Multi-Objective Community Detection

Enroll. No. (s) – 20103012, 20103016, 20103020

Name of Students – Pratishtha Bhateja , Muskan Mittal , Manika Agarwal

Name of Supervisor – Ms. Kirti Aggarwal

May-2023

Submitted in partial fulfilment of the Degree ofBachelor of Technology

In

Computer Science Engineering

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING & INFORMATION TECHNOLOGY

JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY, NOIDA

(I)

TABLE OF CONTENTS

Chapter No.	Topics	Page No.
Chapter-1	**Introduction**	**1-11**
	1.1 General Introduction	
	1.2 Problem Statement	
	1.3 Significance/Novelty of the problem	
	1.4 Empirical Study	
	1.5 Comparison of Existing approaches to the problem framed	
Chapter-2	**Literature Survey**	**12-25**
	2.1 Summary of papers studied	
	2.2 Integrated Summary of the Literature studied	
Chapter-3	**Requirement Analysis and Solution Approach**	**26**
	3.1 Programming Language Used	
	3.2 Framework	
	3.3 Libraries	
	3.4 Features	
Chapter-4	**Modelling and Implementation Details**	**27-32**
	4.1 Control Flow Diagram	
	4.2 Implementation details and issues	
Chapter-5	**Findings, Conclusion, and Future Work**	**32-35**
	5.1 Findings and Result	
	5.2 Conclusion	
	5.3 Future Work	
References		

(II) DECLARATION

We hereby declare that this submission is our own work and that, to the best of our knowledge and belief, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute ofhigher learning, except where due acknowledgment has been made in the text.

Place: ……

Date: 08-05-2023

Signature:

Name: Pratishtha Bhateja

Enrollment No: 20103012

Place: ……

Date: 08-05-2023

Signature:

Name: Muskan Mittal

Enrollment No: 20103016

Place: ……

Date: 08-05-2023

Signature:

Name: Manika Agarwal

Enrollment No: 20103020

(III) CERTIFICATE

This is to certify that the work titled **"MULTI-OBJECTIVE COMMUNITY DETECTION"** submitted by **" PRATISHTHA BHATEJA "**, **"MUSKAN MITTAL "** and **"MANIKA AGARWAL "** in partial fulfilment for the award of degree of Bachelor of Technology of Jaypee Institute of Information Technology, Noida has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma

Signature of Supervisor : ……

Name of Supervisor : Ms. Kirti Aggarwal

Designation : Assistant Professor - II

Date : 08-05-2023

(IV) ACKNOWLEDGEMENT

We would like to express our special thanks of gratitude to our project supervisor **Ms. Kirti Aggarwal** as well as Jaypee Institute of Information Technology, Noida who gave us the golden opportunity to do this wonderful project on the topic **"Multi-Objective Community Detection"**, which also helped us in doing a lot of research and we came to know about so many new things which we are really thankful for. Secondly, we would also like to thank our friends who helped us a lot in the collection of the text descriptions required for this project within the limited time frame.

Signature of Student : ……

Name of Student : Pratishtha Bhateja

Enrollment Number : 20103012

Date : 08-05-2023

Signature of Student : ……

Name of Student : Muskan Mittal

Enrollment Number : 20103016

Date : 08-05-2023

Signature of Student : ……

Name of Student : Manika Agarwal

Enrollment Number : 20103020

Date : 08-05-2023

(V) SUMMARY

Multi-objective community detection is a technique used to identify communities or clusters in a network, where multiple objectives or criteria are considered simultaneously. These objectives can include factors such as network modularity, edge density, node similarity, and community size balance.

The goal of multi-objective community detection is to find a set of solutions, known as the Pareto-optimal set, that represents the best trade-offs between the different objectives. These solutions are not dominated by any other solutions in the set, meaning that no other solution performs better in all objectives simultaneously. Multi-objective community detection has applications in various fields, such as social network analysis, bioinformatics, and transportation network analysis. It can help identify meaningful communities in a network that have different characteristics and can be used for various purposes.

_____ _____

Signature of Student

Name : Pratishtha Bhateja

Date : 08-05-2022

Signature of Supervisor

Name : Ms. Kirti Aggarwal

Date : 08-05-2023

_____

Signature of Student

Name : Muskan Mittal

Date : 08-05-2023

_____

Signature of Student

Name : Manika Agarwal

Date : 08-05-2023

(VI)

LIST OF FIGURES

Fig number	LABEL	Pg no.
1	PSO algorithm	7
2	The topology of the graph.	8
3	An arbitrary solution and its corresponding community structure.	8
4	Pseudocode for MOPSO	18
5	Control flow diagram of MOPSO	27
6	Karate club – dataset graph representation	29
7	Dolphins – dataset graph representation	30
8	Communities in Karate-Club dataset	34
9	NMI value for Karate-Club dataset	35
10	Communities in Dolphins dataset	35
11	NMI value for Dolphins dataset	35
12	Resulted NMI	36

1. INTRODUCTION

1.1 General Instruction

Community detection is a powerful tool for understanding the structure and function of complex networks. To perform community detection, you need to define the network you want to analyze, choose an appropriate algorithm, and analyze the results. There are many algorithms available, each with its strengths and weaknesses, so it's important to choose one that's appropriate for your network and goals. Once you have identified communities, you can analyze their properties and understand their roles within the network. Overall, community detection is a valuable technique for understanding the organization of networks and the behavior of their constituent nodes.

1.2 Problem Statement

In this project we will try to evaluate all the communities in a graph using multi-objective Particle – Swarm Optimization , the objectives being modularity NMI and conductance

1.3 Significance/Novelty of the problem

The significance of community detection in multi-objective lies in its ability to provide a more comprehensive understanding of complex networks by uncovering multiple structures within a single network. It also allows for the identification of communities that satisfy multiple objectives, which can provide valuable insights into the underlying mechanisms that govern the network's behavior. Moreover, multi-objective community detection can be applied to various fields, such as social networks, biology, and computer science, where understanding the structure and function of networks is crucial. Overall, multi-objective community detection represents a significant advancement in the field of network analysis, enabling a more holistic approach to understanding complex systems.

1.4 Empirical Study (Field Survey/Existing Tool Survey/Experimental Study)

1.4.1 Tool Survey

1.4.1.1 IPython Notebook - Jupyter

Interactive Python (or IPython) is a tool or basically a command prompt/shell for interactive computing in Python programming language, but now it has extended support for different other languages like R. Although it is based on browser, it offers a notebook-like interface with support for code, math expressions, plots that are inline, text and other media. The notebooks offer rich graphs/diagrams, shell syntax and tab completion. The shells are interactive, cell-based where each cell can be executed independently. It also offers support for data visualization

1.4.1.2 Google Colab

Colaboratory is a free Jupyter notebook environment provided by Google that runs entirely in the cloud. It requires no setup and is a platform for building machine learning models with GPU support. Google Colab enables one to write and execute code and access computing resources for free from the browser. It comes with libraries which are used for accessing various services provided by Google conveniently. It saves files to Google Drive and allows easy sharing of Jupyter notebooks Google Colaboratory is hosted by Google designed for data experimentation and analysis.

But on the downside, there are some limitations, like for example the entire environment (i.e. libraries, packages, and data files) do not persist across sessions Programs cannot be executed for longer than a certain amount of time.

1.4.1.3 Gephi

Gephi is an open-source network analysis and visualization software that allows users to explore, analyze, and visualize complex networks. It provides an intuitive graphical interface for creating and manipulating networks and provides a range of tools for network analysis, including community detection, centrality analysis, and clustering analysis.

One of the main features of Gephi is its ability to import data from various sources, including CSV files, Excel spreadsheets, and other network analysis software. It also provides a range of layout

algorithms that enable users to visualize networks in different ways, such as circular, force-directed, and hierarchical layouts.

Gephi also allows users to customize the appearance of their networks by changing the size, color, and shape of nodes and edges. Users can also add labels, tooltips, and other annotations to their networks to provide additional information and context.

In addition, Gephi supports the use of plugins, which can extend the software's functionality and provide additional tools and features for network analysis and visualization. The software has an active community of users and developers, who contribute to its ongoing development and provide support and resources for users.

1.4.2 Libraries Survey

1.4.2.1 Networkx

Networkx is a Python library for studying complex networks, such as social networks, transportation networks, biological networks, and many others. It provides tools for creating, manipulating, and analyzing graph structures, and can be used for a wide range of applications, including network visualization, graph algorithms, and statistical analysis.

The library provides a simple and intuitive API for working with graphs, which are represented as a collection of nodes and edges. Networkx supports many different types of graphs, including directed, undirected, weighted, and bipartite graphs, and provides a variety of algorithms for working with these graph types.

Some of the key features of Networkx include:

- **Graph creation and manipulation:** Networkx provides a variety of functions for creating and manipulating graphs, including adding and removing nodes and edges, combining graphs, and finding subgraphs.
- **Graph algorithms**: Networkx provides a wide range of algorithms for analyzing graphs, including centrality measures, shortest path algorithms, and community detection algorithms.
- **Visualization:** Networkx includes functions for visualizing graphs, both in 2D and 3D, using a variety of different layout algorithms.

