UNCOVERING COMMUNITIES IN COMPLEX NETWORKS USING ANT COLONY OPTIMIZATION

CHIN YI HENG

A project report submitted in partial fulfilment of the requirements for the award of Bachelor of Science (Honours) Applied Mathematics with Computing

Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

September 2023

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature : 

Name : Chin Yi Heng

ID No. : 1903338

Date : 8 September 2023

APPROVAL FOR SUBMISSION

I certify that this project report entitled **"UNCOVERING COMMUNITIES IN COMPLEX NETWORKS USING ANT COLONY OPTIMIZATION"** was prepared by **CHIN YI HENG** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Science (Honours) Applied Mathematics with Computing at Universiti Tunku Abdul Rahman.

Approved by,

Signature : 

Supervisor : Chin Jia Hou

Date : 8 September 2023

Signature :

Co-Supervisor :

Date :

ACKNOWLEDGEMENTS

I would like to specially thank Dr. Chin Jia Hou who contributed to the successful completion of this project. I would like to express my gratitude to my research supervisor, Dr. Chin Jia Hou for his invaluable advice, guidance, and his enormous patience throughout the development of the research.

ABSTRACT

Networks often refer to a set of connections between vertices with edges. A network is considered complex if it exhibits complex properties, such as a community structure. Recently, various community detection methods have been proposed by researchers to analyze complex networks. In this research, the Ant Colony Optimization (ACO) algorithm is implemented by incorporating with the Label Propagation algorithm (LPA) to detect communities. The ACO algorithm forms the foundation for initial communities, which are then propagated to become the final communities using LPA. The ACO algorithm has also been extended to handle weighted and directed networks, allowing it to detect communities in such contexts. The performance of the proposed method will be evaluated using different benchmark networks, and the results will be compared with those obtained from existing community detection methods. Furthermore, the proposed method will be extended for implementation in real-world networks to detect communities.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS / ABBREVIATIONS

Sorencen-Dice Index (SDI)

SDI_{ij}	SDI between node i to node j
b_i	Number of connected nodes for node i

Normalize-weight

W_{ij}	Weight of edges from node i to node j
s	Laplace smoothing constant

Probabilistic function

α	Weight of pheromone value
β	Weight of heuristic information
η_{ij}	Heuristic information between node i to node j
τ_{ij}	Pheromone value between node i to node j
θ	Constant value for heuristic information

Fitness score

k_{IC}	Intra-community degree of node
N_r	Set of nodes travelled by ants during tour r

Normalize mutual information

C_x	Number of communities in network x
N_{ij}	Number of nodes determined in exact clustering solutions
N_i	Number of nodes in cluster i

Modularity

m	Number of edges
A_{ij}	Adjacency matrix
$\delta(C_i, C_j)$	Function that return 1 if node i and node j are in the same community

LFR networks

N	Number of nodes
k_{avg}	Average degree of nodes
k_{max}	Maximum degree of nodes
s_{min}	Minimum community size
s_{max}	Maximum community size
μ	Mixing parameter for topology
μ_w	Mixing parameter for edge weights

CHAPTER 1

INTRODUCTION

1.1 Background of study

A network refers to a set of vertices, which are often called nodes, with edges that represent the connection between two vertices. A network is said to be complex when they include unique features such as community structure, scale-free degree distribution, community structure, and small-world property (Chin and Kamali, 2019). Thus, network analysis is carried out to study the relationship between interconnected components or elements inside a network system inside a complex network. By analyzing network systems, it provides new insights for us to have a better understanding of real-world networks such as social networks and biological networks.

Among all the properties of complex networks, community structure is considered an essential feature in real-world networks. According to Girvan and Newman (2002), Community Structure can be defined as the tendency for nodes inside a network to be densely connected within a group of nodes other than with the rest of the nodes in the network. Through analyzing the structure of communities, the relationships between groups of nodes can be identified, as well as recognize the common interests that are shared among these groups within the network (Girvan and Newman, 2002). For instance, studying the community structure in a social network can reveal individuals with common interests or similar behaviors who are more likely to form groups.

Past few decades, researchers have proposed various community detection methods for the study of community structure. In the year 2004, Newman and Girvan (2004) have proposed the first community detection method which is modularity optimization and it has attracted people's attention. The concept of modularity is introduced as the measure of the quality of detected community structure in a complex network, while the modularity optimization method has been proposed as a method to detect communities in a network by maximizing the modularity. In the meantime, the performance of the proposed method was evaluated by comparing it with other community detection methods and result in good accuracy and efficiency.

In the year 1999, Dorigo and Di Caro proposed a new metaheuristic approach called Ant Colony Optimization (ACO) which takes inspiration from the foraging behaviors of ants. In the ACO algorithm, the pheromones will be deposited on the optimal path to label them for the other ants. Apart from that, the authors have also utilized the ACO algorithm to tackle the Travelling Salesman Problem (TSP) by implementing the ACO algorithm to search for the optimal solution (Dorigo and Di Caro, 1999). The algorithm produced a remarkable result in solving the TSP and other combinatorial optimization problems. In recent years, metaheuristic-based community detection methods have become more and more popular. For instance, Wang et al. proposed a community detection method that involves adjusting the number of ants parameter dynamically in ant colony optimization to enhance the efficiency of the approach (Wang et al., 2020). In addition, Hosseini et al. proposed an advanced Label propagation algorithm that incorporates ACO with a single objective modularity optimization for community detection in complex networks (Hosseini et al., 2020). These successes have revealed the potential of the ACO algorithm, leading to the continued development of ACO to deal with other real-world problems.

1.2 Problem statement

In network analysis, community detection has played an important role, which aims to identify the groups of nodes with similar properties or attributes. In this paper, a community detection method incorporated with Ant Colony Optimization (ACO) was proposed. In fact, several studies have been conducted by other researchers on ACO approaches, but most of them mainly focus on heuristic information. Undoubtedly, heuristic information has become crucial in the development of the ACO algorithm, and various approaches have been proposed based on the network that is being analyzed. Thus, an ACO approach will be proposed in this paper by optimizing the parameters of the ACO algorithms as well as the allocation of ants in the algorithm.

1.3 Aim and Objectives

In this research, we aim to develop a community detection method by incorporating Ant Colony Optimization algorithms with the Label Propagation

algorithm. Additionally, the parameter settings of the ACO algorithm will be improved to enhance the algorithm's effectiveness in detecting communities. These parameters include ant population size, ant’s initial allocation, pheromone value, heuristic information, alpha and beta values, and the number of iterations. Finally, the proposed method's performance will be evaluated against benchmark networks, and its implementation will be further extended to real-world networks.

1.4 Significance of study

In the development of our ACO-based community detection method, the parameters of the ACO algorithm have often received limited attention within existing literature. This gap in the literature highlights the possible improvement of existing ACO-based community detection. Thus, this study is conducted to enhance the existing ACO-based algorithm, extending its applicability to weighted and directed networks.

Our research will focus on investigating the different parameter settings of the ACO algorithm, aims to improve the performance of the ACO algorithm in terms of the detection accuracy. The research also contributes to an understanding of importance of parameter in ACO algorithm.

Furthermore, the proposed ACO-based algorithm serves as a foundational component for existing community detection methods. In this research, the ACO algorithm will incorporate with Label Propagation Algorithm to detect the communities in complex networks. Finally, the proposed method may offer a valuable point of reference for researchers in developing an advanced ACO-based community detection method.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

In the year 1999, Dorigo and Di Caro (1999) have proposed an ant colony optimization approach to solve the famous Traveling Salesman Problem (TSP) which is a combinatorial optimization problem. The proposed method has opened up a new meta-heuristic approach and it can be also applied to other combinatorial optimization problems, including community detection in a complex network. Therefore, we are going to review various papers on ACO-based approaches as well as their parameters.

2.2 Literature Review

In this section, we seek into various aspects of community detection method that incorporated with ACO algorithm. This subsection below highlighted both traditional and innovative approaches in detecting communities by offering a comprehensive exploration of these methodologies.

