Mutual connection of Evolutionary algorithms and Complex networks

Bc. Iftekharuddin Syed

Master's thesis
2023

Tomas Bata University in Zlín
Faculty of Applied Informatics

Tomas Bata University in Zlín
Faculty of Applied Informatics
Department of Informatics and Artificial Intelligence

Academic year: 2022/2023

ASSIGNMENT OF DIPLOMA THESIS

(project, art work, art performance)

Name and surname:	**Iftekharuddin Syed**
Personal number:	**A19768**
Study programme:	**N3902 Engineering Informatics**
Branch:	**Information Technologies**
Type of Study:	**Full-time**
Work topic:	**Relace mezi evolučními algoritmy a komplexními sítěmi**
Work topic in English:	**Mutual Connection of Evolutionary Algorithms and Complex Networks**

Theses guidelines

1. Literature survey on a given topic.
2. Create a complex network model based on real data.
3. Design a system for transforming the dynamics of a selected evolutionary algorithm into a suitable representation using a complex network.
4. Analyze the settings of the control parameters of the algorithm and the selection of test problems so that the dynamics of the algorithm transformed into a complex network corresponds to the model of the real network.
5. Verify the possibilities of predicting the dynamics of a complex network based on the created model and draw a conclusion.

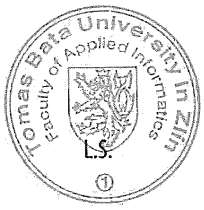
Form processing of diploma thesis: **printed/electronic**

Recommended resources:

1. YIN, Peng-Yeng, ed. *Trends in developing metaheuristics, algorithms, and optimization approaches*. Hershey, PA: Information Science Reference, c2013, 1 online zdroj (xxx, 345 s.). ISBN 9781466621466. Dostupné také z: http://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=454300&lang=cs&site=ehost-live&authtype=ip,shib&custid=s3936755.
2. KACPRZYK, Janusz a Witold PEDRYCZ, ed. *Springer handbook of computational intelligence*. Dordrecht: Springer, 2015, lvi, 1633 s. ISBN 9783662435045.
3. ZELINKA, Ivan, Václav SNÁŠEL a Ajith ABRAHAM, ed. *Handbook of optimization: from classical to modern approach*. Berlin: Springer, c2013, xii, 1100 s. Intelligent systems reference library. ISBN 9783642305030.
4. KENNEDY, James, Russell C EBERHART a Yuhui SHI. *Swarm intelligence*. San Francisco: Morgan Kaufmann, c2001, xxvii, 512 s. ISBN 1-55860-595-9.
5. ESTRADA, Ernesto. *The structure of complex networks: theory and applications*. New York: Oxford University Press, 2012, xii, 465 stran. ISBN 978-0-19-959175-6.
6. LATORA, Vito, Vincenzo NICOSIA a Giovanni RUSSO. *Complex networks: principles, methods and applications*. Cambridge: Cambridge University Press, 2017, xxi, 559 s. ISBN 9781107103184.
7. CHEN, G., Xiaofan WANG a Xiang LI. *Fundamentals of complex networks: models, structures, and dynamics*. Singapore: Wiley, 2015, xiv, 366 s. ISBN 9781118718117.

Supervisors of diploma thesis: **doc. Ing. Roman Šenkeřík, Ph.D.**
Department of Informatics and Artificial Intelligence

Date of assignment of diploma thesis: **December 2, 2022**
Submission deadline of diploma thesis: **May 26, 2023**



doc. Ing. Jiří Vojtěšek, Ph.D. m.p.
Dean

prof. Mgr. Roman Jašek, Ph.D., DBA m.p.
Head of Department

In Zlín December 7, 2022

Name of the student: Iftekharuddin Syed

Thesis topic: Mutual connection of Evolutionary algorithms and Complex networks

I hereby declare that:

- I understand that by submitting my Master´s thesis, I agree to the publication of my work according to Law No. 111/1998, Coll., On Universities and on changes and amendments to other acts (e.g. the Universities Act), as amended by subsequent legislation, without regard to the results of the defence of the thesis.
- I understand that my Master´s Thesis will be stored electronically in the university information system and be made available for on-site inspection, and that a copy of the Master´s Thesis will be stored in the Reference Library of the Faculty of Applied Informatics, Tomas Bata University in Zlín, and that a copy shall be deposited with my Supervisor.
- I am aware of the fact that my Master´s Thesis is fully covered by Act No. 121/2000 Coll. On Copyright, and Rights Related to Copyright, as amended by some other laws (e.g. the Copyright Act), as amended by subsequent legislation; and especially, by §35, Para. 3.
- I understand that, according to §60, Para. 1 of the Copyright Act, TBU in Zlín has the right to conclude licensing agreements relating to the use of scholastic work within the full extent of §12, Para. 4, of the Copyright Act.
- I understand that, according to §60, Para. 2, and Para. 3, of the Copyright Act, I may use my work - Master´s Thesis, or grant a license for its use, only if permitted by the licensing agreement concluded between myself and Tomas Bata University in Zlín with a view to the fact that Tomas Bata University in Zlín must be compensated for any reasonable contribution to covering such expenses/costs as invested by them in the creation of the thesis (up until the full actual amount) shall also be a subject of this licensing agreement.
- I understand that, should the elaboration of the Master´s Thesis include the use of software provided by Tomas Bata University in Zlín or other such entities strictly for study and research purposes (i.e. only for non-commercial use), the results of my Master's Thesis cannot be used for commercial purposes.