- **I/O:** Networkx supports reading and writing graphs from and to a variety of different file formats, including GraphML, GEXF, and JSON**.**

1.4.2.2 Numpy

NumPy is a Python library used for working with arrays. Array programming provides a powerful, compact and expressive syntax for accessing, manipulating and operating on data in vectors, matrices and higher-dimensional arrays. NumPy is the primary array programming library for the Python language. It also has functions for working in domain of linear algebra, fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely. NumPy stands for Numerical Python. The NumPy array is a data structure that efficiently stores and accesses multidimensional arrays17 (also known as tensors), and enables a wide variety of scientific computation. It consists of a pointer to memory, along with metadata used to interpret the data stored there, notably „data type“, „shape“ and „strides“.

1.4.2.3 Matplotlib

Matplotlib is a cross-platform, data visualization and graphical plotting library for Python and its numerical extension NumPy. As such, it offers a viable open-source alternative to MATLAB. Developers can also use matplotlib's APIs (Application Programming Interfaces) to embed plots in GUI applications. Matplotlib is a python library used to create 2D graphs and plots by using python scripts. It has a module named pyplot which makes things easy for plotting by providing feature to control line styles, font properties, formatting axes etc. It supports a very wide variety of graphs and plots namely - histogram, bar charts, power spectra, error charts etc.

1.4.2.4 Pandas

Pandas is defined as an open-source library that provides high-performance data manipulation in Python. It is used for data analysis in Python. Data analysis requires lots of processing, such as restructuring, cleaning, or merging, etc. There are different tools are available for fast data processing, such as Numpy, Scipy, Cython, and Panda. But we prefer Pandas because working with Pandas is fast, simple, and more expressive than other tools. Before Pandas, Python was capable for data preparation, but it only provided limited support for data analysis. So, Pandas came into the picture and enhanced the capabilities of data analysis. It can perform five significant steps required

for processing and analysis of data irrespective of the origin of the data, i.e., load, manipulate, prepare, model, and analyse.

1.4.2.5 Math

The math library is a standard Python library that provides a collection of mathematical functions and constants. These functions cover a wide range of mathematical operations, including basic arithmetic, trigonometry, logarithms, and statistical functions.

Some of the key features of the math library include:

- **Basic arithmetic:** The math library provides functions for performing basic arithmetic operations, such as adding, subtracting, multiplying, and dividing numbers.
- **Trigonometry:** The math library provides functions for computing trigonometric functions, such as sine, cosine, and tangent.
- **Logarithms:** The math library provides functions for computing logarithms, both natural logarithms and logarithms with a specified base.
- **Constants:** The math library provides a number of important mathematical constants, such as pi (π) and the Euler constant (e).
- **Statistical functions:** The math library provides a range of statistical functions, such as mean, variance, and standard deviation.

In addition to these functions, the math library also includes a number of utility functions for working with floating-point numbers, such as functions for computing the absolute value, rounding, and comparing values.

1.4.2.6 networkx.algorithms.community

The networkx.algorithms.community library is a sub-library of the Networkx Python library, which provides a collection of community detection algorithms for analyzing complex networks. These algorithms aim to identify groups of nodes in a network that are more densely connected to each other than to the rest of the network, and are often used to gain insights into the underlying structure and function of the network.

Some of the key community detection algorithms provided by the networkx.algorithms.community library include:

- **Girvan-Newman algorithm:** This algorithm works by iteratively removing edges from the graph to find communities based on betweenness centrality.
- **Louvain algorithm:** This algorithm is a greedy optimization method that maximizes modularity, a measure of the quality of a partition of a network into communities.
- **Label propagation algorithm:** This algorithm assigns nodes to communities based on the labels of their neighbors, and iteratively updates the labels until convergence.
- **Spectral clustering algorithm:** This algorithm uses the eigenvectors of the graph Laplacian to partition the nodes into communities.
- **Infomap algorithm:** This algorithm uses information theory to identify communities in the network.

These algorithms provide a range of different approaches to community detection, and are suited to different types of networks and applications. In addition, the networkx.algorithms.community library also provides a range of functions for evaluating the quality of community partitions, such as modularity and conductance.

Overall, the networkx.algorithms.community library is a powerful tool for analyzing the community structure of complex networks, and can be used in a wide range of applications, such as social network analysis, bioinformatics, and recommender systems.

1.4.2.7 Sklearn

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modelling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib. Scikit-learn is mainly coded in Python and heavily utilizes the NumPy library for highly efficient array and linear algebra computations. Some fundamental algorithms are also 5 built in Cython to enhance the efficiency of this library. Support vector machines, logistic regression, and linear SVMs are performed using wrappers coded in Cython for LIBSVM and LIBLINEAR, respectively. Expanding these routines with Python might not be viable in such circumstances.

1.5 Comparison of existing approaches to the problem framed

Most of the existing approaches for community detection use a single objective to find the communities. This might lead to inaccurate results as community structure often depends on more than one objective. For community detection, evolutionary techniques like genetic evolution or differential evolution are widely used.

1.5.1 PSO

Particle Swarm Optimization (PSO) is a population-based stochastic optimization algorithm that is inspired by the behavior of social organisms, such as bird flocks and fish schools. In PSO, a population of candidate solutions, called particles, moves through the search space to find the optimal solution to an optimization problem. Each particle is characterized by its position and velocity, which are updated based on its own best-known position and the best-known position of the entire swarm. The PSO algorithm can be customized by adjusting several parameters, such as the number of particles and the acceleration coefficients, and it is particularly effective for problems with multiple local optima or non-linear fitness landscapes. Overall, the PSO algorithm is a powerful optimization algorithm that can be used to solve a wide range of optimization problems.

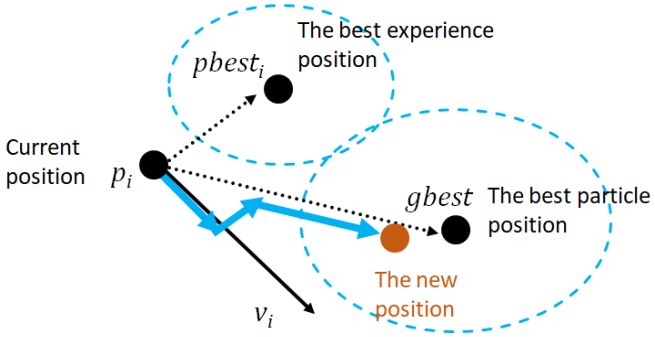

Fig-1 – PSO algorithm

Usually velocity is used to change the position of particles, but we have used two point crossover to bring about changes in the particle. Then the particle is mutated.

1.5.2 LAR – (Locus Based Adjacency Representation)

In a locus-based adjacency representation, each locus is associated with a list of adjacent nodes. This representation is particularly useful for graphs where the number of nodes is large and the adjacency relationships are sparse. By storing only, the adjacent nodes for each locus, it can reduce the storage space required compared to other representations that explicitly store the entire adjacency matrix or adjacency list. There are several advantages in using this type of representation. For instance, by using this scheme, our algorithm does not need to know the number of communities in advance. This is since the number of communities is determined during the decoding process. In addition, the decoding procedure of a solution can be done in a linear time. Also, using LAR scheme, the standard crossover operators can be easily performed on the solutions.

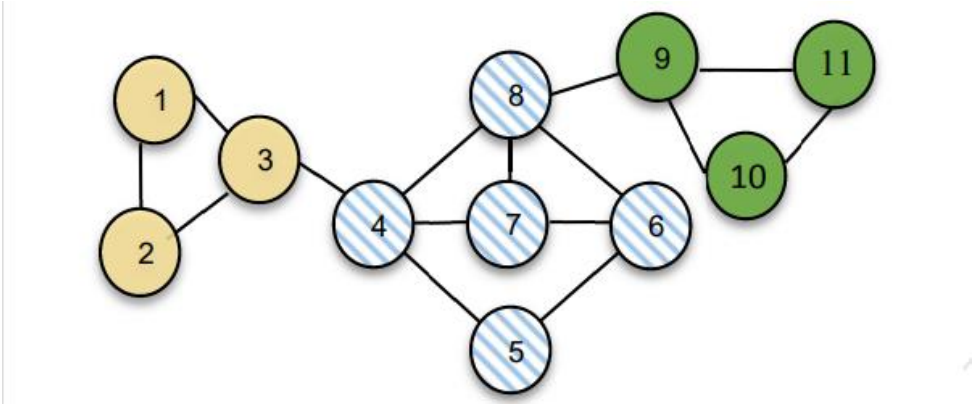

Fig-2 The topology of the graph.

Position:	1	2	3	4	5	6	7	8	9	10	11
Genotype:	2	1	1	7	4	7	8	4	11	11	9

Fig-3 An arbitrary solution and its corresponding community structure.

HOW IT WORKS ?