2.2.1 Pearson Correlation as heuristic value of ACO algorithm

In the year 2013, Chang et al. proposed a community detection method using ACO. The paper has emphasized the modularity optimization approach, which takes the maximization of modularity, a measure of the quality of the detected communities in a network, as an objective of the study. In addition, the author has specified the benefits of the approach, which does not require any prior knowledge about the total number of communities within the network (Chang et al., 2013). The ACO-based community detection was proposed based on the max-min ant system (MMAS) framework. MMAS is a variant of the ACO algorithm which function as balancing the exploration and exploitation of ants during the search process by limiting the maximum and minimum value of pheromone trails in order to avoid stagnation. The algorithm records every community partitioning solution using local-based adjacency representation. A new heuristic approach is employed in the algorithm and they are adapted to fit the requirement of the community detection method.

In the solution construction section, a probability function based on pheromone trails and heuristic information was formulated. In other words, the probability makes use of pheromone trails and heuristic information to indicate which edges will be chosen as the solution component. In the meantime, parameters alpha and beta will be used to determine the weight of the pheromone trail and heuristic information in the probability function.

Besides, the proposed method takes the similarity between two vertices as heuristic information and they are measured using Pearson correlation. The higher similarity will be resulting in a value closer to 1 while the lower similarity will result in a value closer to -1. Since the range of the Pearson correlation was between -1 and 1, a logistic function was applied to avoid the negative value outputs.

Furthermore, the author has also mentioned the importance of pheromone trail management. The proposed method followed the MMAS mechanism, in which all pheromone trails are set to a high value to have a better exploration of the solution at the initial phase of the search process. At the same time, the pheromone value will also be adjusted in each of the iterations. The pheromone will only be updated when solutions are constructed by all ants in each iteration. At first, the pheromone trails will be decreased by a fixed value which is commonly known as pheromone evaporation, in order to guide the ants in solution exploration. The process is followed up by a reinforcement of the solution, which deposits pheromone on the edges of the network during the iteration that contains an improvement in the modularity value. The range of pheromone value will be limited in order to avoid stagnation.

On top of that, the author has also highlighted that optimizing the parameters for ACO has exceeded the scope of the study. Therefore, the parameters of the ACO algorithm were set to be a fixed value of alpha, beta, and pheromone evaporation value, and the maximum number of iterations to be 1, 2, 0.8, and 100 respectively. Last but not least, the performance of the proposed method was evaluated by adopting modularity and normalized mutual information (NMI) evaluation methods. The proposed method was tested on the real-life network by comparing it with other community detection

algorithms and showed good performance in terms of modularity while having a satisfactory result in terms of NMI.

The author has demonstrated the potential of the ACO approach in detecting communities. However, it should be noted that the paper primarily focuses on static networks. In future work, the author suggests applying the ACO approach to real-world social networks, which are dynamic networks, to explore their potential in detecting communities in different types of networks.

2.2.2 ACO approach in detecting initial distinct communities

In the year 2019, Chin and Kamali proposed a new community detection method with an ant colony approach to detect initial partial communities in a complex network. The proposed method aims to identify the initial partial communities in both unweighted and undirected networks by utilizing the ACO algorithm (Chin and Kamali, 2019). In the algorithm, the ants that follow the routes will highest fitness score, and the nodes along the routes will form a community. Once the detection of initial communities is completed, the nodes will be combined and form meta-nodes hence turning the original network into a new network. The new network will then be applied with other community detection methods thus completing the community detection.

At first, the author introduced the concept and idea of the ACO algorithm used in the proposed method. The ACO algorithm is initiated by determining the number of ants used based on the total number of nodes in a network. The ants' allocation is then determined by comparing the degree of a node to the average degree of other nodes. The ants will be allocated to the nodes with a higher ratio of the degree of the node. Apart from that, the author has defined a probability function based on the pheromone value and heuristic value for the movement of ants in the network. The weight of the pheromone value and heuristic value in the ACO algorithm is controlled by the parameters alpha and beta, respectively. The proposed method utilized the Sorencen-Dice Index (SDI), which is the measurement of similarity between nodes, as the heuristic values of the probability function. It is important to note that if either one of the pheromone values and heuristic value is equal to zero will result in 0 probability indicating that the ants will never travel through the path. Therefore, a constant value was added to the heuristic value and the

pheromone value is set to be 0.2 at the start of the algorithm to avoid 0 probability. The decision of ant movements will be decided by a Roulette function based on the calculated probability. The ants will move a number of steps to be considered as finished its tour. After the ants have finished their tour, the visited nodes will be used to calculate their fitness score. If the fitness score is found to be improved compared to previous tours, the pheromone value will be modified by a constant value rho. The movement of ants, fitness score calculation, and pheromone update will be repeated until they reached convergence. At the end of the process, the best routes and fitness score of the tour for each of the iterations will be stored in two separate lists for future labeling uses. The entire process is repeated several times and the result is stored inside the lists to obtain multiple solutions.

Once the lists are completed, the routes with fitness scores that are above average will only be considered in the node labeling process. In the first phase of the process, the nodes in the route with the lowest fitness score are labeled with the starting node's label. This is repeated for routes with higher fitness scores, and nodes that appeared multiple times will consider the label of highest fitness score routes. In the second phase, the process borrows the idea of label propagation algorithm (LPA) which takes the label that occurred most frequently as the label of all the nodes in the routes. The process starts from the highest to lowest fitness score to prioritize the higher fitness score solutions. In the final phase, the nodes with identical labels will be classified as a community and the community with fewer than 3 nodes will not be considered. The nodes in the detected community will be combined into one node and the weighted edges can be determined by aggregating the intra-community edges, hence, forming a new weighted network. Lastly, the existing community detection method will be used to detect the distinct communities of the new weighted network.

Last but not least, the performance of the proposed method was evaluated with normalized mutual information (NMI) and modularity (Q). The capability of the proposed method was tested with Lancichinetti-Fortunato-Radicchi (LFR) synthetic network and real-world networks. The author has shown that the proposed method is capable of detecting a good quality of initial communities. The proposed method will then be compared with the

existing community detection method without forming meta-nodes. Although the proposed method's results show slightly lower performance compared to existing community detection methods, it is capable of preventing trivial detections. The author has also emphasized the improvement that can be made to the proposed method such as refining the solution before the formation of meta-nodes to improve the quality of the initial detection of the community.

2.2.3 Ant's population adjustment for ACO-based community detection

In the year 2020, Wang et al. proposed a community detection method incorporate with ACO. In the proposed method, the number of ant parameters in ACO is focused. According to Wang et al., the number of ants used in the ACO algorithm has a significant impact on the time complexity and the searching efficiency of the algorithm (Wang et al., 2020). Thus, the number of ants will be investigated in order to improve the ACO algorithm in terms of both computational cost and the time complexity of the algorithm.

In this research, the proposed method begins by initializing a probability matrix for the pheromone trails. The transfer probability formula is utilized to compute the probability matrix of each node. The matrix will be updated based on the local solutions and global solutions constructed by the ants. Finally, the completed matrix is then used for community detection, where the community of the node is identified by the columns with the highest probability in the probability matrix.

Besides that, the author has also dynamically adjusted the number of ants at the end of each iteration based on certain conditions. The author highlighted that the ants might be inactive during the process of community detection. In other words, the ants may get trapped in local optima, providing no improvement in the solutions but still consuming the computation resources of the algorithm. Therefore, the number of ants is modified dynamically based on a certain condition during the process. When the quality of solutions remains stable, the number of ants will be increased to help the ants from escaping the local optima, whereas when the quality of solutions shows a slight improvement, the number of ants will be reduced to decrease computational costs. Finally, when the quality of solutions improves

significantly, the number of ants will be increased significantly to enhance the search for better solutions.

Last but not least, the author has carried out several experiments to investigate the impact of the number of ants on community detection. The proposed method is compared with the existing ACO-based community detection method and the performance of the algorithm will be evaluated as well. The result clearly showed that the proposed method has a significant improvement in terms of the computational time of the algorithms. However, the quality of the solutions obtained from the results only has slight improvement, indicating a need to overcome this issue. Overall, the proposed method is a promising approach for community detection as they reduce the computational cost of the algorithm effectively while maintaining the quality of solutions constructed.

2.2.4 Other state of the art community detection method

In the year 2007, Raghavan et al. utilized the label propagation method to propose a localized community detection algorithm. Initially, the process starts with labeling the nodes with unique labels. At each iteration of the process, each node will replace its label with the label that occurred most frequently among its neighboring nodes. After a few iterations, the labels will start to converge hence forming the communities.