- I understand that, if the output of my Master´s Thesis is any software product(s), this/these shall equally be considered as part of the thesis, as well as any source codes, or files from which the project is composed. Not submitting any part of this/these component(s) may be a reason for the non-defence of my thesis.

I herewith declare that:

- I have worked on my thesis alone and duly cited any literature I have used. In the case of the publication of the results of my thesis, I shall be listed as co-author.
- That the submitted version of the thesis and its electronic version uploaded to IS/STAG are both identical.

In Zlín; dated: 26.05.2023 .....
Student´s Signature

ABSTRAKT

Cílem této diplomové práce je prostudovat a popsat vzájemnou souvislost mezi dynamikou evolučních algoritmů a komplexními sítěmi. Pro samotné testování byl pořízen soubor reálných dat týkajících se citací mezi členy katedry FAI UTB k vytvoření reálné citační síě. V rámci řešení této diplomové práce byl poté prověřen a implementován systém pro zachycení dynamiky algoritmu PSO (Particle Swarm Optimization) a především pak transformace zachycené dynamiky do podoby komplexní sítě. Originálním přínosem této práce je, že hyperparametry algoritmu PSO jsou optimalizovány pomocí různých technik tak, aby zachycená dynamika algoritmu PSO co nejblíže odpovídala citační síti vytvořené z reálných dat. Nakonec jsou porovnány provedené předpovědi dynamiky zachycené algoritmem PSO a pomocí regresních modelů za krátké období.

Klíčová slova: Optimalizace rojením částic, evoluční algoritmy, regresní modely, komplexní sítě, citační sítě.

ABSTRACT

The goal of this thesis is to study and understand the mutual connection between evolutionary algorithms and complex networks. To begin with, a set of real data is taken related to citation between the faculty members from department of Informatics and Artificial Intelligence, UTB and a citation network is created. Then a system is implemented to capture the dynamics of Particle Swarm Optimization (PSO) algorithm and transform the captured dynamics into complex network. The original contribution of this work is that the hyperparameters of the PSO algorithm are then optimized using different techniques, so that the captured dynamics of the PSO algorithm correspond closer to citation network created from real data. At last predictions are made about the dynamics captured from the PSO algorithm using regression models over a period.

Keywords: Particle Swarm Optimization, Evolutionary Algorithms, Regression models, Complex network, Citation network.

I want to express my sincere appreciation to my supervisor, Prof. Ing. Roman Šenkeřík, Ph.D., and co-supervisor, Ing. Tomáš Kadavý, for their unwavering support throughout the development of this thesis. Their guidance, feedback, and encouragement have been invaluable to me.

I am also grateful to my teachers, doc. Ing. Zuzana Komínková Oplatková, Ph.D., Ing. Peter Janků, Ph.D., Ing. Pavel Vařacha, Ph.D. and Prof. M.Sc. Roman Jašek, Ph.D., DBA., their insightful courses and class work, which laid the foundation for this thesis. Special thanks also to doc. Ing. Marek Kubalčík, Ph.D., for his support throughout the studies.

I want to thank my parents and siblings for their love and encouragement in making my studies possible.

I hereby declare that the print version of my Master's thesis and the electronic version of my thesis deposited in the IS/STAG system are identical.

CONTENTS

INTRODUCTION

Considering real-world problems like traffic optimization, internet routing, supply chain optimization, epidemiology, and so on, these problems affect our daily lives in one way or another. Therefore, solving these categories of problems is essential to improve our living standards and sustainability. All these problems and many others can be transformed into networks by abstracting the underlying details and using some general methods for solving and analysis. This analysis can help us understand the problem in detail and improve the existing solution. Solutions using a complex network to solve such problems and analysis is of growing interest among the research community [1].

Complex networks are everywhere, from artificial networks like railway and social networks to natural networks like protein-protein interaction networks, neural networks, etc. On the other hand, evolutionary algorithms are inspired by biological evolution and are powerful computational tools for problem-solving and optimization. Understanding the connection between the two fields of study can help us solve the problem better by converting the problem as a network, and the algorithm could be better fine-tuned to a specific problem or improve the overall robustness or performance.

The primary objective of this thesis is to examine the mutual relationship between Evolutionary algorithms and complex networks. The intention is to leverage the characteristics of complex networks to understand the dynamics of Evolutionary algorithms better and make predictions concerning the dynamics of the network structures captured from these algorithms. The goals of this thesis are divided as follows:

- Develop a literature search on the given topic.
- Build a complex network model based on real data.
- Design a system for transforming the dynamics of the selected evolutionary algorithm into a suitable representation using a complex network.
- Analyze the settings of the control parameters of the algorithm and the choice of test problems so that the dynamics of the algorithm transformed into the form of a complex network correspond to the real network model.
- Verify the possibilities of predicting the dynamics of a complex network based on the created model and make a conclusion.

The thesis starts with the literature survey, where all the theoretical concepts needed in the thesis are presented in detail. The literature part mainly deals with complex networks and concepts related to heuristics. The practical part begins with section 3, where a citation network is created based on real data. It is followed by section 4, using a methodology for capturing the dynamics of evolutionary algorithms and creating a complex network based on the dynamics. The parameters of the evolutionary algorithms are then optimized using different techniques in section 5. Finally, section 6 focuses on predicting the complex network's behavior by applying various methods over a designated timeframe.