- **PSO**

The Particle Swarm Optimization (PSO) algorithm works by iteratively improving a population of candidate solutions, called particles, to solve an optimization problem. The basic steps of the PSO algorithm are as follows:

1. **Initialization:** The algorithm starts by randomly initializing a population of particles in the search space by using Locus based adjacency representation.
2. **Evaluation:** Each particle is evaluated based on its fitness, which represents the quality of its position in the search space with respect to the optimization problem. The modularity, conductance and NMI of the particle is calculated.
3. **Initialize personal best:** Initialize the personal best of the particle with the particle itself.
4. **Find pareto optimal set:** Using the modularity and conductance, find pareto set. These solutions are not dominated by any other solutions in the set, meaning that no other solution performs better in all objectives simultaneously.
5. **Set global best:** From the pareto set, find the particle with the highest NMI and set it as the global best.
6. **Search strategy:** Search strategy of the PSO is based on moving each particle toward its local best and moving it toward the global best of the swarm. In order to move each particle toward the best positions, genetic operators such as crossover and mutation are used.
    - Moving toward personal best: Each particle, at first, performs a two-point crossover with its personal best. As a result, two new particles are obtained as the output of the crossover operation. Then, a dominance comparison is performed between the obtained results. If a solution dominates another one, it is selected as temporary position of current particle. If none of them dominates each other, an output with the highest NMI value is selected as temporary particle.
    - Moving toward global best: After moving a particle toward its personal best, it is also moved toward the global best. To this end, a two-point crossover between its temporary position and the global best is performed. In this case, also two new

solutions are obtained. These two outputs are compared together to specify the new position of the global best.

- Mutation: To mutate the particles around the whole search space, one-point neighbour-based mutation is performed on all particles such that, for each particle, a gene is picked randomly and then it is replaced by a possible value from its neighbouring nodes. This limitation is to guarantee producing possible solutions in the solution space. The result of mutation operator for each particle is compared to its personal best. In this way, if a mutated particle dominates its personal best, the personal best is replaced by the mutated particle.

7. Obtain pareto set, and then assign the particle with highest NMI from this set as gbest
8. **Output:** The best-known position of the entire swarm is returned as the solution to the optimization problem.

The PSO algorithm can be customized by adjusting several parameters, such as the number of particles and the number of iterations. The choice of these parameters can affect the performance of the algorithm, and they need to be carefully tuned for each specific optimization problem.

1.5.3 MODULARITY

Modularity is a measure of the degree to which a network can be divided into non-overlapping subgroups, or modules, based on the connections between nodes. In other words, *it measures the extent to which the nodes in a network are more densely connected within groups than between groups.* A network with high modularity has many tightly-knit modules, while a network with low modularity has fewer or no well-defined modules.

Modularity with PSO :

PSO can be used to optimize the modularity of a network by adjusting the assignment of nodes to communities. One approach is to define a fitness function that evaluates the modularity of the network based on the current assignment of nodes to communities. The fitness function can be formulated as the modularity value of the network, which is maximized by adjusting the community assignments of the nodes. The PSO algorithm can then be used to search for the optimal community assignments that maximize the modularity value. In this approach, the particles in the PSO algorithm represent candidate solutions that correspond to different community assignments of

the nodes in the network. Each particle is characterized by its position, which corresponds to a specific community assignment, and its velocity, which determines the direction and magnitude of the change in the community assignment. The fitness of each particle is evaluated based on the modularity of the network corresponding to the community assignment represented by its position.

PSO with modularity optimization can be an effective approach for community detection in complex networks, as it can efficiently explore the space of possible community assignments and converge to high-quality solutions. However, the performance of PSO for modularity optimization depends on the choice of the fitness function, the parameters of the PSO algorithm, and the structure of the network.

1.5.4 NMI

Normalized Mutual Information (NMI) is a measure of the similarity between two partitions of a set of objects. In the context of clustering, NMI is used to compare the clustering results obtained from different algorithms or with different parameter settings. NMI is a normalized version of the Mutual Information (MI) measure, which quantifies the amount of information shared between two partitions. MI measures the reduction in uncertainty about one partition given knowledge of the other partition. NMI is defined as the ratio of the MI between two partitions to the geometric mean of their entropies. The result is a value between 0 and 1, where 0 indicates no agreement between the partitions, and 1 indicates perfect agreement.

1.5.6 Conductance

The conductance of a community is defined as the measures the quality of a community by comparing the number of edges between a community and the rest of the network to the total number of edges in the community. Lower conductance values indicate better community structure.

More specifically, if we let G = (V,E) be a graph, where V is the set of nodes (vertices) and E is the set of edges, and let S be a subset of V, then the conductance of S is defined as follows:

Conductance(S) = (sum of the weights of the edges connecting nodes in S to nodes outside of S) / (2 * sum of the weights of all edges incident on nodes in S)

2. LITERATURE SURVEY

2.1 **Paper 1:** Community detection in social networks

A *social network* is depicted by a social network graph G consisting of n number of nodes denoting n individuals or the participants in the network. The connection between node i and node j is represented by the edge eij of the graph. A directed or an undirected graph may illustrate these connections between the participants of the network. The graph can be represented by an adjacency matrix A in which Aij = 1 in case there is an edge between i and j, else Aij = 0.. A *community* can be defined as a group of entities closer to each other in comparison to other entities of the dataset. A community is formed by individuals such that those within a group interact with each other more frequently than with those outside the group. *Clustering* is the process of grouping a set of similar items together in structures known as clusters. Clustering the social network graph may give a lot of information about the underlying hidden attributes, relationships, and properties of the participants as well as the interactions among them.

Community Detection Algorithms :

- *Graph Partitioning-based Community Detection :* Graph partitioning-based methods have been used in the literature to divide the graph into components such that there are few connections between the components. The algorithm, however, is quite fast, with a worst-case running time of $O(n^2)$.
- *Clustering-based Community Detection:* The main concern of community detection is to detect clusters, groups, or cohesive subgroups. The basis of a large number of community-detection algorithms is clustering. The algorithm focused on edges that are most 'between' the communities, and communities are constructed progressively by removing these edges from the original graph. The worst-case time complexity of the edge-betweenness algorithm is $O(m^2n)$ and is $O(n^3)$ for sparse graphs, where m denotes the number of edges, and n is the number of vertices.

Datasets :

1. Real-time datasets like the *Karate club* network by Zachary represent the relationships between 34 members of a karate club over a period of 2 years.
2. *Dolphin* Social Network depicts the social interactions and behavior of bottlenose dolphins for a period of 7 years as studied by Lusseau
3. The *American College Football* Network19 dataset consists of the football teams in America.

2.2 **Paper 2:** An Improved PSO algorithm for community detection

The core idea of the PSO is to simulate the process of optimal in solution space as birds hunting. PSO can converge fast and has a few parameters. The main purpose of this paper is to improve community detection efficiency by applying PSO. In this paper, different kinds of particle swarm algorithms are implemented and compared. To deal with the inadequacy of basic PSO for community detection, this paper presents a binary PSO algorithm based on velocity probability and roulette strategy.

Modularity : Modularity is a traditional metric to evaluate the result of community detection. Its basic idea is to compare the dense within the community and the dense of random network as Formula (1). The value of modularity is defined from 0 to 1. The larger modularity is, the better communities are detected. PSO community detection is actually to optimize the modularity value. Therefore the modularity is correspondingly the fitness of particle.

$$Q = \frac{1}{W} \sum_{C \in P} \sum_{i,j \in C} [A_{ij} - \frac{k_i k_j}{W}] \quad (1)$$

$$v_i^{k+1} = w v_i^k + c_1 r_1 (pbest_i^k - x_i^k) + c_2 r_2 (gbest_i^k - x_i^k) \quad (2)$$

$$x_i^{k+1} = x_i^k + v_i^{k+1} \quad (3)$$

$$v_{ij} = \begin{cases} v_{ij} = v_{\max}, v_{ij} \geq v_{\max} \\ v_{ij} = -v_{\max}, v_{ij} \leq -v_{\max} \end{cases} \quad (4)$$

sPSO and bPSO: The idea of PSO is to generate some feasible solutions, particles, randomly and adjust the velocity and position by the pbest and gbest calculated according to the change of fitness of particles as they move in the solution space. The iteration of the process updates the pbest and gbest and further adjusts the velocity and the position of particles, thereby searches a better solution. The mathematical model of PSO is as follows :

Experiment design and result analysis : This paper tests two network benchmarks, the *karate club network* and the *dolphin social network*. The karate club network consists of 34 members as the nodes, and 78 relationships between members constitute the 78 edges of the network. And the dolphin social network consists of 62 dolphins as the nodes and 159 connections as the edges in the network. The two networks are shown in Figure 5 and Figure 6, respectively:

Analysis of VP-bPSO :

- The process of VP-bPSO is as follows:
- Initialize the network structure, the number of communities and the parameters.
- Initialize the particle swarm.
- Calculate the modularity of each particle.

- Update the pbest of each particle.
- Update the gbest of particle swarm.
- Train the velocity of particles by Formula (2).
- Calculate the velocity probability of particles by Formulae 9, 10 and 11.
- Roulette decides the scheme of community detection according to the velocity probability.
- Judge by the iteration or convergence, if it's completed, go to Step 10, if not, go to Step 3.
- Output the result.