Furthermore, the author has explained in detail how the label propagation algorithm can be applied to detect community structure (Raghavan et al., 2007). In LPA, the nodes will tend to join the community that is formed by most of its neighbor nodes. If the same maximum number of times among its neighbors occurred, the node will randomly select one of those nodes to join their community. Initially, each node will be given a unique label. The label will spread through the network and the nodes that are densely connected to each other will be more likely to convert their label into a common label hence forming groups. The groups will continue to grow as the process go until no changes were made which are said to be converged. In the end, the nodes that share the same label are grouped thus forming a community.

In addition, Raghavan et al. also highlighted the advantage of the proposed method which the algorithm does not require any prior knowledge about the network structure, thus making it easy to be implemented onto a variety of networks. Besides, the performance of the proposed method was evaluated using a real-world network with predefined communities. According to the results obtained, the proposed method is capable of finding the community structure effectively.

CHAPTER 3

METHODOLOGY AND WORK PLAN

This chapter provides an in-depth exploration of the application of the ACO algorithm incorporating with Label Propagation Algorithm (LPA) to detect communities. The chapter covers various aspects of this approach, including parameter setting, solution construction, ACO labeling, incorporation with the LPA, evaluation criteria, datasets used, and the tools and software used in the research. Also, this chapter aims to present a comprehensive overview of how ACO can be leveraged to enhance the performance of community detection by incorporating with LPA.

3.1 Introduction of ACO

Ant Colony Optimization (ACO), is a metaheuristic algorithm that takes inspiration from the foraging behavior of ants. The pheromone will be deposited on the optimal path to the food to create a label for other ants. In ACO, a number of artificial ants are generated to search for the optimal solution to a given problem. Each of the ants will build an independent solution by iteratively searching for a solution based on the pheromone trails left by previous ants. The pheromone trails will be updated at the end of each iteration based on the quality of the solutions found. In the end, the algorithm is able to converge on the best solution due to the feedback of pheromone trails and the reinforcement of ants.

In this project, the algorithm is implemented to detect initial communities, perform labeling, and complete the detection by incorporating with existing community detection method. The performance of the algorithm will be evaluated to benchmark its capabilities in detecting communities. The process is structured with a sequence that includes parameter settings, solution construction, node labeling, complete detection, and performance evaluation. The proposed algorithm's framework is derived from the work of Chin and Kamali (Chin and Kamali, 2019). Their paper served as the foundation for the development of the algorithm.

3.2 Parameter settings

In the ACO algorithm, our first step involves initializing all the essential parameter settings for the ACO algorithm. This section outlines the key parameters that play a crucial role in the proposed method for initial community detection. Effective tuning of these parameters is essential for achieving optimal results. The subsections below discuss the specifics of each parameter and their significance and impact on the overall algorithm performance.

3.2.1 Ant population size

Regarding the ant population size, the performance of the ACO algorithm can be affected by the number of ants used in the algorithms. The number of ants should be sufficient to effectively cover the exploration of the solution space. Theoretically, a bigger ant population size will provide a more efficient search and increase the convergence speed. However, it will result in increased computational time and violate the principle of efficiency, which states that metaheuristic algorithms should be superior in terms of both accuracy and efficiency. On the other hand, if the ant population size is found to be small, there is a chance of missing potentially good solutions, as the ants cover insufficient paths leading to premature convergence to local optima. Thus, it is important to balance the size of ant populations in order to achieve desirable results.

3.2.2 Ants allocation and steps

In the community detection method, the initial allocation of ants can significantly affect the algorithm's speed and performance. The allocation of ants will be carried out based on the ratio of a node's degree to the average degree of its neighboring nodes. A higher degree of nodes will have relatively more connections between nodes within the network, indicating that these nodes play a more crucial role as they potentially carry key solutions. In addition to that, allocating ants to well-connected nodes can help the ants to escape local optima by utilizing the alternative paths especially when dealing with complex networks. Regarding the degree of nodes, only the outgoing degree will be considered as it reflects their ability to influence other nodes

within the network. Identifying effective influencer nodes can significantly affect the detection of initial communities as they have a high potential to impact other nodes.

Besides that, the artificial ants will move through the network within a fixed number of steps to construct a solution, with the ants that complete their steps considered to have completed a tour. The purpose of limiting the number of steps is to prevent taking too long to find a solution. Therefore, the number of steps should be small to ensure the initial communities formed remain small. This approach prevents the dominance of individual communities and leads to the formation of trivial solutions.

3.2.3 Pheromone value

Pheromone values play a crucial role in the movement of ants and are considered an essential component in the ACO algorithm as they guide the ants toward optimal solutions. However, if the ants over-rely on pheromones, this may limit the ants' ability to explore alternative paths and find potentially better solutions. The bias in exploiting the current known path may hinder the algorithm's ability to discover other possible optimal solutions. To address the issue, we may initiate the pheromone value at a low level at the early stage of the algorithm and increase the value at the later stages. This approach allows the ants to explore various paths, promoting a wider search for optimal solutions, and enabling them to shift the focus towards exploitation in order to reach convergence at the later stages.

3.2.4 Heuristic information

Aside from the pheromone value, the heuristic information played another important role in the movement of ants within the ACO algorithm. The heuristic value provides additional information about the problem at hand and helps ants in making more informed decisions by considering factors other than just pheromones. In this research, the Sorencen-Dice-Index (SDI) served as the heuristic information and was utilized to measure the similarity between two nodes. The SDI between the nodes and can be expressed in the following equation:

$$SDI = \frac{2|b_i \cap b_j|}{|b_i| + |b_j|} \tag{3.1}$$

where b_i and b_j are the total number of connected nodes for node i and node j respectively. The $|b_i \cap b_j|$ represents the number of mutual connected nodes of i and j. The connected number of nodes for b_i will not consider b_j as the connected node when calculating the SDI and vice versa.

On top of that, the SDI can also be incorporated with the weights of the edges in weighted networks to enhance the heuristic information's ability to navigate ants in exploiting the network effectively. The weights help ants in determining the significance of connections between nodes, providing a more comprehensive meaning to differentiate the importance of various paths. This approach can be accomplished by multiplying the Sorensen-Dice Index (SDI) with the normalized weights. As the normalization method ranges from 0 to 1, a Laplace smoothing technique is implemented to avoid the 0 probability. The normalized weight of edges from node i to node j can be expressed in the following equation:

$$NW_{ij} = \frac{W_{ij} - \min_j\{W_{ij}\} + s}{\max_j\{W_{ij}\} - \min_j\{W_{ij}\} + s} \tag{3.2}$$

where W_{ij} represents the weights of edges from node i to node j. The $\min_j\{\}$ and $\max_j\{\}$ functions select the minimum and maximum values among a set of values indexed by j, respectively. The parameter s represents the Laplace smoothing value used to prevent 0 probability in normalized weights. Thus, the heuristic information of edge from node i to node j can be expressed in:

$$\eta_{ij} = SDI_{ij} * NW_{ij} \tag{3.3}$$

where SDI_{ij} are the SDI between the node i and j and NW_{ij} are the normalized weight of edges for node i to node j. For unweighted network, the NW_{ij} will be equal to one as they have no weights between the edges.

3.2.5 Alpha and Beta value

In the ACO algorithm, alpha and beta are two key parameters that influence the algorithm's behavior and strike a balance between exploration and exploitation in the computation of ant movements. Alpha, often referred to as the pheromone influence factor, determines the weight assigned to pheromone information. A higher alpha value places greater emphasis on exploiting known optimal solutions, which speeds up convergence but may also lead to premature convergence to suboptimal solutions. On the other hand, beta, known as the heuristic influence factor, regulates the importance of heuristic information. A higher beta value encourages greater exploration of the search space, promoting the discovery of new, and potentially better solutions. Therefore, finding the proper balance between alpha and beta values is crucial for optimizing algorithm performance.

3.2.6 Number of iterations

In this project, the number of iterations used for ant's enhancement (inner loop) and reiterating the process (outer loop) is set to be a small value. Despite the number of iterations being limited, the number is sufficient to yield satisfactory detections of initial communities for most of the networks. However, increasing the number of iterations may increase the computational times without increasing the quality of the initial community's detections. Thus, the values are intentionally set to sufficiently small while still maintaining their functionality. In the future, the number of iterations will be taken into consideration to improve the algorithm's efficiency.

3.3 Solution construction

Once the parameters have been set up, the ants start to travel around the nodes within the network to construct solutions. This section provides an overview of the iterative process underlying the proposed community detection algorithm.