The research presented in this thesis may help other readers to gain knowledge about understanding the usage of complex networks along with evolutionary algorithm to capture dynamics and make some predictions about the dynamics of evolution algorithms.

I. THEORY

1 COMPLEX NETWORKS

This section concerns the theory related to complex networks [2]. It starts with a background about networks in general and the history behind the usage of networks, important terminologies, and parameters concerning networks and network models. It also focuses on the characteristics of network models. The section also discusses complex networks, their characteristics, and network centralities.

1.1 Networks - meaning and definition.

As per the definition of the word network from the Cambridge dictionary [3] "*a large system consisting of many similar parts that are connected together to allow movement or communication between or along the parts, or between the parts and a control center*"

With the definition mentioned above, the general opinion about a network is that it is viewed as several objects connected. Referring to this basic definition, the connection or interconnection of objects is the most crucial attribute of a network. [3]

Networks are everywhere railways, friends, phone networks, electric grids, friends connected on social media representing the social network, and many networks out there. Therefore, networks are an excellent universal abstract representation of a particular system. The crucial property of networks is that there is a general method to represent any given system (natural or artificial) by abstracting the underlying details, which helps understand the system's behavior - *Universal Abstract representation.* Most networks from real-world data represent some part with regularity and others with randomness; hence such networks are *not regular but not random*. The networks can help us understand the system's behavior and the reason for these regularities and differences. Surprisingly, networks from various fields of science and biology, like social networks, metabolic networks, biological networks, transportation networks, and so on, possess very similar properties (Global Properties of Networks). Networks, in general, are *complex systems,* which means if one takes the system and cuts it into pieces, one can study each piece separately. But we need to find out how the system works overall. Therefore, one needs to study the system wholistically as in its entirety. For example: like humans are complex systems, you can study how the kidneys, eyes, brains work, etc. But studying each of the organs does not tell how humans work. [4]

Generalizing the concept of a network in technical terms: Most systems can be graphically represented as a network consisting of *nodes(vertices)* that designate entities, and these

nodes are connected using *links (edges, relations)*; for example, a network of computers where computers are nodes and the connection between them are cables. The entities (or nodes) represent any object like railway stations connected by railway lines (links), routers connected by cables, hyperlinks of webpages connected by URL, and so on. [4]

Network and graph are the terms that are used interchangeably in the literature. Euler was the first who introduce the concept of a graph in 1735 to solve the problem related to the seven bridges in the East Prussian city of Königsberg. The problem was finding a path through the seven bridges traversing over the river Pregel and an island, Kneiphof, such that one should cross each bridge once and only once. *Figure 1* shows Euler's idea of transforming the seven-bridge problem into a graph. [5]

Figure 1: Representing Euler's idea transforming seven bridge problem into a graph [6]

Euler abstracted the real problem as a graph with nodes representing a piece of land on the island and edges representing bridges connecting the island. Euler suggested two solutions to the seven-bridge problem mentioned above: Firstly, a graph can be traversed without passing twice on the same edge if all its nodes have an even number of connections (degree of the node). Secondly, if two of them have an odd number of connections, it is necessary for this second solution that one starts from one of the two nodes with an odd number of connections ending the walk on the other one. This concept of abstraction into a graph introduced by Euler was so elegant that it could help solve a wide range of similar classes of problems that fall under this category. [5]

Links or connection between the nodes plays a major role in network classification. This classification depends on the weight of the link and the type of link. The weight of a link is some positive number assigned to the link, and the network is called a weighted network. Weight can represent various real-world concepts like the interaction between the

entities, the transfer of information in a certain time interval, etc. The type of link also differs in a network; if two nodes are connected using a simple line, then it is a simple network. If nodes are connected using some directed lines, then it is called a directed network, and similarly undirected network for undirected links between the nodes. There are also a pseudo networks in which one node is connected to itself using links, and such links are called self-loops. [7]