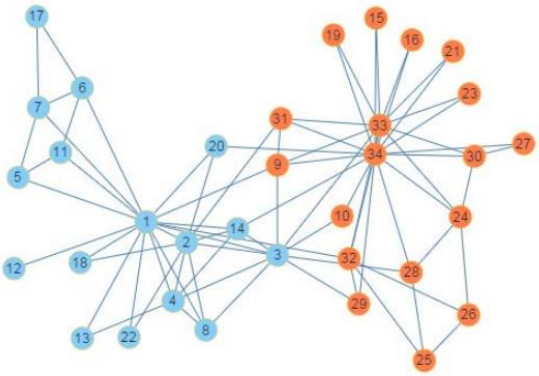

Figure 5. Karate club network

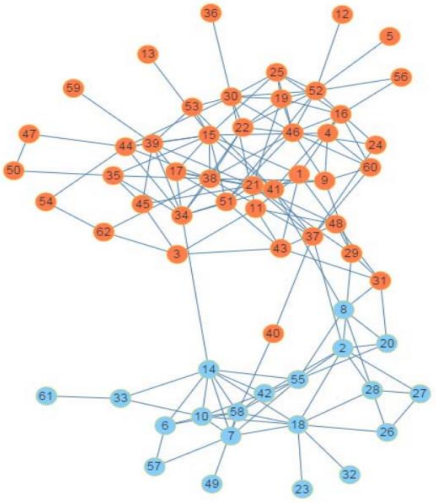

Figure 6. Dolphin social network

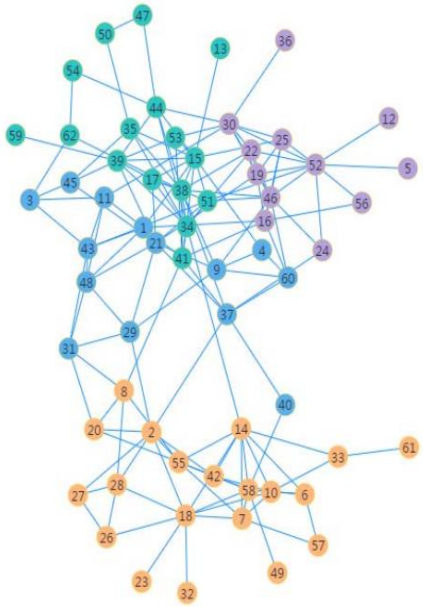

Figure 10. Dolphin social network divided into 4 communities by VP-bPSO

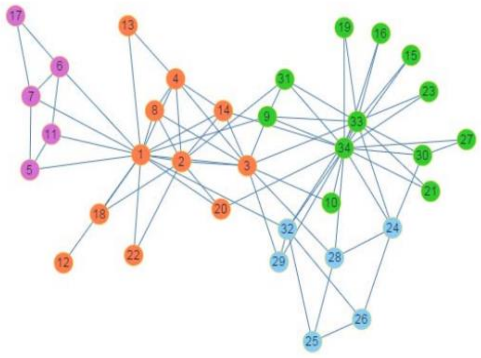

Figure 8. Karate club network divided into 4 communities by VP-bPSO

2.3 **Paper 3:** A Brief Review on Particle Swarm Optimization: Limitations & Future Directions

In PSO, the behavior of a flock of birds is used as a metaphor to optimize a problem. In this metaphor, a swarm of particles represents a flock of birds. Each particle corresponds to a bird, which moves through the search space to find the best position to obtain food. Initially, each bird moves randomly throughout the search space. However, it adjusts its direction based on two things: the position of the best food it has ever found so far (personal best) and the position of the best food found by the flock (global best). The personal best represents the best position the bird has ever found during its exploration, and the global best is the position found by the best bird in the swarm. As the birds continue to move through the search space, they gradually converge towards the global best position, just like how a flock of birds fly towards their destination. However, they still maintain a certain level of exploration, allowing the swarm to discover new areas of the search space. In each iteration, the position and velocity of each bird are updated based on the current position, the personal best, and the global best positions. The algorithm continues to iterate until it converges to a solution or reaches a maximum number of iterations.

$$V_i(k+1) = V_i(k) + c1 * \text{rand}( ) * (P_i(k) - X_i(k)) + c2 * \text{rand}( ) * (g(k) - X_i(k)) \quad (1)$$

$$X_i(k+1) = X_i(k) + V_i(k+1)$$

where

$V_i(k)$ is velocity of particle i at iteration k.

$X_i(k)$ is the position of particle i at iteration k.

$V_i(k+1)$ is velocity of particle i at iteration k+1.

$X_i(k+1)$ is the position of particle i at iteration k+1.

rand( ) is random number between (0,1)

c1 cognitive acceleration coefficient

c2 social acceleration coefficient.

Limitations: Original PSO approach (1995) is to optimize the solution using global best there is chance to trapped in local area. No suggestion is provided for such situation. As the algorithm considers the best value found by neighbors it is more efficient for small number of particles. As the number of particles increases, gbest version is more beneficial. A Modified Particle Swarm Optimizer (1998) works better but only small benchmark function it uses to test. There is difficulty to select probable value of inertia weight. The swarm and the queen: Towards deterministic and adaptive particle swarm optimization by Clerc M.(1999) did not clear whether optimal value is dependant on φ. This creates difficulty to select value of φ. All the three methods mentioned above

did not give constant result. Sometimes Rehope method works better sometimes not. It occurs for every method used.

Future Scope : Future work will try to understand where the algorithm suffers in order to understand any limitations and apply it to the proper contexts. One such difficulty seen already was with simple uni-modal functions, where regrouping is unnecessary since particles quickly and easily approximate the global minimizer to a high degree of accuracy, and where there is no better minimizer to be found. While the regrouping mechanism has been tested in conjunction with standard gbest PSO in order to demonstrate the usefulness of the mechanism itself, there does not seem to be anything to prevent the same regrouping mechanism from being applied with another search algorithm at its core.

2.4 Paper 4: A multi-objective particle swarm optimization algorithm for community detection in complex networks.

Most algorithms developed for community detection take advantage of single objective optimization methods which may be inefficient for complex networks. In this paper a multi-objective community detection based on a modified version of PSO is proposed.

A community is defined as a subset of nodes within a graph such that connections between these nodes are denser than connections with the rest of the network. Community detection can be applied in many practical applications like cancer detection, product recommendation, link prediction etc.

To solve the community detection problem more efficiently, in this paper, we propose a novel approach based on multi-objective particle swarm optimization (MOPSO) called the MOPSO-Net. The main advantages of using PSO method compared to the others are its fast convergence speed, the simple implementation, and existing of large amounts of PSO variants algorithms. In MOPSO-Net, we adopted the Pareto dominance concept to PSO to solve multi-objective tasks. The proposed method minimizes two objective functions Kernel K-means and the Ratio Cut.

The framework of the proposed method can be explained in two main steps including; initialization (solution representation and fitness computation), and moving strategy (search strategy). In the initialization step, at first, a specific data structure is used to represent the solutions, which illustrates a community structure of a network, in a proper format. Thereupon, using this representation, the solutions are randomly initialized. Then, for each particle, two objective functions including Kernel K-Means (KKM) and Ratio Cut (RC) are calculated. Thereafter, by

using the nondominated sorting mechanism, a set of non-dominated solutions is obtained. To obtain local and global bests, the Normalized Mutual Information (NMI) value is computed for each particle at the nondominated set and then the particle with highest NMI is considered as the global best of the whole swarm. Inspired from PSO search strategy, in the second step, the particles are moved around the search space through several iterations, to optimize the above-mentioned objective functions.

The MOPSO-Net algorithm uses the locus-based adjacency representation (LAR) as its representation scheme [42]. LAR data structure considers each solution as an array of N genes, each of which belongs to a node in the graph. Each gene is connected randomly to one of its neighbour's and thus it can take a value from the range of 1 to N. Each solution can be decoded as a graph which the value of * for the gene indicates there is an edge between node and node * in this representation. Then, connected components of each solution are identified as the solution communities.

Each particle, at first, performs a two-point crossover with its personal best. As a result, two new particles are obtained as the output of the crossover operation. Then, a dominance comparison is performed between the obtained results. After moving a particle toward its personal best, it is also moved toward the global best. To this end, a two-point crossover between its temporary position (resulted in the previous subsection) and the global best is performed. Finally, to mutate the particles around the whole search space, one-point neighbour-based mutation is performed on all particles such that, for each particle, a gene is picked randomly and then it is replaced by a possible value from its neighbouring nodes.

Datasets Used:

The Zachary's karate club network.

The Bottlenose Dolphins network.

American College Football network.

The Books about US politics network

2.5 **Paper 5:** Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art

Optimization problems that have more than one objective function are rather common in every field or area of knowledge. In such problems, the objectives to be optimized are normally in conflict with respect to each other, which means that there is no single solution for these problems. Instead, we aim to find good "trade-off" solutions that represent the best possible compromises among the objectives. Est possible compromises among the objectives. Particle Swarm Optimization (PSO) is a heuristic search technique that simulates the movements of a flock of birds which aim to find food.

In PSO strategy, the solution set of a problem with multiple objectives does not consist of a single solution (as in global optimization). Instead, in multi-objective optimization, we aim to find a set of different solutions (the so-called Pareto optimal set).