3.3.1 Ant's movement

As detailed in section 3.2, the ACO algorithm initiates ants in nodes with a high ratio degree of nodes. Once initiated, the ants begin to travel to construct solutions by seeking optimal routes. The movement of ants is guided by a probability function, but rather than solely relying on the highest probability, the selection of nodes to travel is determined using a roulette function. A roulette function generates random outcomes based on specified probabilities assigned to each possible outcome, covering a variety of different outcomes for a particular solution. This mechanism encourages ants to explore different paths rather than always choosing the path with the highest probability, which leads to getting trapped inside a local optima. Therefore, by fitting the probabilities for the possible traversable nodes into the roulette function, it will determine the next node for the ants to travel to. The probability function can be expressed using the following equation:

$$P_{ij} = \frac{(\tau_{\mathrm{ij}})^{\alpha}((\eta_{ij}+\theta))^{\beta}}{\sum_{l\in N_i}(\tau_{il})^{\alpha}(\eta_{il})^{\beta}} \tag{3.4}$$

where τ_{ij} is the pheromone trail value between component i and j, while η_{ij} is the heuristic information between i and j. The parameter α controls the weight given to the pheromone information, while the parameter β controls the weight given to the heuristic information. N_i represents the group of connected components while θ is a constant value.

3.3.2 Ants performance evaluation

Once the ants have completed their tour, the algorithm uses the nodes that the ants visited to calculate the fitness score for their respective solutions. The fitness score for tour r can be expressed in the following equation:

$$f(r) = \sum_{Z \in N_r} k_{IC}(z) \quad (3.5)$$

where N_r represents the nodes that ants visited during tour r, while the intra-community degree of a node is denoted by k_{IC}. The fitness score of routes will be used in the later section of pheromone updates and the ants' population adjustment.

3.3.3 Pheromone value update & Adjustment of ant's population

Upon the completion of the fitness score calculation, the pheromone value will be updated based on the performance of ants. Specifically, ants with fitness score improvement compared to the previous iteration will receive an increment of a small constant value to all of their corresponding edges within that particular route. This approach aims to reinforce the paths taken by ants that have contributed positive feedback to the algorithm, hence increasing the probability of being selected in the following iterations.

At the same time, those ants with unsatisfactory fitness scores will be selected for the next iterations, giving them an opportunity to improve their performance. This approach not only improves the overall performance of ants but also decreases the unnecessary computational times at the same time. This is because the movement of ants are decided randomly with a roulette function, and there are chances were the ants performed well in current iterations may yield bad result in the next iterations. Thus, we decided to retain well-performing ants and exclusively reinforce those ants that do not achieve the target in the upcoming iterations.

3.3.4 Generating multiple routes

When the ants have reached the maximum time of reinforcement, the best fitness score and its corresponding route among that iteration's results are stored in a final list. The updated pheromone value will be reset to the initial pheromone value and the entire process will be repeated to generate different routes and fitness scores, they will be stored in the final list for labeling purposes.

3.4 ACO Label

Once the final list has been fully updated, the routes with fitness scores greater than the average fitness score will only be considered in the labeling process. The process starts by labeling the nodes in the routes with the lowest fitness score by the first nodes' label. The process is repeated from the lowest fitness score to the highest fitness score. The label will be replaced by a higher fitness score label if a node carries multiple routes. After labeling all the nodes, nodes with the same label are treated as the initial community. The figure below illustrates the labeling process of nodes.

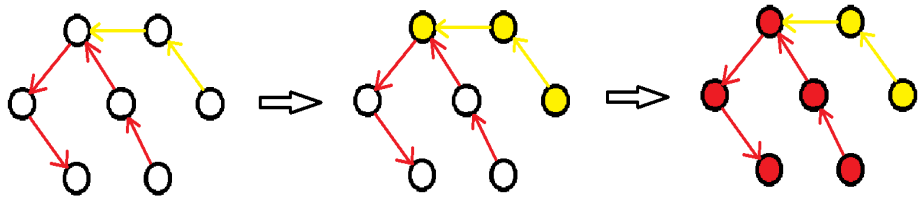

Figure 3.1: The example of the process of the first phase of the labeling process.

The yellow color routes have a lower fitness score as compared to the red color routes. Thus, the yellow label is replaced by the red label.

3.5 Complete detection using Label Propagation Algorithm

Label propagation algorithm (LPA) is a community detection algorithm that aims to identify communities in a complex network. The concept of LPA is to propagate the labels among the nodes in a network based on the similarity of their connections. LPA is also known as a fast algorithm due to its simple and efficient label propagation process. The process is said to be simple because the process only involves determining the similarity between nodes, and updating the node labels accordingly and iteratively until convergence has been reached. The computational efficiency of LPA allows it to be easily implemented on a large-scale network.

In the proposed method, LPA is implemented in the initial communities formed by the ACO algorithm. The labels are propagated for a number of iterations or until no changes during the propagation process.

Finally, the nodes with the same propagated labels are treated as final communities.

3.6 Evaluation Criteria

In this section, we will discuss the evaluation criteria used to assess the performance and effectiveness of the proposed algorithm. Specifically, we will focus on two key metrics: Normalized Mutual Information (NMI) and Modularity. These metrics provide insights into the quality of the detected community and the overall robustness of our method. The performance can be benchmarked by comparing these evaluations with the existing community detection methods.

3.6.1 Normalize mutual information

NMI is a measure used to evaluate the similarity between two communities in a complex network. NMI is typically used in clustering evaluation to evaluate the quality of the clustering. In this paper, NMI can be applied to evaluate the similarity between two detected communities obtained by the proposed method. The NMI between network X and network Y can be expressed in the following equation:

$$NMI(X|Y) = \frac{-2\sum_{i=1}^{C_X}\sum_{j=1}^{C_Y} N_{ij}\left(log\frac{N_{ij}N}{N_{i.}N_{.j}}\right)}{\sum_{i=1}^{C_X} N_{i.}\left(log\frac{N_{i.}}{N}\right)+\sum_{j=1}^{C_Y} N_{.j}\left(log\frac{N_{.j}}{N}\right)} \tag{3.3}$$

where the total number of community in network X and Y are denoted as C_X and C_Y , while the sum of the rows i and sum of the columns j are denoted as $N_{i.}$ and $N_{.j}$ respectively. NMI ranges between 0 and 1, where the value close to one indicates the communities are identical and close to zero indicates the communities are different from each other.

3.6.2 Modularity

Modularity is a measure of the quality of a community structure within a network. Generally, modularity is used in community detection to evaluate the performance of the algorithm. In other words, it measures the degree of

detected communities within a network. The modularity, Q can be expressed in the following equation:

$$Q = \frac{1}{2m}\sum_{i,j\in V}(A_{ij} - \frac{k_i k_j}{2m})\,\delta(C_i, C_j) \tag{3.4}$$

where the set of nodes represented by V and the total number of edges are represented by m. The adjacency matrix is denoted by A_{ij}, where the value will equal to 1 if node i and node j are connected, and otherwise 0. The degree of node i and node j are denoted by k_i and k_j respectively. The $\delta(C_i, C_j)$ function will be equal to 1 if node i and node j falls in the same community, otherwise 0. In general, the modularity equation ranges between 0 to 1, where the value closer to one indicates the nodes inside a community are densely connected to each other while the value close to 0 indicates the nodes inside a community are not densely connected to each other.

3.7 Dataset

The proposed method will be tested using both artificial benchmarking networks and several real-world networks. The real-world network datasets are available from the KONECT open-source website. For the artificial benchmarking network, the Lancichinetti–Fortunato–Radicchi (LFR) network model is used to create artificial benchmarking networks for evaluating the performance of community detection algorithms. The LFR model will generate networks with pre-defined community structures, enabling us to easily compare and evaluate performance.

Besides, the real-world networks can be classified into two categories, those with pre-defined community structure and those without. The networks with pre-defined community structures are used to test the accuracy of community detection for a given method while those without pre-defined community structures will require additional exploratory analysis to detect the communities.

3.8 Tools and software

In this project, the R programming language is utilized as the primary programming language in the development of the ACO algorithm. One of the reasons that R programming languages are selected is because R has a significant amount of library packages that can be capable of statistical computing, data analysis, as well as data visualization. Meanwhile, Microsoft Excel is utilized to prepare the raw data and convert it into CSV files as the R programming can easily understand the data in CSV format. Then, the CSV files will be imported into R programming to insert the network data.

