithm.

8 Applications of community detection

The goal of community detection algorithms is to deduce the properties and relationship of nodes, which is not possible from direct observations/measurements. This section explores what can be done by using community detection algorithms.

There are many networks such as social networks, informational networks, biological networks, and even economic networks, where community detection is playing an important role.

Table 1: Comparison between Louvain and Leiden algorithm

	Louvain	**Leiden**
Year	2008	2019
Proposed by	Blondel et al.	V.A. Traag et al.
Phase	2	3
Advantage(s)	• Performs well on small and medium-scale networks. • Static networks.	• Higher quality function, less time to run. • Temporal networks. • The community are well-connected when running the algorithm iteratively.
Disadvantage(s)	• Communities are badly connected or disconnected, especially when running the algorithm iteratively. • Disable detect in overlapping community.	• Extra step in the process.

8.1 Applications in Social Networks

Social networks are getting more and more popular today. An online social network is the interaction of people with each other through the web. Community detection has been widely used in this area.
Twitter is a popular social networking service around the world. Various studies have utilized different community detection methods to analyze public emotional reaction and visualize user relationships and characteristics. Researchers analyzed the reactions of Twitter users on the social and political issues. They used a community-based model to analyze and predict how these big events can influence users.[16] In another work, researchers used non-negative matrix factorization framework to cluster the politically inclined users on Twitter network into political communities.[23]
During **crisis situations**, people often use social media to seek for help. Researchers have used a large scale contextualized community detection to find and recommend potential and relevant collaborators through social media, which has good time performance as well.[25]

8.2 Applications in Biological Networks

Proteins have unique and complicated structures, which can be divided into functional modules and protein complexes.[5] These two parts of proteins have been explored in various studies. Researchers have used simple Girvan-Newman algorithm to detect functional modules in protein-protein interaction networks.[8] Besides, Protein constitutes the fundamental material of human's brain cells. Therefore, community detection can also been applied to **neurosciences** to study brain networks and even find the

mechanism of Alzheimer's disease.[35]
In another research, researchers have proposed an algorithm called **disease-gene network detecting algorithm**, based on Principal Component Analysis (PCA), which can extract the communities in a bipartite network.[19] This algorithm is aimed at disease prevention and medical diagnosis.

8.3 Applications in Economics

In economics and financial data analysis, it is important to categorize companies based on their economic value, stock price, current achievement, etc.
In the **stock market**, each stock can be represented by a vertex and edge represents the correlations of stock values in the market. Researchers have stated the way to construct the network of stock market and detect communities in it. They revealed community structure by using modularity Q, which helps to the analysis and decision-making of the stock market.[13]

8.4 Other Applications

Besides having applications mentioned above, community detection has various general application.
For example, community detection is also being employed for **refactoring of software packages**. Refactoring is an effective way to improve the quality of the existing code. Researchers have presented a novel method to refactor object-oriented software package structure by using classes and their dependencies in software networks[24], which has been used in a lot of modern integrated development environments (IDE).

9 Conclusion

In this note, we have discussed some key properties, presented several commonly used algorithms and highlighted the industrial application of Louvain algorithm. We also reviewed the multidisciplinary applications of community detection in various domains such as sociology, biology, economics, software engineering, etc. However, there are still some questions left to be discussed.

9.1 Future Direction

Deep learning is absolutely a promising direction of community detection. Beyond simply examining network topologies for detecting communities, some strategies also explore semantic descriptions as node features in the data.
Recent researches have modified deep learning models based on community properties, and got promising results.[30] Through a transitional graph embedding method, node distributions could be used to preserve the network structure to improve community detection in reverse.

Temporal analysis of community detection is also a future direction. Real world networks evolve over time, and the communities in such networks also change accordingly. Therefore, exploiting temporal characteristics are crucial to learn deep insights about the communities.

9.2 Open Problems

In this section, we discuss several broad challenges, or the longest standing issue in community detection.