Leaders in Multi-Objective Optimization:

The most straightforward approach is to consider every nondominated solution as a new leader and then, just one leader has to be selected. In this way, a quality measure that indicates how good is a leader is very important. One possible way of defining such quality measure can be related to density measures. Here two important density measures are used:

• Nearest neighbour density estimator

• Kernel density estimator

```
Begin
    Initialize swarm
    Initialize leaders in an external archive
    Quality(leaders)
    g = 0
    While g < gmax
            For each particle
                    Select leader
                    Update Position (Flight)
                    Mutation
                    Evaluation
                    Update pbest
            EndFor
            Update leaders in the external archive
            Quality(leaders)
            g++
    EndWhile
    Report results in the external archive
End
```

Fig-4 Pseudocode for MOPSO

2.6 Paper 6: A PSO Based Community Detection in Social Networks with Node Attributes

In this paper, we leverage the concept of Particle Swarm Optimization (PSO) and cluster the graph based on node attributes. In the proposed clustering process, we assign attribute data to the particles in the swarm. As the execution starts, the swarm is divided into several subswarms in which clustering process is executed asynchronously. Grouping of particles is done based on the similarity threshold which is calculated using a fitness function. Each particle in the sub swarm will exchange clustering information with other particles in swarm using shared memory. During this process, the particle can exchange information with a greater number of diversified particles. This approach improves the quality of clusters through better exploitation and exploration of the swarm.

Calculation of velocity:

$$v_{ij}{}^{t+1} = v_{ij}{}^{t} + c_1 r_{1j}{}^{t} * \left[pBest\ i^t - x_{ij}{}^{t}\right] + c_2 r_{2j}{}^{t} * \left[gBest\ i^t - x_{ij}{}^{t}\right] \quad (1)$$

$$x_i = x_i + v_i \quad (2)$$

Particle representation: Particle (Particle_ID, isMapped, Attribute_Info, Neighbouing_Cluster_Info).

Shared memory : We use shared memory for the swarm to store the clustering information with an objective to share the existing information. Shared memory consists of the following information: Attribute Information, cluster ID, particle ID. During the clustering process, when the distance between the particles is less than a specified value, denoted as ‘dmax’, the particle will first read the isMapped attribute of its neighboring particles.

Similarity Index and Fitness Function:

$$S_{ij} = \frac{1}{k} \sum_{a=1}^{k} S_{ija}$$

Evaluating Criteria :

1. Omega Index:
2. Silhouette Coefficient:
3. Tanimoto Coefficient:

2.7 Paper 7 : MOPSO : A Proposal for Multiple Objective Particle Swarm Optimization

This paper introduces a proposal to extend the heuristic called “particle swarm optimization” (PSO) to deal with multi-objective optimization problems. The analogy of particle swarm optimization with evolutionary algorithms makes evident the notion that using a Pareto ranking scheme could be the straightforward way to extend the approach to handle multi-objective optimization problems. The historical record of best solutions found by a particle could be used to store nondominated solutions generated in the past .

The algorithm of MOPSO is the following:

1. Initialize the population POP:
   (a) FOR i = 0 TO MAX /* MAX = number of particles
   (b) Initialize POP[i]
2. Initialize the speed of each particle:
   (a) FOR i = 0 TO MAX
   (b) VEL[i] = 0
3. Evaluate each of the particles in POP.
4. Store the positions of the particles that represent nondominated vectors in the repository REP.
5. Generate hypercubes of the search space explored so far, and locate the particles using these hypercubes as a coordinate system where each particle's coordinates are defined according to the values of its objective functions.
6. Initialize the memory of each particle (this memory serves as a guide to travel through the search space. This memory is also stored in the repository):
   (a) FOR i = 0 TO MAX
   (b) PBESTS[i] = POP[i]
7. WHILE maximum number of cycles has not been reached DO
   a) Compute the speed of each particle
      VEL[i] = W * VEL[i] + R1 *(PBESTS[i] - POP[i]) + R2 * (REP[h] - POP[i])
   b) Compute the new positions of the particles adding the speed produced from the previous step: POP[i] = POP[i] + VEL[i]
   c) Evaluate each of the particles in POP.

d) Update the contents of REP together with the geographical representation of the particles within the hypercubes.

e) PBESTS[i] = POP[i]

f) Increment the loop counter

7. END WHILE

2.8 Paper 8: A Sorting Based Algorithm for Finding Non-Dominated Set in Multi-Objective Optimization

A sorting based algorithm is proposed in this paper for finding non-dominated set in Multi-Objective optimization. The algorithm is composed by sorting step and dominated solutions deleting step. Some enhancement techniques including primary non-dominated solutions, scoring and summation sequence are used to reduce the computational complexity.

Definition 1 (Non-dominated set) Among a set of solutions P, the non-dominated set of solutions P' are those that are not dominated by any member of the set P.

Definition 2 (Pareto-optimal set) If the set of solutions P is the entire search space, the resulting non-dominated set P' is called the Pareto-optimal set.

Sorting based Algorithm:

1. The population of solutions is sorted according to the descending order to every objective function value. Thereafter, several solution sequences could be presented, and each of them corresponds to one objective. Every solution could be scored according to the position in each solution sequence. More anterior position it takes, more higher score it gains. Then, each solution could get various scores corresponding to various objectives. Take the scores summation of every solution, and sort it to get one new solution summation sequence. The new summation sequence could be used for finding non-dominated set by deleting all the dominated solutions inside.
2. The bottom of the summation sequence is used as the start of the compared solution, while the top of the summation sequence is used as the start of the comparing one. Once the compared solution is observed to be dominated by any one anterior to it, it is deleted. When the compared solution becomes the top one of the sequence, the non-dominated set is drawn out.

One of the key steps of the sorting based algorithm is the scoring procedure. more anterior position the solution takes, more higher score the solution gains. But still, the scoring value should be under control. Generally, arithmetical progression and geometric progression are common used.

2.9 Paper 9: Multi-objective Optimization Using Dynamic Neighbourhood Particle Swarm Optimization

PSO is modified by using a dynamic neighbourhood strategy, new particle memory updating, and one-dimension optimization to deal with multiple objectives. Several benchmark cases were tested and showed that PSO could efficiently find multiple Pareto optimal solutions.

It uses the common evolutionary computation techniques:

1. It is initialized with a population of random solutions.
2. It searches for the optimum by updating generations.
3. The reproduction is based on the old generations. In PSO, the potential solutions, called particles, are “flown” through the problem space by following the current optimal particles.

Sharing many characteristics with other evolutionary algorithms, PSO could be a potential method

for multi-objective optimization. However, basic global and local version PSO algorithms are not suitable for there is no absolute global optimum in multi-objective functions. It is not easy to define a single gBest or IBest during each generation.

Compared with genetic algorithms (GAS), the information sharing mechanism in PSO is significantly different. In GAS, chromosomes share information with each other. So the whole population moves like a one group towards an optimal area. In PSO, only gBest (or IBest) gives out the information to others. It is a one-way information sharing mechanism. The evolution only looks for the best solution. All the particles tend to converge to the best solution quickly even in the local version.

Here a dynamic neighbourhood PSO is presented. In each generation, after calculating distances to every other particle, each particle finds its new neighbour’s. Among the new neighbors, each particle finds the local best particle as the 1Best. The problem is how to define the &stance and how to define the local best particle.

The algorithm used to search for local optima in each generation is defined as follows: 1. Calculate the distances of the current particle from other particles in the fitness value space of the first objective function (not the variable space). Find the nearest m particles as the neighbors of the

current particle based on the distances calculated above (m the neighbourhood size). Find the local optima among the neighbors in terms of the fitness value of the second objective function.

Another modification of the PSO is the update of the pBest. The pBest is the best position in particle's history. Only when a new solution dominates the current pBest, is the pBest updated.

2.10 Paper 10: Multi-objective Particle Swarm Optimization

Evolutionary algorithms (EAs) are search procedures based on natural selection [2]. They have been successfully applied to a wide variety of optimization problems [4]. Particle Swarm Optimization (PSO) [1,7] is a new type of evolutionary paradigm that has been successfully used to solve a number of single objective optimization problems (SOPs). However, to date, no one has applied PSO in an effort to solve multi-objective optimization problems (MOPs).

The particles of a swarm are arranged in a ring topology [6]. In this topology, the neighbourhood of particle i consists of particles i-l, i, and i+l. Figure 1 shows the topology of a five particle swarm. Using Figure 1 as an example, the neighbourhood of 0 would be 4, 0, and 1. The neighbourhood of 4, would be 3, 4, and 0.

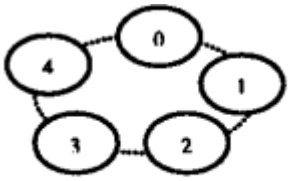

Figure 1: Swarm Topology

Figure 2 - provides an illustration of a particle.