CHAPTER 4

COMPUTATIONAL DETAILS

In this chapter, we delve into the core components that form the foundation of our project. Each subsection addresses a critical aspect, providing essential insights and details to help readers understand the structure of our work.

4.1 Datasets

In this project, the utilized datasets include both benchmark and real-world networks. Regarding the types of networks, four types of networks will be included such as unweighted undirected, unweighted directed, weighted undirected, and weighted directed networks. The algorithm is tested with benchmark networks to evaluate its performance in detecting communities. This is because the presence of ground truth within the benchmark networks allows for a better comparison between the outputs and the actual community structures. The evaluation framework is then extended to real-world networks, which may lack of established ground truths.

4.1.1 LFR benchmark network

For benchmark networks, the Lancichinetti–Fortunato–Radicchi (LFR) benchmark network is utilized to evaluate the capabilities of various algorithms. The network model offers the advantage of known ground-truth community structure, allowing more precise evaluation for community detection algorithms. In addition to that, the model is also capable of replicating real-world network complexities such as power-law degree distributions and overlapping communities, thus enhancing its relevance in algorithm testing.

In order to generate LFR benchmark network, there are some of the necessary parameters required to be set up:

- Number of nodes, N
- Average degree of nodes , k_{avg}
- Maximum degree of nodes, k_{max}

- Minimum communities size, s_{min}
- Maximum communities size, s_{max}
- Mixing parameter of topology, μ
- Mixing parameter of weights, μ_w

The size of the network generated by the LFR benchmark will be decided by the parameter N which sets the total number of nodes in the networks. The degree of nodes refers to the number of connections (or edges) for a particular node, thus, setting up the average degree of nodes determine the average number of connection for a node while the maximum degree of nodes indicates the maximum number of connection for a node to have.

Moreover, the range of community size can be established by setting up the minimum and maximum community size. If both of the values are set relatively low, the LFR benchmark will generate a network consisting of small communities, leading to a high number of distinct communities. On the other hand, if the value is set higher, a network consisting of large communities will be generated, resulting in fewer number of distinct communities.

Apart from that, the mixing parameter of topology stands as a key factor in shaping the complexity of network structure when generating LFR benchmark networks. This parameter influences the degree to which nodes from different communities are connected to each other. In simple words, the parameter adjusts the degree of nodes, which the connection comes from the node within the same communities or different communities. When the mixing parameter of topology is set to be low, it indicates that nodes tend to connect more nodes that came from the same communities, resulting in a more distinct community structure. In other words, the communities are well-defined and relatively isolated from each other. Conversely, if the parameter is set to be high, nodes are more likely to have connections that bridge between communities, resulting in more interconnection, where nodes from different communities are more entangled.

In order to generate a weighted network, the mixing parameter of weights has to be set to a value higher than 0. This parameter determines the distribution of weights assigned to the connection between nodes. When the parameter is set to a lower value, the weights are more evenly distributed

among the networks, while on the other hand, the weights will be unevenly distributed when the parameter is set to a higher value. Last but not least, the unweighted network can be generated by setting the parameter to 0.

4.1.2 Real world networks

In community detection, methods are often tested with real-world networks as they reflect the complexities and actual systems, making them more relevant to real-life applications. Evaluating an algorithm with real-world networks helps to determine the algorithm's ability to generalize across different scenarios and datasets. In order to have a better evaluation, the algorithms are tested with famous real-world benchmark networks, including the karate, dolphin, football, and political book networks. Finally, the evaluation process is extended to real-world networks to demonstrate the algorithm's effectiveness in addressing real-world scenarios. The datasets for real-world networks were downloaded from an open-source website known as ‘Konect’.

4.2 Parameters

In this section, the parameters of ACO algorithm will be details and discuss. The following is a list of the parameters utilized in ACO algorithm:

- Initial ant population size
- Alpha and Beta value
- Initial pheromone value
- Pheromone update value
- Number of steps
- Number of iterations for inner loop and outer loop

Firstly, we begin the algorithm by setting the initial ant population size as half of the total nodes in the network. The values are quite decent as they have balanced between exploration and exploitation. After the first iterations, the ant population will be adjusted based on their performance.

Besides that, the alpha and beta values in the probability function are set to 0.5 for both parameters considered a common practice in various algorithms. An equal weight assigned to these parameters strikes a balance

between exploration and exploitation, ensuring the algorithm doesn't favor one aspect over the other. In addition, an equal weights assignment makes the algorithm more user-friendly and less sensitive to variations in data inputs.

Furthermore, the initial pheromone value is set to 0.2 which is considered a lower value. This helps ants avoid biases and encourages ants to explore the paths during the early stages of the algorithm. At the end of each iteration, if the ants find that they have improved their fitness scores compared to previous iterations, the pheromone value for all the edges along the path will be updated. The pheromone update value is set to be increased by 0.05. This low update value helps the algorithm maintain a balance between exploration and exploitation by preventing rapid and excessive pheromone accumulation on certain paths. By slowly increasing the pheromone levels, the algorithm gives sufficient time for ants to explore and evaluate different paths, reducing the chances of getting stuck in suboptimal solutions.

At every iteration, the ants will move 4 steps and they are considered completed their routes. The decision to have ants move only 4 steps is aimed at intentionally keeping the size of the communities small. These communities are not final solutions but rather starting points. This design encourages focused exploration within specified limits, enabling efficient search for solutions.

Finally, the number of iterations for both the inner loop (ant's enhancement) and the outer loop (reiterating the process) is limited to a maximum of 10 iterations. The values achieve a balance between seeking a satisfactory result in the initial community detection phase and maintaining the algorithm's speed at an optimal level. This approach prioritizes efficiency without compromising the quality of the outcomes.

4.3 Code and Implementations

In this project, the ACO algorithm is implemented using the R programming language, utilizing its powerful data analysis capabilities. The algorithm is divided into four subsections for detecting initial communities.

4.3.1 Libraries and data

Firstly, the initial section begins by importing necessary libraries, such as the "igraph" and "doparallel" packages. The "igraph" package in R is a robust tool for analyzing graphs and networks, offering features like graph creation, analysis, visualization, and data import/export. Additionally, the "doparallel" package allows the utilization of multiple CPU cores, which can enhance computational speed. However, it's worth noting that parallelization may not always result in improved performance, as it can increase the time and resources required for setting up parallel components, potentially leading to worse performance. After importing the libraries, the network can be loaded from edge list format files. Also, variables need to be adjusted based on the type of network imported. If ground truth networks are available, they can also be imported as well.

4.3.2 Parameter setup

The next section involves setting up the necessary parameters for the entire process. Initially, each node is assigned a unique label for differentiation. Then, node degrees are calculated using the "degree" function from the "igraph" library. For directed networks, only the out-degree is considered, as it reflects a node's ability to influence other nodes. Additionally, the SDI for each edge in the edge list is calculated using the formula in Section 3.2. For weighted networks, SDI calculations differ, as edge weights affect heuristic information. The edge weights will be normalized using min-max scaling, and a small Laplace smoothing constant is added. The normalized weights are then incorporated into the heuristic information. Furthermore, the ratio of a node's degree to the average degree of its neighbors will be calculated. The ants will then be assigned to the nodes based on the calculated ratio of the degree of nodes. Lastly, parameters such as the number of steps, alpha and beta values in the probabilistic function, ant population size, and the number of iterations for ant generations and ant enhancement are set according to the value suggested in Section 4.2.

4.3.3 ACO algorithm

Moving on, the following section delves into the ants, which are the core components of the ACO algorithm. To provide a clearer explanation, parts covering multiple iterations are illustrated using curly brackets.

[Outer loop]{

The ACO algorithm begins after initializing the ants at nodes with a high degree ratio. Pheromone values are assigned to every edge and reset to the initial pheromone value at the start of each outer loop iteration. Then, the ants start constructing solutions, which undergo multiple iterations of enhancement within the inner loop.

[Inner loop]{

If this is the first iteration of the inner loop, the number of ants is set to half the size of the network to create routes. Starting from the second iteration, the ant population size is adjusted based on the performance of ants in the previous iteration. Ants with fitness scores below a threshold are selected to undergo reinforcement in the next iteration. Each ant constructs independent routes within a specified number of steps.