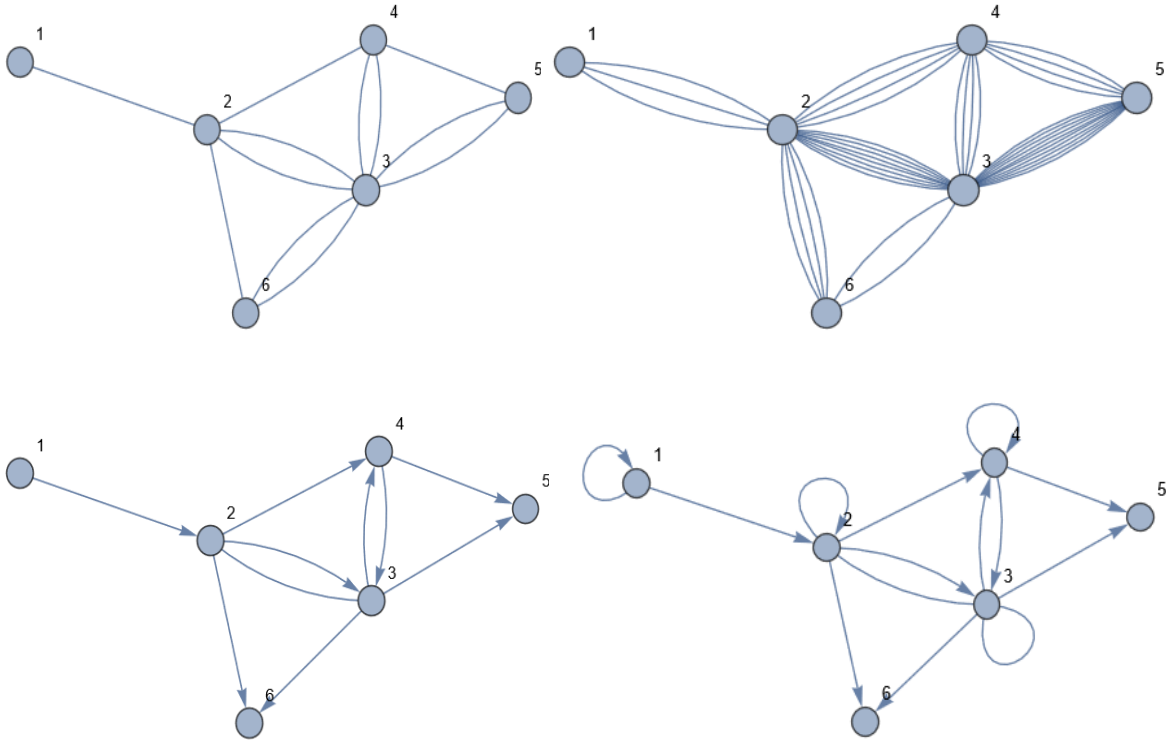


Figure 2: Types of networks based on links between the nodes - top leftmost is simple network, top rightmost is multi-link network, bottom leftmost is directed network and bottom rightmost is self-loop or pseudo network.

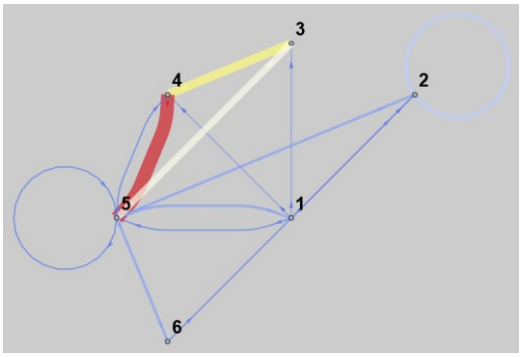


Figure 3: Network with weights on edges.

The weight of the edge can also be used to represent the number of links or interactions between the two nodes. The thicker the edge between the node the more is the interaction between the nodes. As shown in the *Figure 3*, the red line indicates the edge with more weight and hence more edges between the nodes 4 and 5, than other nodes. Apart from nodes and edges, some other vital concepts help understand a network. These include a description of nodes, their connections, and their role in a network. [7]

1.2 Network models

Network models are graph models with specific characteristics that help us categorize a network. Some of the standard network models are as follows:

1.2.1 Random Network model

This model is also called the *Erdős-Rényi network* after the name of the mathematicians that helped in understanding this network [8]. Random networks are the networks created by randomly placing links between the nodes. It has two definitions proposed:

a) $G\ (N, L)$ model: The number of nodes N are randomly connected to links L. The average node degree is fixed for a particular node and is calculated as $\langle k \rangle = 2m/n$.

b) $G\ (N, p)$ model: The set of nodes N are connected based on probability p. The probability that two nodes are connected is fixed.

The steps to construct a random network are as follows:

- Given N disconnected nodes.
- For each pair of nodes, randomly generate a number between 0 and 1. If the number exceeds fixed probability p, then the nodes are connected; otherwise, they are not.

- The second step is repeated for each $N(N\text{-}1)/2$ pairs of nodes.

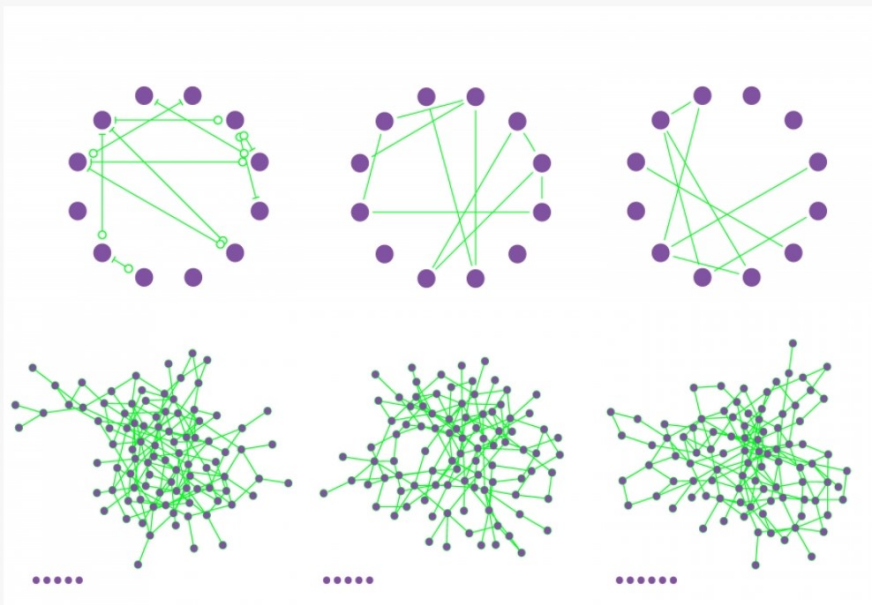


Figure 4: Random networks with same probability and number of nodes [9]

Random networks are truly random in nature, the above *Figure 4* shows three networks with the same probability and number of nodes as p=1/6 and N=12 but different networks are created even though there are same parameters. [9]

The typical characteristics of a Random Network are as follows [9]:

- High node degree, which corresponds to a short average path length.
- Degree distribution follows Normal Distribution.
- Small Betweenness centrality.
- High Transitivity.