9.2.1 Network Dynamics

Changing dynamics can affect either the network topology or the node attributes. Topological changes, such as adding or deleting a node or edge, not only cause changes in a local community, but also leads to devastating changes across an entire network. With dynamic networks, we need to recalculate the whole graph over a series of snapshots. The technical challenge lies in the deep feature extraction of dynamic networks.

Actually, a lot of methods have been used to deal with sequential data in machine learning, such as long short-term memory (LSTM).[15] Therefore, deep learning methods for detecting communities with dynamic spatial and temporal properties are very likely to be developed.

9.2.2 Large-scale Networks

Nowadays, large-scale networks can contain millions of nodes, edges, and structural patterns, as networks like Twitter and Weibo, which has also brought a lot of problems. For instance, large-scale networks may have their inherent characteristics, such as **scale-free**. There exists lots of mega hubs in the network, which can influence the performance of algorithms in community detection.

Besides, scalability seems to be another tricky problem to detect communities in large-scale networked environments.

9.2.3 Number of Communities

In fact, most algorithms of the community detection require the **number of communities** beforehand as a hyperparameter, including deep learning.

There exist two approaches for the estimation of the number of communities in a network. One method is using modularity-based algorithm, like Girvan-Newman algorithm, and other one is based on **statistical interference**, which is also proposed by Newman.[22] However, both methods have poor performance under large-scale networks.

References

[1] Aynaud, T., and Guillaume, J.-L. Static community detection algorithms for evolving networks. In *8th International symposium on modeling and optimization in mobile, Ad Hoc, and wireless networks* (2010), IEEE, pp. 513–519.

[2] Blondel, V. D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment 2008*, 10 (2008), P10008.

[3] Brandes, U., Delling, D., Gaertler, M., Gorke, R., Hoefer, M., Nikoloski, Z., and Wagner, D. On modularity clustering. *IEEE transactions on knowledge and data engineering 20*, 2 (2007), 172–188.

[4] Cavallari, S., Zheng, V. W., Cai, H., Chang, K. C.-C., and Cambria, E. Learning community embedding with community detection and node embedding on graphs. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management* (2017), pp. 377–386.

[5] Chen, B., Fan, W., Liu, J., and Wu, F.-X. Identifying protein complexes and functional modules—from static ppi networks to dynamic ppi networks. *Briefings in bioinformatics 15*, 2 (2014), 177–194.

[6] De Meo, P., Ferrara, E., Fiumara, G., and Provetti, A. Generalized louvain method for community detection in large networks. In *2011 11th international conference on intelligent systems design and applications* (2011), IEEE, pp. 88–93.

[7] Dean, J., and Ghemawat, S. Mapreduce: simplified data processing on large clusters. *Communications of the ACM 51*, 1 (2008), 107–113.

[8] Dunn, R., Dudbridge, F., and Sanderson, C. M. The use of edge-betweenness clustering to investigate biological function in protein interaction networks. *BMC bioinformatics 6*, 1 (2005), 1–14.

[9] Fazlali, M., Moradi, E., and Tabatabaee Malazi, H. Adaptive parallel louvain community detection on a multicore platform. *Microprocessors and Microsystems 54* (2017), 26–34.

[10] Fortunato, S., and Barthelemy, M. Resolution limit in community detection. *Proceedings of the national academy of sciences 104*, 1 (2007), 36–41.

[11] Fortunato, S., and Hric, D. Community detection in networks: A user guide. *Physics Reports 659* (2016), 1–44.

[12] Garey, M. R. A guide to the theory of np-completeness. *Computers and intractability* (1979).

[13] Gui, X., Li, L., Cao, J., and Li, L. Dynamic communities in stock market. In *Abstract and Applied Analysis* (2014), vol. 2014, Hindawi.

[14] He, D., Song, Y., Jin, D., Feng, Z., Zhang, B., Yu, Z., and Zhang, W. Community-centric graph convolutional network for unsupervised community detection. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence* (2021), pp. 3515–3521.