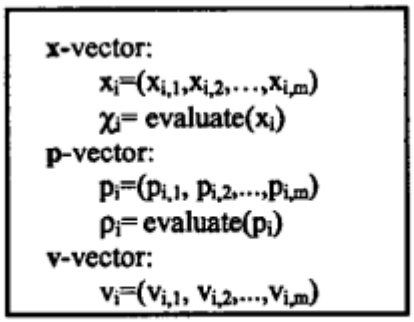

Figure 2: Elements Of A Particle

PSO Algorithm :

PSO begins by randomly initializing the x and v vectors. Initially, the p-vector is set equal to the x-vector. Each time the x-vector of a particle is updated, Zi is compared to Pi. If Zi is less than Pi, Pi

is set equal to xi. Therefore, the p-vector always contains the best PS discovered by a particle. To find the best PS in the neighbourhood of some particle i, the p values of i's neighbourhood are compared, and the index of the particle that has the best p is returned. Each parameter, d, of the x vector is then updated using the following equation: $v_{i,d} = v_{i,d} + \eta ((\varphi_{i,d} (p_{i,d} - x_{i,d})) + (\psi_{i,d} (p_{g,d} - x_{i,d})))$ In this equation, $\varphi_{i,d}$ and $\psi_{i,d}$ are random numbers between 0.0 and 1.0, 1"1 represents the learning rate, and g represents the particle in the neighbourhood with the best p. The new velocity value, Vi.d, is then checked to see if it is within the bounds [-vmax, vmax]. The value, vmax, denotes the maximum amount of change that can be applied to the values of the parameters. The x-vector is then updated by adding the v-vector. The updated x-vector represents a new PS that is located somewhere between xi, Pi, and pg.

Two experiments were conducted in order to test the effectiveness of our multi-objective particle swarm optimizer (MPSO).

The performances of our MPSO for the two experiments are shown in Tables 1 and 2. In both experiments, a swarm size of 20 particles was used. In the Tables, the leftmost column indicates the values that were used for vmax (1, 2, 4, 8, and 16) and the top row indicates the values used for r I (again we chose the values 1, 2, 4, 8, and 16). A total of 25 instances of our MPSO were compared. The value within each cell of the tables represents the average number of nondominated solutions found over 121 runs. For both experiments, one run consisted of 296 iterations.

	1	2	4	8	16
1	63	79	108	137	106
2	37	37	49	62	39
4	30	23	22	28	17
8	30	21	15	10	4
16	29	21	15	7	3

Table 1: Results of Experiment 1

	1	2	4	8	16
1	2849	2770	2906	3017	2957
2	3249	3287	3234	3230	3089
4	3233	3487	3817	4112	4406
8	3216	3477	3838	4160	4222
16	3211	3484	3838	4255	4425

Table 2: Results of Experiment 2

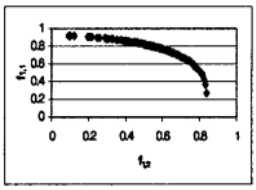

Figure 4: Graphed Results of Experiment 1

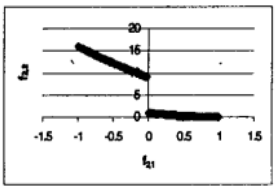

Figure 5: Graphed Results of Experiment 2

2.11 Integrated Summary of the Lecture studied

A *social network* is depicted by a social network graph G consisting of n number of nodes denoting n individuals or the participants in the network. The connection between node i and node j is represented by the edge eij of the graph. A *community* can be defined as a group of entities closer to each other in comparison to other entities of the dataset. A community is formed by individuals such that those within a group interact with each other more frequently than with those outside the group. *Clustering* is the process of grouping a set of similar items together in structures known as clusters[1].Most algorithms developed for community detection take advantage of single objective optimization methods which may be inefficient for complex networks. In this paper a multi-objective community detection based on a modified version of PSO is proposed [4]. The core idea of the PSO is to simulate the process of optimal in solution space as birds hunting. PSO can converge fast and has a few parameters [2]. In PSO, the behaviour of a flock of birds is used as a metaphor to optimize a problem. In this metaphor, a swarm of particles represents a flock of birds. Each particle corresponds to a bird, which moves through the search space to find the best position to obtain food. Initially, each bird moves randomly throughout the search space. However, it adjusts its direction based on two things: the position of the best food it has ever found so far (personal best) and the position of the best food found by the flock (global best) [3]. To solve the community detection problem more efficiently, in this paper, we adopt a novel approach based on multi-objective particle swarm optimization (MOPSO) called the MOPSO-Net. In MOPSO-Net, we adopted the Pareto dominance[8] concept to PSO to solve multi-objective tasks. The proposed method minimizes two objective functions Kernel K-means and the Ratio Cut. The framework of the proposed method can be explained in two main steps including; initialization and moving strategy. By using the nondominated sorting mechanism[8], a set of non-dominated solutions is obtained. To obtain local and global bests, the Normalized Mutual Information (NMI) value is computed for each particle at the nondominated set and then the particle with highest NMI is considered as the global best of the whole swarm.[4] while the modularity[2] and conductance is computed at each step. The analogy of particle swarm optimization with evolutionary algorithms makes evident the notion that using a Pareto ranking scheme could be the straightforward way to extend the approach to handle multi-objective optimization problems. The historical record of best solutions found by a particle could be used to store nondominated solutions generated in the past [7]. For this we have used the two datasets amongst all :

- *Karate club* network by Zachary[1]
- *Dolphin* Social Network [1]

3. REQUIREMENTS ANALYSIS

Description: Community detection is a process of identifying groups or clusters of nodes in a network where the nodes within the same group are more connected to each other than to nodes in other groups. This is an important area of study in network science and has applications in a wide range of fields, such as social network analysis, biological network analysis, and transportation network analysis.

3.1 Programming Language Used :

i. Python

3.2 Libraries

i. Matplotlib
ii. Pandas
iii. Numpy
iv. Networkx
v. Math
vi. networkx.algorithms.community
vii. random
viii. sklearn

3.3 Features

i. Find communities
ii. Evaluate communities
iii. Evaluating score of communities

4. MODELLING AND IMPLEMENTATION DETAILS

4.1 Control flow Diagram

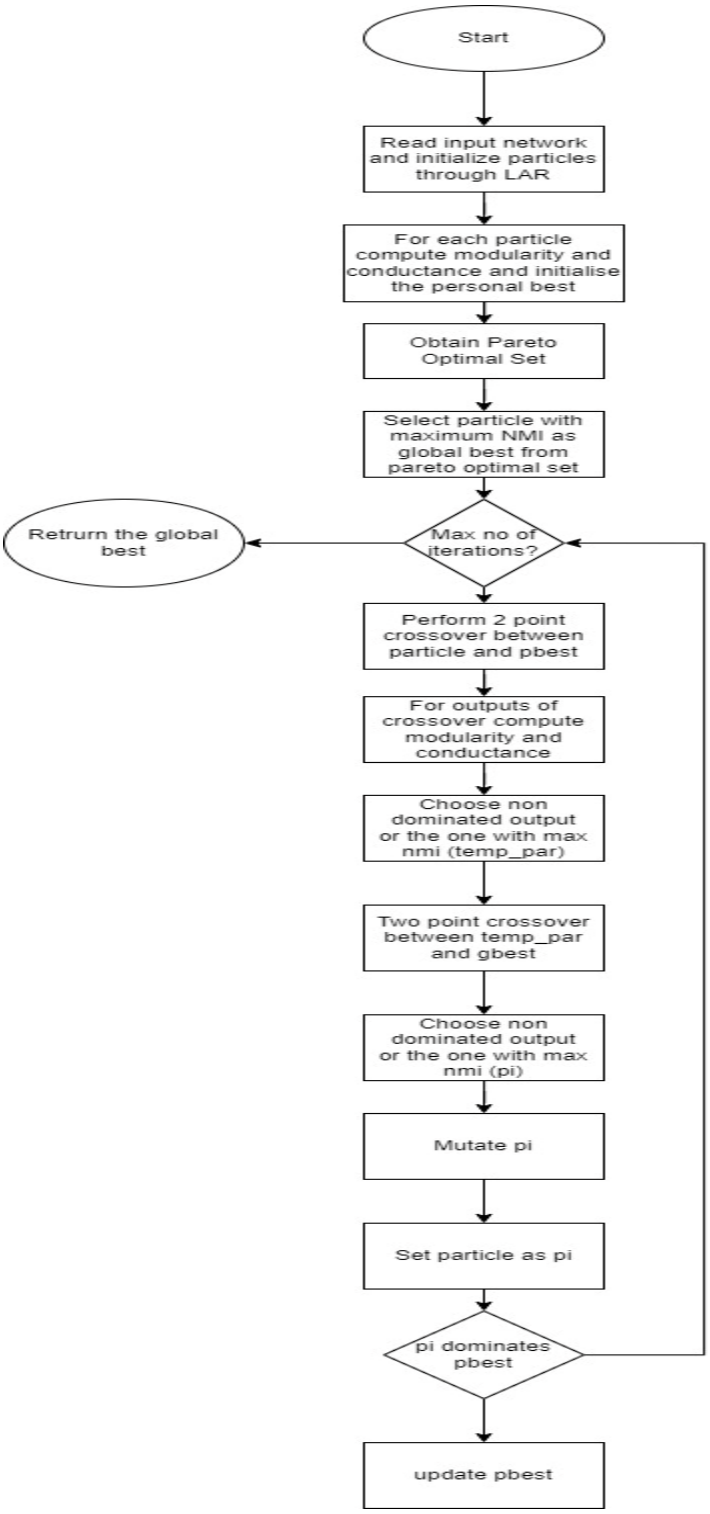

Fig.5 Flowchart

4.2 Implementation details

Multi-objective community detection is a type of community detection that considers multiple objectives or criteria simultaneously. This approach seeks to find communities in a network that optimize more than one criterion at the same time, such as maximizing modularity and minimizing the number of communities.