[Ants' movement]{

Each ant is designed to create routes within a maximum number of steps. During solution construction, ants choose their next steps using a probabilistic function to determine their destination. The probabilistic function is calculated before ant movement and updated with each iteration. The roulette function is then implemented in the probabilistic function to select a node to travel to from its neighboring nodes.} [End of ants' movement]

Ants continue moving until they reach the maximum number of steps to create a route.

Once ants have completed their steps, the fitness score of the routes they created is computed and stored in a list.} [End of inner loop]

The ant enhancement process iterates several times. If ants find a route with a better fitness score, the current fitness score replaces the list storing the best fitness score for that particular route. Additionally, the pheromone values for all edges in the route are updated by a constant value. When ants have reached

the maximum number of enhancement iterations, the routes, and their fitness scores are stored in a final list.} [End of outer loop]

The entire process runs multiple times to populate the list with different results.

4.3.4 Node labelling

Finally, the last section is the node labeling process. In this step, routes with fitness scores lower than the average are removed from the list, and the remaining routes undergo labeling. All nodes within a route are assigned the label of the first node in the route. The label replacement is performed from the lowest fitness score to the highest fitness score. Nodes with the same label are treated as communities. If a route has fewer than three nodes, its community is disbanded, and its labels are replaced with -1. Note that label removal does not reset the labels to their initial unique labels. Instead, they are set to -1. The ACO algorithm is now complete, and the nodes with the same labels will be treated as initial communities. The labels will then propagate by implementing LPA to complete the community detection.

4.4 Evaluation process

As elaborated in section 3.6, the algorithm's performance can be assessed through the use of two key metrics: Normalized Mutual Information (NMI) and modularity. NMI determines the accuracy of the algorithm when predefined networks are available, while modularity evaluates the structure of the detected communities. The finalized detection labels will be compared with the predefined labels using the "igraph" library's comparison function by assigning the parameter “method” equal to “nmi”. Meanwhile, the modularity of detected communities will be evaluated using the modularity function from the 'igraph' library.

4.5 Process flowchart

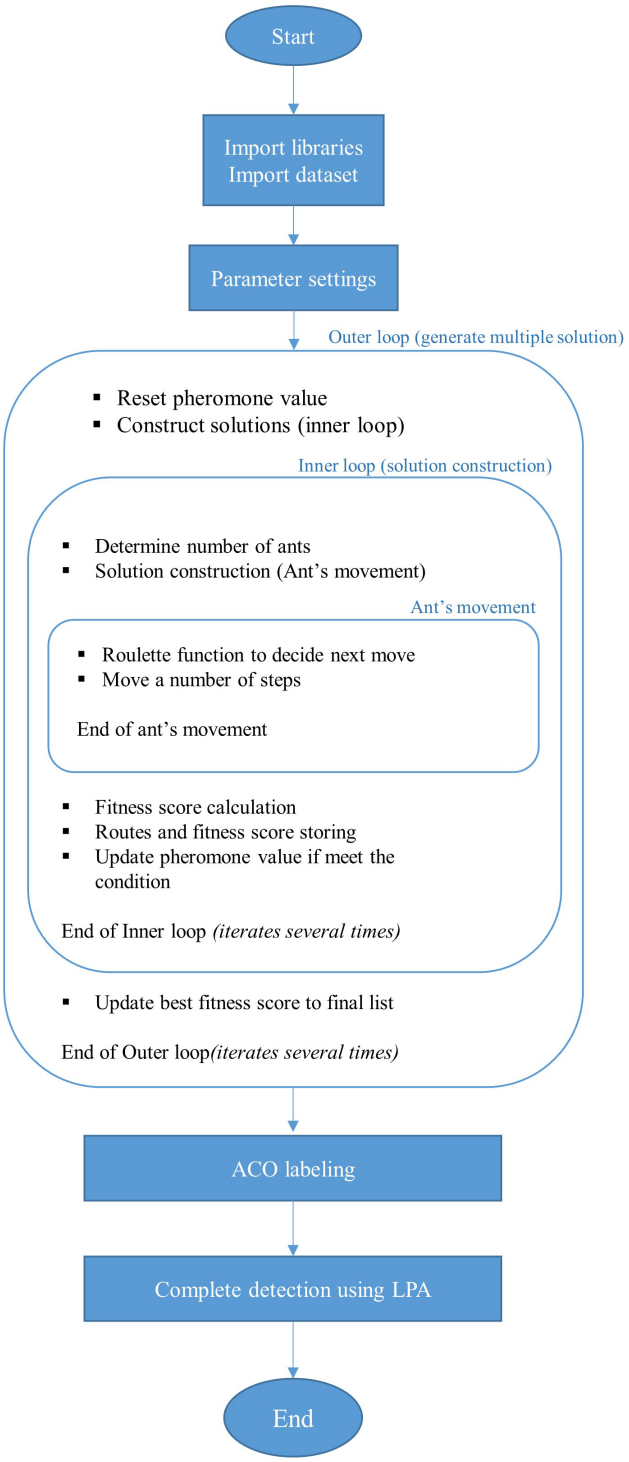

Figure 4.1: Flowchart of community detection process of ACO algorithm with LPA.

CHAPTER 5

RESULTS AND DISCUSSION

In this chapter, we utilized the LFR benchmark and real-world benchmark network to evaluate the performance of the proposed method. The NMI will be computed and compared to the ground truth network to determine the algorithm's effectiveness. Meanwhile, a comparative analysis is conducted by comparing our results with the results obtained using the existing community detection method. Subsequently, the algorithm will be applied to real-world networks for the purpose of community detection.

5.1 LFR benchmark

In this subsection, the algorithm is assessed across all types of LFR networks, encompassing unweighted/undirected (UW/UD), unweighted/directed (UW/D), weighted/undirected (W/UD), and weighted/directed (W/D) scenarios. The primary focus of the evaluation process is on the Normalized Mutual Information (NMI) metric, and the results are systematically compared with existing community detection methods, including standalone Label Propagation Algorithm (LPA) and Infomap.

For all the LFR networks, a constant network size of 1000 nodes is maintained, while the mixing parameter for topology is systematically varied. This approach allows us to assess the algorithm's performance across different levels of complexity of network structures. Additionally, in the case of weighted networks, the mixing parameter for topology will be fixed at 0.5, while the mixing parameter for edge weights is dynamically adjusted. This variation in edge weights assesses the algorithm’s adaptability and effectiveness in handling weighted networks. On top of that, the methods were executed five times for each of the algorithms as they might produce variable outcomes due to the randomness produced by LPA. Thus, the average of the results will be considered for a more robust assessment.

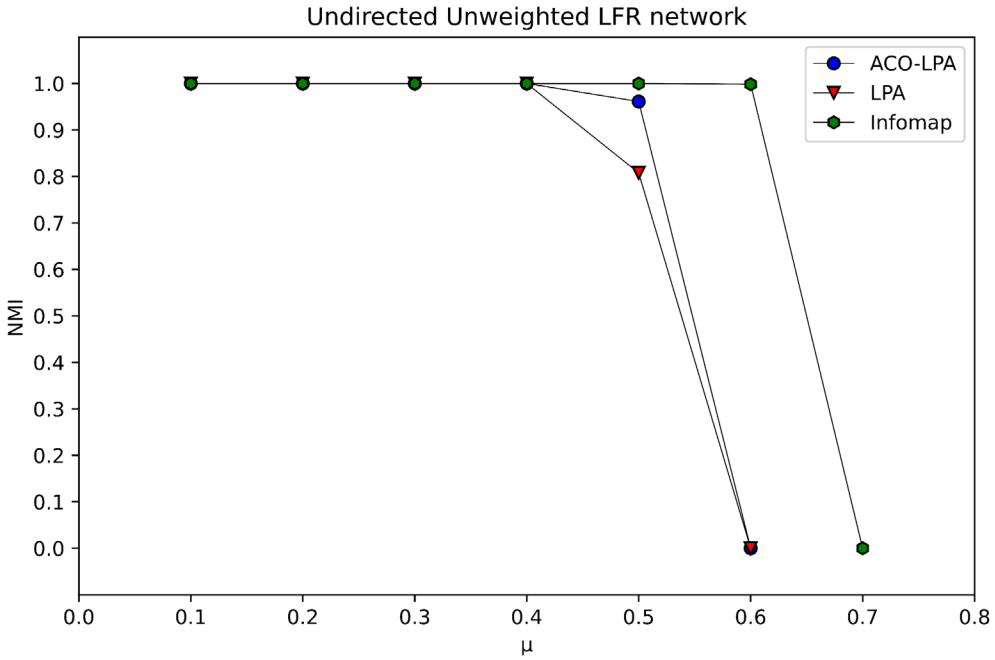

Figure 5.1: NMI scores of community detection methods in Undirected Unweighted LFR network with different levels of mixing parameters for topology