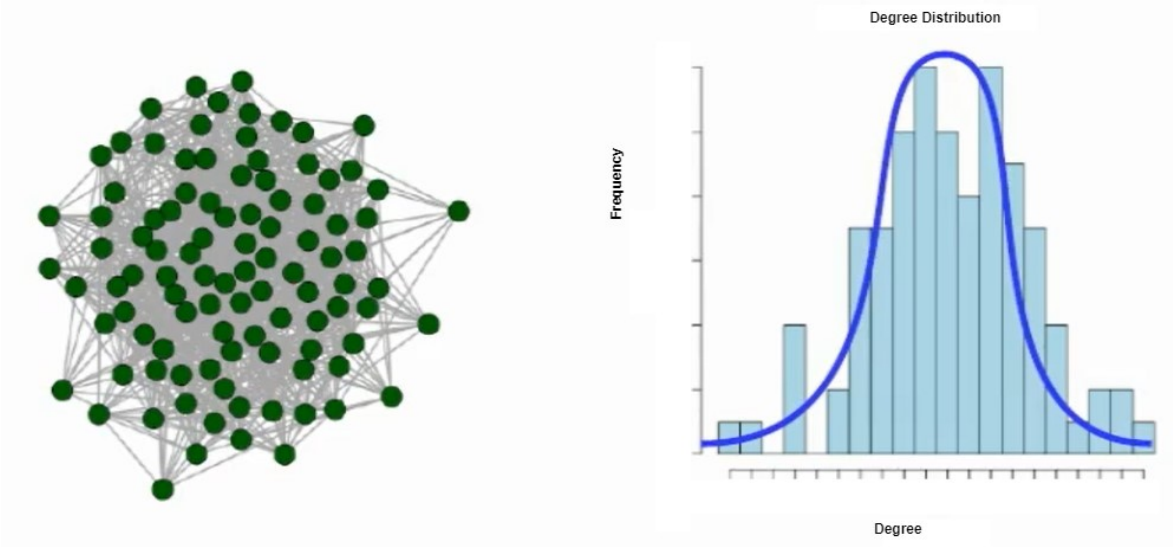


Figure 5: Degree distribution for Random networks [9]

The above *Figure 5* shows degree distribution for random networks, as shown the distribution follows Normal distribution.

1.2.2 Small World Network

The small-world networks are based on the *small-world phenomenon* (also named as *six degrees of separation*), which states that everyone in this world can meet any other individual with just six acquaintances (handshakes) between them. And the individuals living in the same city are just a few handshakes away from each other. [9]

The small world networks generate lattice, nodes are initially linked to *k* closest neighbors and apply rewiring probability.

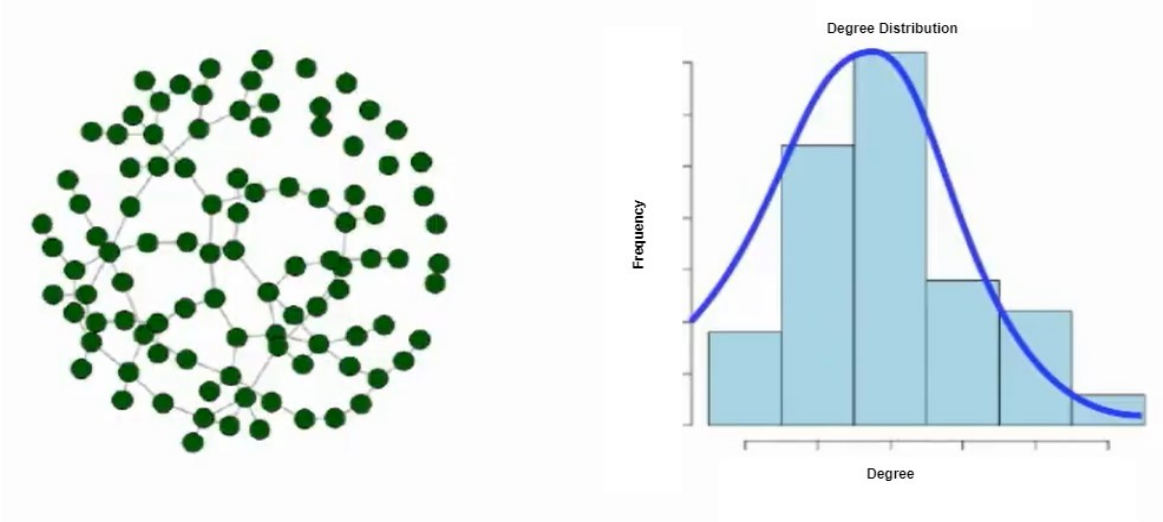


Figure 6: Degree distribution for typical small world network [9]

The *Figure 6* above shows degree distribution of a sample small world network, as seen from the figure the degree distribution follows Poisson Distribution.