The process of multi-objective community detection involves the following steps:

1. Define the objectives
2. Choosing the optimization algorithm
3. Formulating the objective function
4. Initializing the algorithm
5. Evaluating the solutions
6. Applying selection, crossover, and mutation
7. Evaluating the new population
8. Choosing the final solution

Overall, multi-objective community detection can be a powerful tool for finding communities in complex networks that optimize multiple objectives simultaneously. By considering multiple objectives, this approach can provide a more nuanced understanding of the structure of a network and the relationships between its nodes.

4.2.1 Datasets used :

Name	Nodes	Edges	Type	Relationship
The bottlenose dolphin community of Doubtful Sound	62	159	Undirected	An undirected social network of frequent associations between 62 dolphins in a community living off Doubtful Sound, New

				Zealand.
Zachary's Karate Club Dataset	34	78	Undirected	Social network of friendships between 34 members of a karate club at a US university in the 1970s.

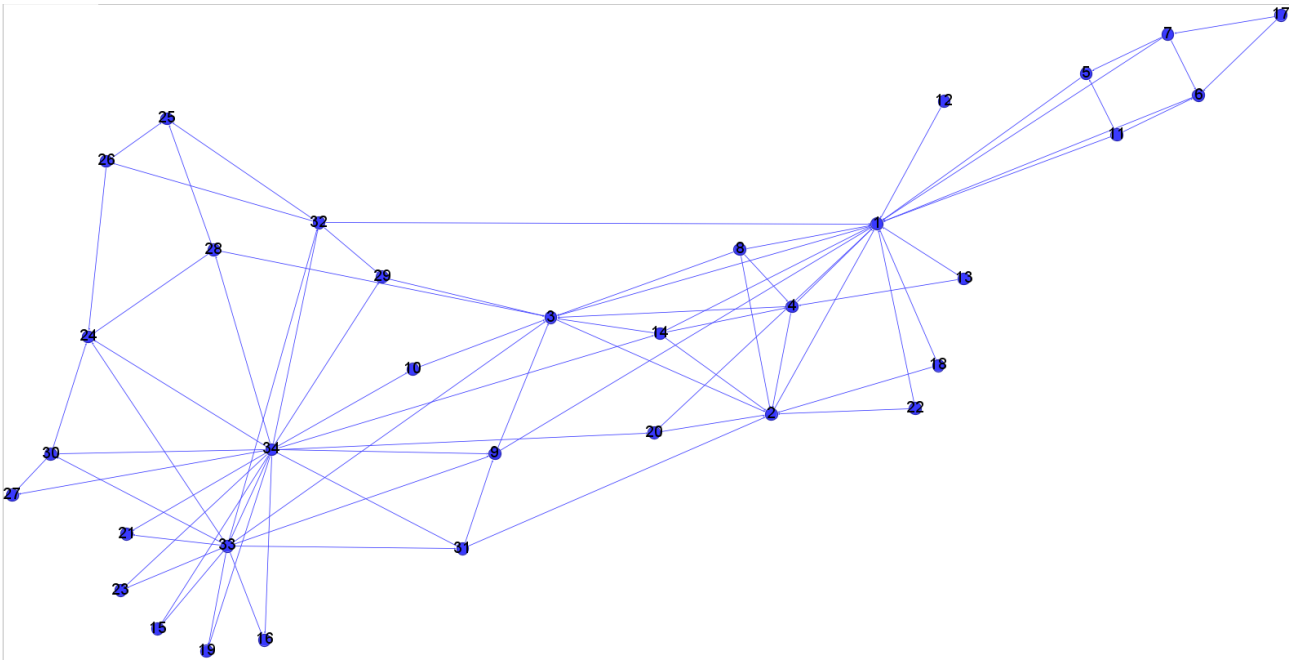

Fig.6 Karate Club Dataset

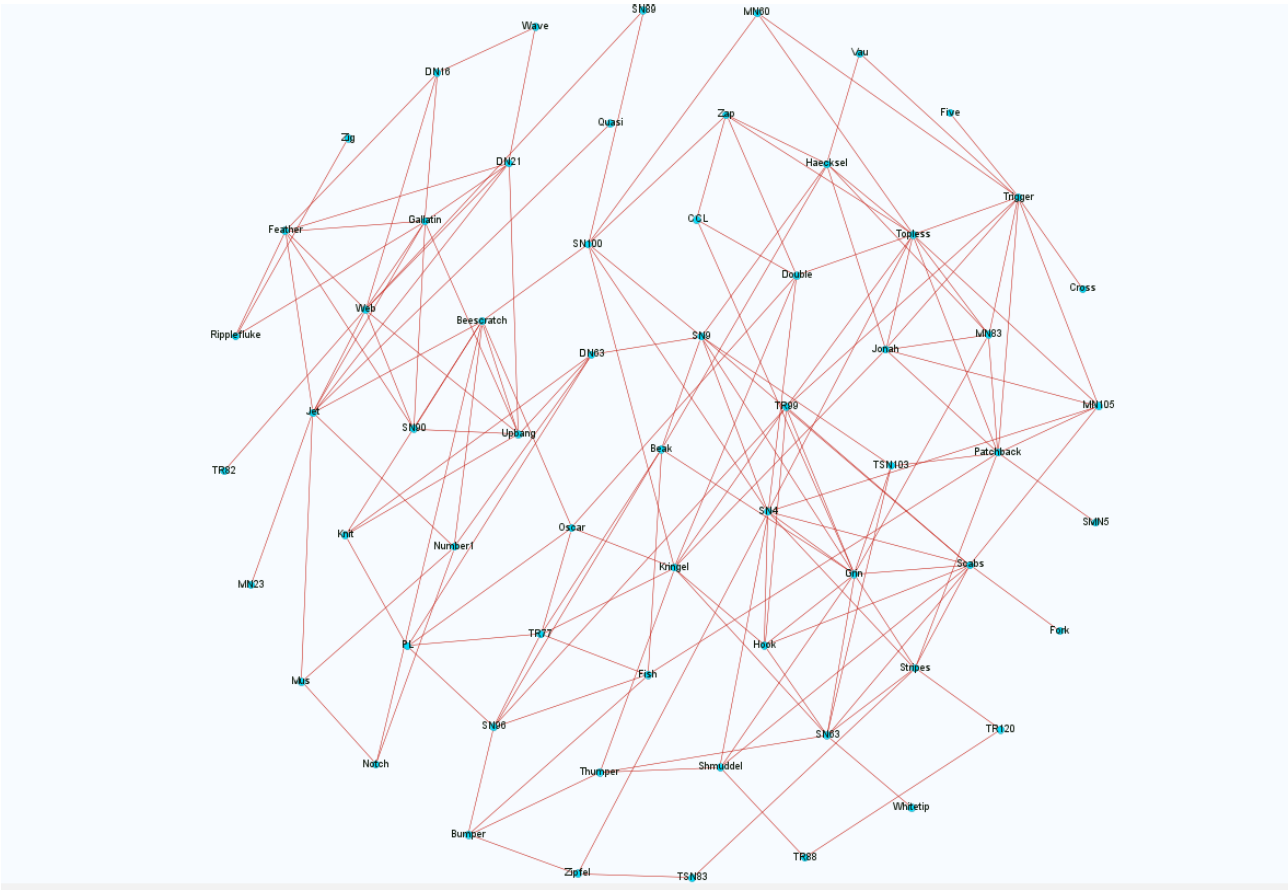

Fig.7 Dolphins dataset

4.2.2 PSO (Particle Swarm Optimization)

Particle Swarm Optimization (PSO) is a population-based optimization algorithm that was first introduced by Kennedy and Eberhart in 1995. It is inspired by the collective behavior of animals, such as flocks of birds or schools of fish, where individuals adjust their behavior based on the behavior of their neighbors.

In PSO, a population of particles moves around a multidimensional search space, searching for the optimal solution to a given optimization problem. Each particle in the swarm represents a candidate solution to the problem, and its position in the search space represents the values of the decision variables.

The basic steps of PSO are as follows:

1. **Initialization:** A population of particles is randomly generated in the search space, and each particle is assigned a random velocity.
2. **Evaluation:** Each particle is evaluated for its fitness based on the objective function.
3. **Updating the best positions:** Each particle updates its personal best position based on its own best position so far and the best position of its neighbors.

4. **Updating the velocity:** Each particle updates its velocity based on its current velocity, its personal best position, and the best position of its neighbors.
5. **Updating the position:** Each particle updates its position based on its current position and its velocity.
6. Repeat steps 2-5 until a stopping criterion is met.