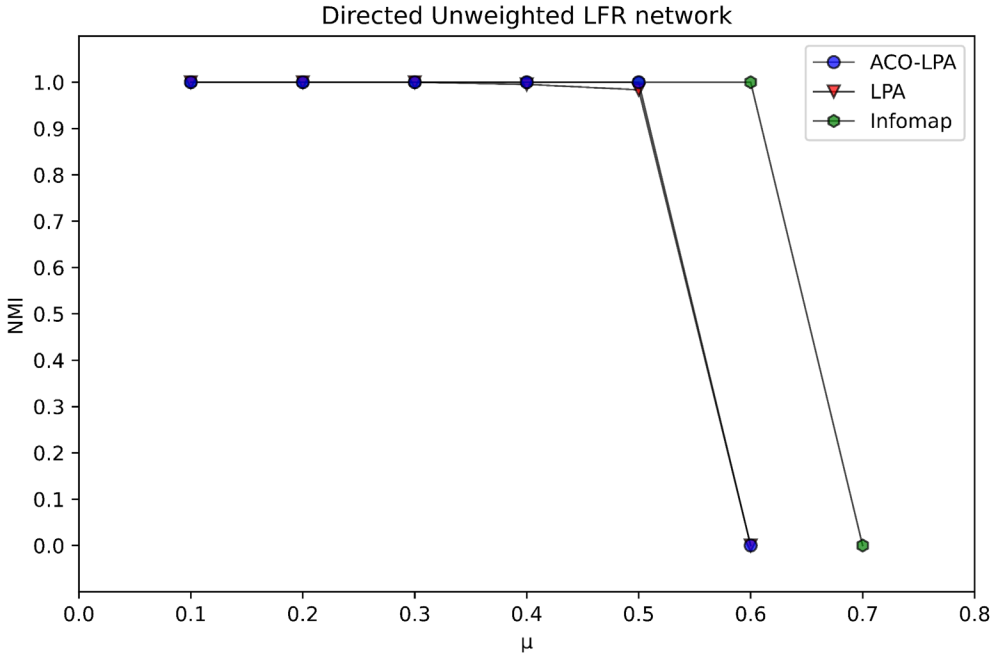

Figure 5.2: NMI scores of community detection methods in Directed Unweighted LFR network with different levels of mixing parameters for topology

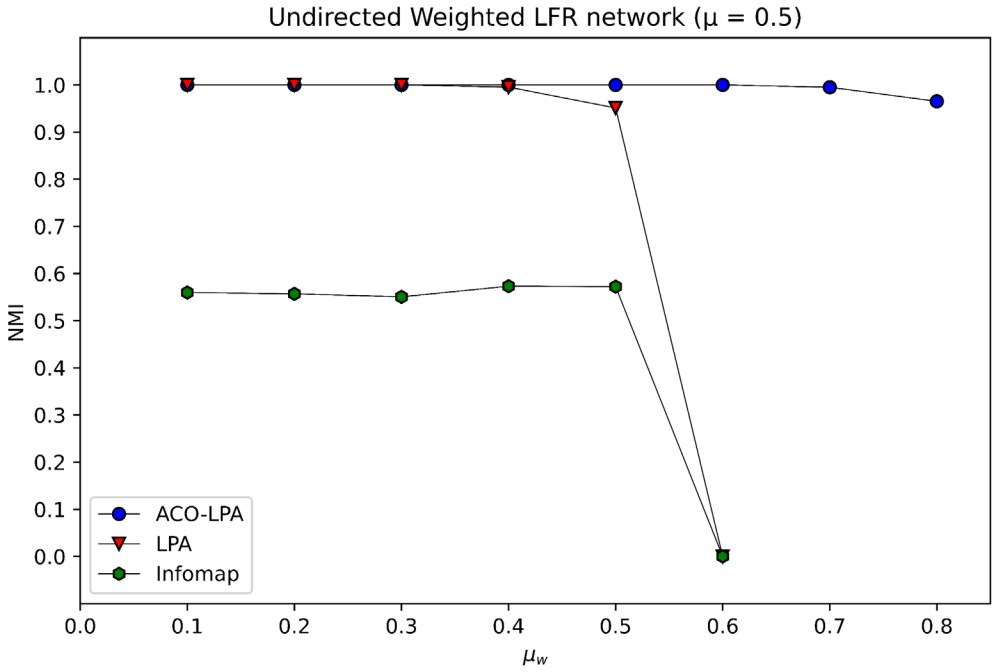

Figure 5.3: NMI scores of community detection methods in Undirected Weighted LFR network with different levels of mixing parameters for edge weights

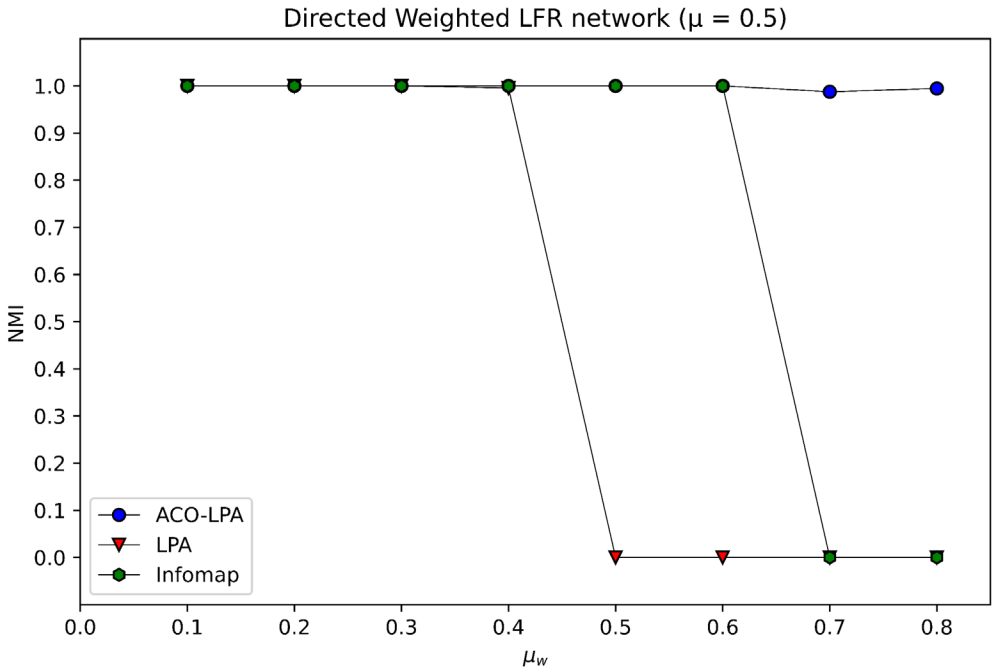

Figure 5.4: NMI scores of community detection methods in Directed Weighted LFR network with different levels of mixing parameters for edge weights

The results of community detection are visually presented in Figure 5.1 to Figure 5.4. The NMI scores assess the similarity between the detected

communities and the predefined communities, with a higher value indicating a greater similarity between the identified communities and their predefined counterparts. Notably, Infomap demonstrates a remarkable ability to detect communities effectively when the mixing parameter for network topology remains below 0.7, surpassing other methods that failed to detect community at 0.6. This underscores Infomap's competence in managing the complexity of network structures. However, in the context of weighted networks, ACO-LPA exhibits a dominant performance across the entire spectrum of mixing parameters for edge weights. In contrast, both LPA and Infomap encounter difficulties in identifying communities when the mixing parameter for edge weights reaches 0.5 or higher in unweighted/undirected (UD/W) networks and 0.6 in directed/weighted (D/W) networks. This observation highlights the proficiency of ACO-LPA in effectively handling weighted networks, where it consistently yields good results across various levels of mixing parameters for edge weights.

5.2 Real-world network with predefined communities

In the following subsection, the evaluation are extended to real-world benchmark networks, applying these methods able to assess their effectiveness in community detection with the presence of community structure.