The characteristics of small-world networks are: [9]

- Small node degree because they follow "*six-degree separation*" principle.
- Degree distribution follows Poisson Distribution.
- Betweenness smaller than Scale free.
- Transitivity: smaller than Random, Higher than scale-free.

1.2.3 Scale Free Networks

Scale-free networks have few nodes with large node degrees and many nodes with small node degrees (80/20 rule). This node degree distribution is an essential feature for a scale-free network, and it makes the scale-free networks robust and resilient, as the failure of one node will not lead to the collapse of the entire network. [9]

The scale-free networks are the networks with following properties [9]

- Relatively smallest node degrees.
- The degree distribution follows a Power law.
- Highest betweenness centrality and

- Smallest transitivity.

The scale-free network working principle is as follows: taking nodes and start connecting these nodes with the main idea that if a node has a high degree, then the probability that a new link is attached to that node is higher than what it is for a node that has a lower degree. In short, the probability that a node is connected to another node is proportional to its degree. Forming a scale-free network using the above method translates into the *Preferential Attachment principle*, where the rich get richer. [9]

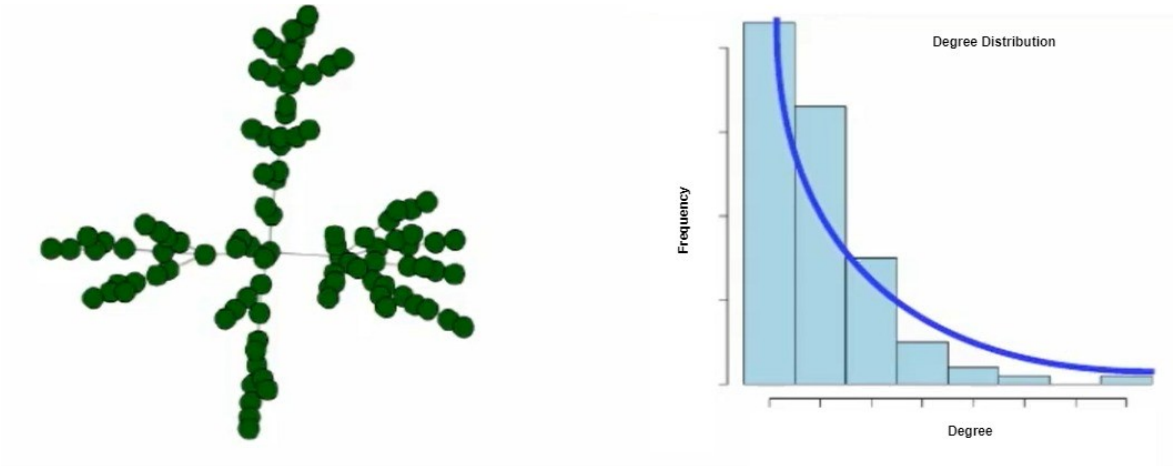


Figure 7: Degree Distribution for sample scale free network [9]

Using this approach, generating a network whose degree distribution follows a Power law. The power law is essentially what is seen in the *Figure 7*. If a network is directed, the scale-free property applies separately to the in- and the out-degrees.

1.3 Important terminology about networks/graphs.

This sub-section is related to important terminology and properties of a network. These properties help in understanding the behavior of the network and make some estimations.

Adjacent nodes: Adjacent nodes are the ones which are connected by the same common edge. [9]

Incident node: If a node is placed on an edge, then it is called incident. [9]

Degree: The total number *N* of edges or links connected to a node represents degree k_i of a node v_i. The equation (1) **Error! Reference source not found.** shows the degree of a node. [9]

$$k_i = |N(v_i)| \tag{1}$$

Average node degree: Average node degree is the average of degrees of all the nodes in a graph. If average node degree is represented by <*k*> then it also equals to twice the total degree *m* of the node divided by the number of nodes *n.* Average node degree can even be shown as twice the number of edges *E* divided by the number of total number of vertices *V.* The equation (2) shows all the relations mentioned. [9]

$$< \mathrm{k} >= \frac{1}{n}\sum_i k_i = \frac{2m}{n} = \frac{2|E|}{|V|} \tag{2}$$

Incoming degree: Incoming node degree is the total count of edges directed to node.

Outgoing degree: Outgoing node degree is the total count of edges directed away from the node.

In a directed graph total node degree

$$k_i = \textit{Incoming degree}\ (k_i^{in}) + \textit{Outgoing degree}\ (k_i^{out})) \tag{3}$$

The average of incoming degrees $< k_i^{in} >$ and outgoing degrees $< k_i^{out} >$ in a directed graph are also equal. [9]

$$< k_i^{in} >= \frac{1}{n}\sum_i k_i^{in} =< k_i^{out} >= \frac{1}{n}\sum_i k_i^{out} = \frac{m}{n} = \frac{E|}{|V|} \tag{4}$$

Degree distribution: Degree distribution in simple terms is counting how many nodes of which node degree is present in a graph. In a connected graph minimum degree of each node is 1 (because each node is connected at-least once) and with every degree there will be certain

number of nodes with that degree. So degree distribution $P(k)$ *or* P_k is just fraction of nodes with a particular degree. If k_i is node degree and n_k is number of nodes which has degree k and total nodes $n = \sum_k n_k$ then degree distribution is a ratio of nodes with degree k to the total number of nodes n.

$$P(k) = P_k = \frac{n_k}{\sum_k n_k} = \frac{n_k}{n} \tag{5}$$

The node degree is important in a graph with small number of nodes and degree distribution is important when there are large number of nodes. The *Figure 8* shows a sample network and corresponding degree distribution.