The key parameters in PSO are the cognitive parameter (c1), the social parameter (c2), and the inertia weight (w). These parameters control the balance between personal and social influences on each particle's behavior. A higher value of c1 places more emphasis on personal experience, while a higher value of c2 places more emphasis on the experience of the swarm as a whole. The inertia weight controls the trade-off between exploration and exploitation. A high value of w promotes exploration, while a low value promotes exploitation.

PSO has been successfully applied to a wide range of optimization problems, including engineering design, function optimization, and feature selection in machine learning. It is a simple yet powerful algorithm that can quickly converge to a good solution with a relatively small computational cost. However, it may get stuck in local optima and may require careful tuning of its parameters to achieve good performance on a specific problem.

4.2.3 Functions used

1. **nx.read_gml():** The read_gml() function specifically allows you to read a graph from a file in GML (Graph Modeling Language) format and create a NetworkX graph object.
2. **np.random.randint():** The random.randint() function specifically allows you to generate random integers from a specified range.
3. **nx_comm.modularity():** It is used to calculate the modularity of a partition of a graph, which measures the quality of the division of nodes into communities or modules.
4. **pairwise_kernels():** The pairwise_kernels() function allows you to compute the kernel matrix or kernel similarities efficiently, without explicitly computing pairwise distances or similarities between all samples.
5. **normalized_mutual_info_score():** It is used to compute the normalized mutual information (NMI) between two clustering's or labelling's of the same dataset.

6. **kernel_kmeans():** Kernel K-Means is a variant of the traditional K-Means clustering algorithm that uses a kernel function to map the data into a higher-dimensional feature space.
7. **nx.conductance():** It is used to calculate the conductance of a set of nodes in a graph, which is a measure of how well-connected the set is to the rest of the graph.

5. Findings, Conclusion, and Future Work

5.1 Findings and result

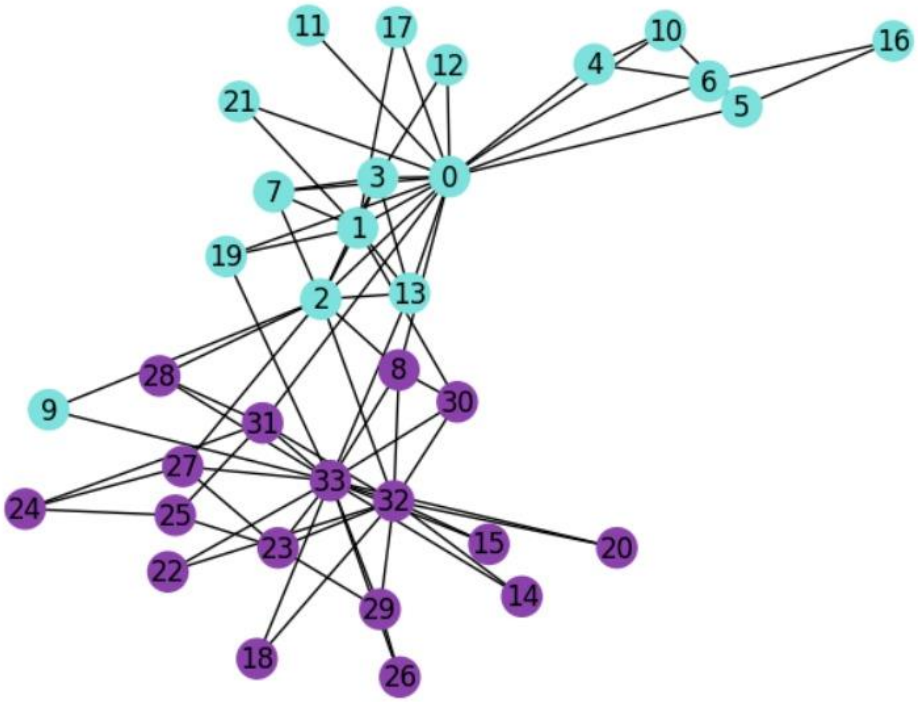

Fig.8 Communities in Karate-Club dataset

```
Ready
Set
Go
Iteration 100, NMI score: 0.8371694628777809
```

Fig.9 NMI value for Karate-Club dataset

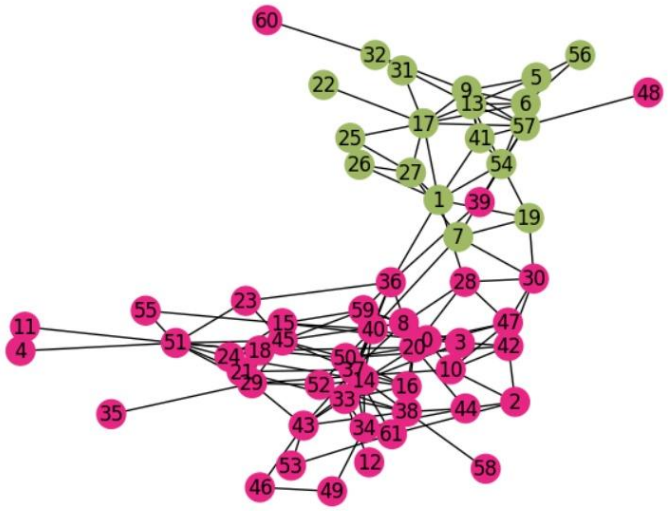

Fig.10 Communities in Dolphins dataset

```
Ready
Set
GO
Iteration 100, NMI score: 0.808250100060895
```

Fig.11 NMI value for Dolphins dataset

DATSET	NMI
Karate-Club	83.71%
Dolphins	80.8%

Fig.12 Resulted NMI

5.2 Conclusion

Overall, multi-objective community detection has been successfully applied to a wide range of real-world problems, including social networks, biological networks, and transportation networks. It can help researchers and practitioners gain insights into the underlying structure of complex systems and can inform decision-making processes in a variety of domains. However, careful consideration of the objectives, optimization algorithm, and objective function is necessary to obtain reliable and meaningful results, and further research is needed to address the challenges of scaling up to larger networks and incorporating dynamic aspects of community structure.

5.3 Future Work

The field of multi-objective community detection is constantly evolving, and there are several areas of future work that can further advance this technique. Some potential directions for future research include:

1. **Developing novel optimization algorithms:** Although several optimization algorithms have been used for multi-objective community detection, there is still room for developing new algorithms that can more efficiently and accurately optimize multiple objectives.
2. **Integrating dynamic community detection:** Most multi-objective community detection methods focus on static networks. However, in many real-world applications, networks change over time. Therefore, developing methods that can detect dynamic communities is an important direction for future research.

REFERENCES

1. Bedi, P., & Sharma, C. (2016). Community detection in social networks. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *6*(3), 115-135.
2. Wu, C., Li, T., Teng, F., & Chen, X. (2015, November). An improved PSO algorithm for community detection. In *2015 10th International Conference on Intelligent Systems and Knowledge Engineering (ISKE)* (pp. 138-143). IEEE.
3. Aote, S. S., Raghuwanshi, M. M., & Malik, L. (2013). A brief review on particle swarm optimization: limitations & future directions. *International Journal of Computer Science Engineering (IJCSE)*, *14*(1), 196-200.
4. Rahimi, S., Abdollahpouri, A., & Moradi, P. (2018). A multi-objective particle swarm optimization algorithm for community detection in complex networks. *Swarm and Evolutionary Computation*, *39*, 297-309.
5. Reyes-Sierra, M., & Coello, C. C. (2006). Multi-objective particle swarm optimizers: A survey of the state-of-the-art. *International journal of computational intelligence research*, *2*(3), 287-308.
6. Chaitanya, K., Somayajulu, D., & Krishna, P. R. (2018, July). A pso based community detection in social networks with node attributes. In *2018 IEEE Congress on Evolutionary Computation (CEC)* (pp. 1-6). IEEE.
7. Coello, C. C., & Lechuga, M. S. (2002, May). MOPSO: A proposal for multiple objective particle swarm optimization. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)* (Vol. 2, pp. 1051-1056). IEEE.
8. J. Du, Z. Cai and Y. Chen, "A Sorting Based Algorithm for Finding a Non-dominated Set in Multi-objective Optimization," *Third International Conference on Natural Computation (ICNC 2007)*, Haikou, China, 2007, pp. 436-440, doi: 10.1109/ICNC.2007.142.
9. Hu, X., & Eberhart, R. (2002, May). Multi-objective optimization using dynamic neighbourhood particle swarm optimization. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)* (Vol. 2, pp. 1677-1681). Ieee.
10. Moore, J., Chapman, R., & Dozier, G. (2000, April). Multi-objective particle swarm optimization. In *Proceedings of the 38th annual on Southeast regional conference* (pp. 56-57).
11. The bottlenose dolphin community of Doubtful Sound: D. Lusseau, K. Schneider, O. J. Boisseau, P. Haase, E. Slooten, and S.Dawson, *Behavioral Ecology and Sociobiology* **54**, 396-405 (2003)
12. Zachary's karate club: W. W. Zachary, An information flow model for conflict and fission in small groups, *Journal of Anthropological Research* **33**, 452-473 (1977).