[15] Huang, Z., Xu, W., and Yu, K. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991* (2015).

[16] Hughes, A. L., and Palen, L. Twitter adoption and use in mass convergence and emergency events. *International journal of emergency management 6*, 3-4 (2009), 248–260.

[17] Javed, M. A., Younis, M. S., Latif, S., Qadir, J., and Baig, A. Community detection in networks: A multidisciplinary review. *Journal of Network and Computer Applications 108* (2018), 87–111.

[18] Lancichinetti, A., and Fortunato, S. Community detection algorithms: a comparative analysis. *Physical review E 80*, 5 (2009), 056117.

[19] Liu, W., and Chen, L. Community detection in disease-gene network based on principal component analysis. *Tsinghua Science and Technology 18*, 5 (2013), 454–461.

[20] Newman, M. E. Modularity and community structure in networks. *Proceedings of the national academy of sciences 103*, 23 (2006), 8577–8582.

[21] Newman, M. E., and Girvan, M. Finding and evaluating community structure in networks. *Physical review E 69*, 2 (2004), 026113.

[22] Newman, M. E. J., and Reinert, G. Estimating the number of communities in a network. *Phys. Rev. Lett. 117* (Aug 2016), 078301.

[23] Ozer, M., Kim, N., and Davulcu, H. Community detection in political twitter networks using nonnegative matrix factorization methods. In *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)* (2016), IEEE, pp. 81–88.

[24] Raji, M. Refactoring software packages via community detection from stability point of view. *CoRR abs/1811.10171* (2018).

[25] Rebhi, W., Yahia, N. B., and Saoud, N. B. B. Towards an intelligent application of large scale community detection to support collaboration during emergency management. In *International Conference on Information Systems for Crisis Response and Management in Mediterranean Countries* (2015), Springer, pp. 39–49.

[26] Rosvall, M., and Bergstrom, C. T. Maps of random walks on complex networks reveal community structure. *Proceedings of the national academy of sciences 105*, 4 (2008), 1118–1123.

[27] Shao, J., Han, Z., Yang, Q., and Zhou, T. Community detection based on distance dynamics. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2015), pp. 1075–1084.

[28] Silva, W., Santana, A., Lobato, F., and Pinheiro, M. A methodology for community detection in twitter. In *Proceedings of the International Conference on Web Intelligence* (New York, NY, USA, 2017), WI '17, Association for Computing Machinery, p. 1006–1009.

[29] Traag, V. A., Waltman, L., and Van Eck, N. J. From louvain to leiden: guaranteeing well-connected communities. *Scientific reports 9*, 1 (2019), 1–12.

[30] Tu, C., Zeng, X., Wang, H., Zhang, Z., Liu, Z., Sun, M., Zhang, B., and Lin, L. A unified framework for community detection and network representation learning. *IEEE Transactions on Knowledge and Data Engineering 31*, 6 (2019), 1051–1065.

[31] Wolf, F. A., Angerer, P., and Theis, F. J. Scanpy: large-scale single-cell gene expression data analysis. *Genome biology 19*, 1 (2018), 1–5.

[32] Wu, L., Zhang, Q., Chen, C.-H., Guo, K., and Wang, D. Deep learning techniques for community detection in social networks. *IEEE Access 8* (2020), 96016–96026.

[33] Yang, Z., Algesheimer, R., and Tessone, C. J. A comparative analysis of community detection algorithms on artificial networks. *Scientific reports 6*, 1 (2016), 1–18.

[34] Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., Stoica, I., et al. Spark: Cluster computing with working sets. *HotCloud 10*, 10-10 (2010), 95.

[35] Zhang, Y. *Cluster Analysis and Network Community Detection with Application to Neuroscience.* PhD thesis, University of Pittsburgh, 2017.

[36] Zhou, C., Feng, L., and Zhao, Q. A novel community detection method in bipartite networks. *Physica A: Statistical Mechanics and its Applications 492* (2018), 1679–1693.