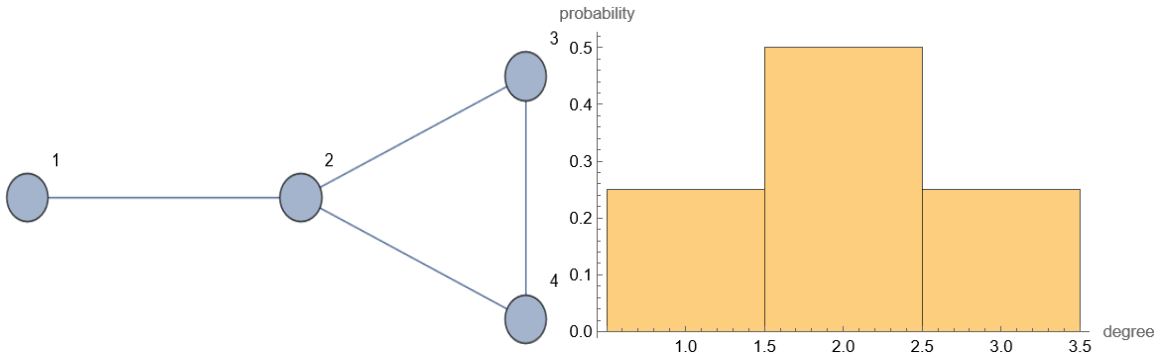


Figure 8: Sample network and corresponding degree distribution

Path: A path is a track of edges that connects two vertices. A graph is called *connected* if there is a path in between two vertices in that network. Graph may have one portion as connected and some portion not connected to the main graph, then the maximum portion of the graph which is connected (leaving smaller portion which is not connected) is called *Connected component.* [9]

Distance: In a connected component (or connected graph), the distance $d_G(v_i, v_j)$between the nodes is the count of edges in the shortest path between two nodes v_i and v_j in a graph. [9]

Diameter: Diameter is a parameter for a graph denoted by *D*, it is the longest shortest path in that graph for a pair of nodes. The diameter represents any longest path (smallest path) between two nodes in a particular graph. The diameter of a graph can be misleading sometimes because it gives information only about the largest shortest path in a graph while other parts of the graph may have smaller diameter. [9]

$$D = max_{i,j} d_G(v_i, v_j) \tag{6}$$

Average path length: Average path length denoted by <*L*> for a graph is the average number of edges needed to travel from one node to another. This parameter is preferred over the diameter of a graph because average path length can smaller while still diameter is higher.

$$< L >= \frac{1}{n(n-1)} \sum_{i \neq j} d_G(v_i, v_j) \tag{7}$$

Transitivity or global clustering coefficient: Transitivity *T* of a graph is a property that tells us how many triangles there in a graph are. If the node *i* is connected to *j* and *j* is connected to *h,* then *i* is connected to *h.* Transitivity is a global matric and gives information about the entire graph. [9]

$$T = \frac{3 \text{ x } number\ of\ triangles(triads)}{number\ of\ connected\ triplets\ of\ vertices} \tag{8}$$

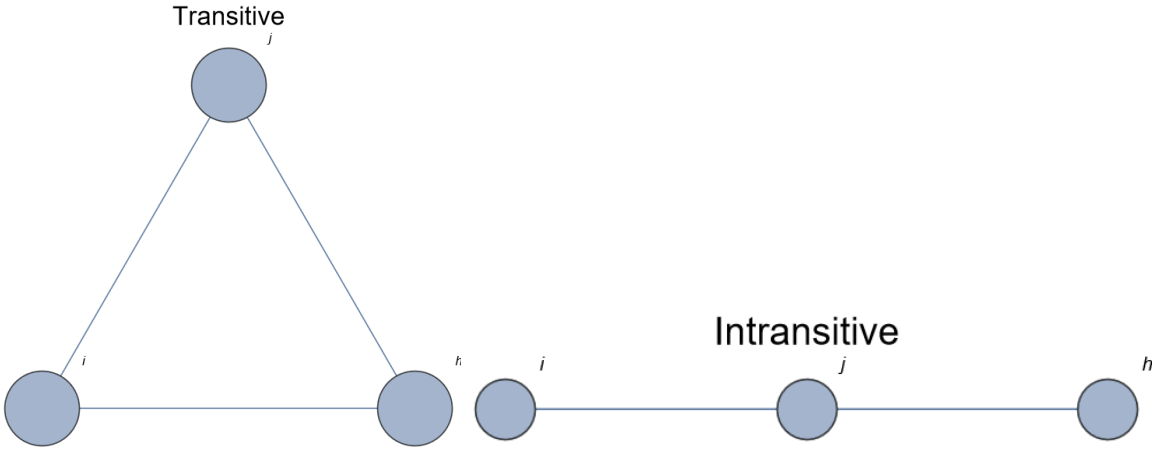


Figure 9: Graphs representing transitive and intransitive networks.