Table 5.1: NMI scores of community detection methods in real-world benchmark networks.

Network	ACO-LPA	LPA	Infomap
Politic book	**0.291**	0.2884	0.2864
Football	0.9527	0.9102	**0.9721**
Dolphin	0.5611	0.5882	**0.5932**
Karate	**0.9241**	0.5915	0.6995

The provided table presents the NMI scores for real-world benchmark networks obtained using ACO-LPA, LPA, and the Infomap community detection method. The analysis involves a set of famous real-world networks such as the karate network, football network, dolphin network, and political book network. These networks are classic real-world benchmark networks,

utilizing the presence of predefined communities, making them ideal for evaluating the algorithm's performance. As shown in Table 5.1, the Normalized Mutual Information (NMI) values across the methods have relatively small variations, highlighting their overall comparability in the effectiveness of the algorithm across most of the datasets. Additionally, the relatively consistent NMI results across the other existing community detection methods highlight the algorithm's robustness and applicability in different real-world scenarios, offering valuable insights into its capability.

Table 5.2: Modularity scores of community detection methods in real-world benchmark networks.

Network	ACO-LPA	LPA	Infomap
Politic book	0.5178	0.4902	**0.5228**
Football	0.5299	**0.6046**	0.6005
Dolphin	0.5201	0.4867	**0.5277**
Karate	0.3715	0.3749	**0.402**

The provided table presents the modularity scores for real-world benchmark networks obtained using ACO-LPA, LPA, and Infomap community detection methods. The modularity scores determine the effectiveness of the methods in identifying well-structured communities within networks, with a higher value indicating a better structure of detected communities. As shown in Table 5.2, the Infomap method consistently shows a superior performance when compared to the other methods. This highlights its capabilities of detecting well-structured communities within the networks. Notably, while the ACO-LPA method may not consistently yield the highest modularity scores, it still maintains a competitive edge, displaying its potential as a promising method for community detection.

5.3 Real-world networks

Finally, the last subsection will implement the algorithms into the real-world networks to evaluate their performance by comparing them to existing community detection methods.

Table 5.3: Modularity scores of community detection methods in real-world networks

Network	Type	ACO-LPA	LPA	Infomap
Taro	D/UW	0.3823	0.3633	**0.4533**
Residence	D/W	0.4278	0.3851	**0.4595**
Les miserable	UD/W	0.4944	**0.5641**	0.5571
Physician	D/UW	0.5527	0.575	**0.6466**
Jazz	UD/UW	**0.3819**	0.3591	0.28
Train	UD/W	0.3904	0.3258	**0.4105**

The provided table presents the modularity scores for real-world networks obtained using ACO-LPA, LPA, and Infomap community detection methods. Due to the absence of predefined communities in these real-world networks, the NMI score of the algorithm cannot be evaluated. Consequently, modularity serves as the primary metric for assessing the quality of community detection. Overall, the results show a notable degree of similarity, with Infomap demonstrating a slightly superior performance compared to the other methods. However, it is essential to underscore that the remaining methods consistently yield promising results when compared to other methods within the same networks. Although ACO-LPA may not shine as brightly in identifying well-structured communities, it is still capable of producing commendable results, rendering it a viable choice for community detection methods.

5.4 Summary

In short, we conducted a comprehensive evaluation of the proposed algorithms across various network scenarios, including the LFR benchmark network, real-world benchmark networks, and real-world networks. Our methods showcased their effectiveness in community detection within weighted networks, often outperforming existing community detection methods. Moreover, our algorithms consistently delivered promising results in identifying well-structured communities, although when the existing methods only slightly outperformed them. As a final point, we extended the applicability of our

proposed methods to real-world networks, where they serve as valuable community detection methods.

CHAPTER 6

RECOMMENDATIONS AND CONCLUSIONS

In this chapter, the limitations of the proposed method and suggestions for future work will be discussed. The project will then be summarized into paragraphs to conclude the necessary key points.

6.1 Limitation and Challenges

In this project, the ACO algorithm is combined with LPA for community detection. In this subsection, the limitations and challenges of the proposed method will be discussed in detail.

Ant Colony Optimization (ACO) is a widely recognized method for community detection. Nevertheless, it required parameter fine-tuning, including pheromone values, and population size, and the balance between exploration and exploitation. Incorrect parameter settings can lead to issues such as premature convergence, excessive computational demands, or convergence to suboptimal solutions. The pursuit of optimal parameter values can be a challenging task as the given problem might be different.

Another challenge in the proposed method lies in its scalability concerning network size. The computational cost for constructing a solution are greatly depend on the network's scales. As the network size increases, it will result in exponentially growing computational costs, leading to a significant scalability concern.

Regarding convergence iterations, the current proposed method is not concerned about this aspects. However, it should be noted that while the algorithm remains capable of efficiently detecting communities within a limited number of iterations, the number might be insufficient as the size of networks grows. By simply increasing the number of iterations to enhance results, it may result in increasing the computational cost exponentially which is impractical.

Despite these limitations, it's important to emphasize that these limitations do not necessarily make ACO-LPA unsuitable for community

detection. The choice of method should depend on the problem at hand and the goal of the analysis.

6.2 Recommendations for future work

In future research, it is advisable to develop more precise and robust methods to set up the algorithm's parameter settings. The approach can be done by utilizing machine-learning-based optimization to fine-tune the optimal value for each of the parameters in the algorithms. The parameters should be a primary focus when seeking to enhance algorithm performance, as they directly influence on the overall algorithm's effectiveness.

In addition, an alternative strategy for ant movement within the network should also be considered. Experimentation with different movement rules can offer fresh insights into optimizing the exploration and exploitation balance, as the current strategies are solely based on the probabilistic function. The approach has limited the potential for ants to discover different solutions within the other side of networks.

By considering these recommendations, future research can contribute to the advancement and refinement of the ACO-LPA community detection method, thus, leading to more effective and versatile solutions for community detection in complex networks.

6.3 Conclusion

In summary, a community detection method has been developed by incorporating the ACO algorithm with LPA. The parameters of the ACO algorithm are optimized to maximize the performance in detecting communities. The proposed method has been assessed using various benchmark networks, comparing its performance to existing community detection methods. The proposed method shows its capability to handle various types of networks and yields promising results when compared to other methods. Nonetheless, certain limitations and challenges encountered by this method can be overcome to improve the algorithm's effectiveness. Therefore, in the future, we hope that the proposed method provides valuable

insights and serves as a foundational point for the development of more effective community detection methods.

REFERENCES

Chang, H., Feng, Z. and Ren, Z. (2013) Community detection using Ant Colony Optimization, Congress on Evolutionary Computation. Available at: https://doi.org/10.1109/cec.2013.6557944.

Chin, J. H. and Kamali, M.Z.M. (2019) An ant colony approach in the detection of communities in complex networks, Proceedings of The International Conference On Mathematical Sciences And Technology 2018 (MATHTECH2018): Innovative Technologies for Mathematics & Mathematics for Technological Innovation. Available at: https://doi.org/10.1063/1.5136488.

Dorigo, M. and Di Caro, G. (1999) Ant colony optimization: a new meta-heuristic, Congress on Evolutionary Computation. Available at: https://doi.org/10.1109/cec.1999.782657.

Girvan, M. and Newman, M. (2002) "Community structure in social and biological networks," Proceedings of the National Academy of Sciences of the United States of America, 99(12), pp. 7821–7826. Available at: https://doi.org/10.1073/pnas.122653799.

Hosseini, R. and Rezvanian, A. (2020) "AntLP: ant‐based label propagation algorithm for community detection in social networks," CAAI Transactions on Intelligence Technology, 5(1), pp. 34‐41. Available at: https://doi.org/10.1049/trit.2019.0040.

Newman, M. and Girvan, M. (2004) "Finding and evaluating community structure in networks," Physical Review E, 69(2). Available at: https://doi.org/10.1103/physreve.69.026113.

Raghavan, U.N., Albert, R. and Kumara, S.R.T. (2007) "Near linear time algorithm to detect community structures in large-scale networks,"

Physical Review E, 76(3). Available at: https://doi.org/10.1103/physreve.76.036106.

Wang, C., Zhang F., Deng Y., Gao C., Li X., Wang Z. (2020) “An adaptive population control framework for ACO-based community detection,” Chaos Solitons & Fractals, 138, p. 109886. Available at: https://doi.org/10.1016/j.chaos.2020.109886.