Clustering coefficient or Local clustering coefficient: Clustering coefficient parameter is related to nodes in a network. It is a fraction of the number of links between nodes in the neighborhood to the maximum possible (total possible) number of nodes. In simple terms clustering coefficient is about how the triangles are present around a particular node i.e., if a particular node belongs to all possible triangles that could exist around it or not. As an analogy to understand the clustering coefficient, for example, in a social network like Facebook, a friend of your friend may be your friend, or two of your friends might be direct friends. [9]

$$C_i = \frac{number\ of\ links\ in\ between\ neighbouring\ nodes\ of\ a\ particular\ node\ i}{k_i(k_i-1)/_2} \quad (9)$$

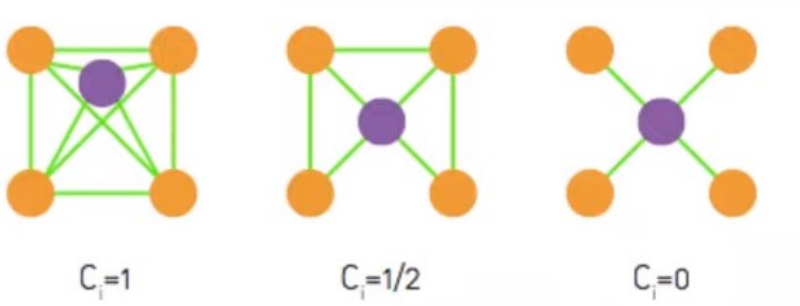


Figure 10: Sample graphs for clustering coefficients [12]

$$< C >= \frac{1}{n}\sum_{i=1}^{n} C_i \quad ,where\ n\ in\ total\ number\ of\ nodes\ in\ a\ graph \quad (10)$$

1.4 Characteristics of Complex Networks and Centralities

This sub-section is concerned with characteristics of complex networks, followed the different centrality measured used in this thesis. The centralities discussed are limited and specific to what is used in the practical part, there are other centralities, but they are out of scope for this study.

The distinct properties that most complex networks possess are as follows [9]:

- Scale-free networks: In mathematical terms, the graph of the node degree distribution follows a specific shape known as the Power law. So most complex networks show a resemblance to power law distribution shape.
- Small-world networks: Complex networks generally have small diameters and small average path lengths, which means these networks are generally very tight. That means traveling from one node to another will take very few steps.
- Transitive networks: Complex networks possess a high clustering coefficient, which means there are seen many triangles in a complex network.

1.4.1 Node centrality

Node centrality is a measure that helps understand the node's role in each network. These centrality measures are essential for analyzing various network tasks like the importance of nodes, network behavior, how individual nodes are arranged in a network, and understanding what's happening in the network. There are many centrality measures in the literature and the usage of each measure depends on what is being looked for in the network; the most common and important ones are as follows. [7]

1.4.1.1 Degree Centrality

The most basic centrality measure for a node is the degree centrality C_D. It is calculated as the number of edges k connected from the node *i* to node *j*. The following are the mathematical expressions to calculate the degree centrality for an undirected graph is shown in equation (11).

$$C_D(i) = k(i) = \sum_j C_{ji} = \sum_j C_{ij} \tag{11}$$

Normalized degree centrality C_D^* is given by equation (12) for *n* nodes in the network:

$$C_D^*(i) = \frac{1}{n-1} C_D(i) = \frac{k(i)}{n-1} \tag{12}$$

Normalized degree centrality is used when comparing two graphs, one with many nodes and the other with a relatively small number of nodes. If a normalized value is not taken, then the larger network will have incomparable values (as it will have nodes with higher degrees) relative to the smaller network. The normalized value is calculated by dividing the degree centrality of a node by the number of nodes in the network (neglecting the node under consideration) hence *n-1* (maximum number of connections a node can have).

In case of a directed graph, the total degree is the total sum of in-degree k_{in} and out-degree k_{out}. In-degree is the number of edges directed towards the node and out-degree is the number of edges pointing away from the node. The normalized form of degree centrality for directed graph is shown below equation 12).

$$C_D^{in}(i) = \frac{k_{in}(i)}{n-1}; \quad C_D^{out}(i) = \frac{k_{out}(i)}{n-1} \tag{13}$$

A node with a higher degree centrality means that the node is in direct contact with many other nodes in the network.

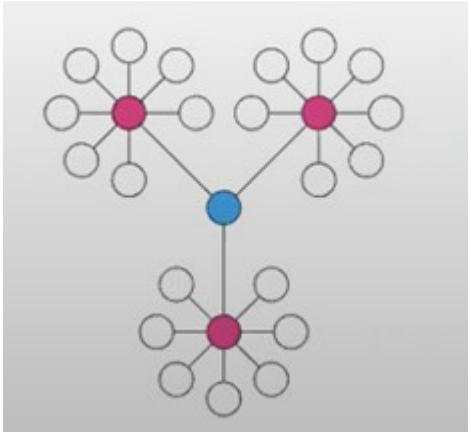


Figure 11: Network displaying nodes with degree and closeness centralities

As seen in the above Figure 11, the pink node has a high degree centrality, and the blue node has a lower degree centrality. A higher degree centrality does not mean that the node is at the center of the network; a node with a higher degree centrality can be anywhere in the network. [7]

1.4.1.2 Closeness Centrality

Closeness centrality is used to know whether the node is at the geometric center of the network or not. In other words, closeness of the node with other nodes in the network. It is calculated as the ratio of the shortest path length from the current node to every other node in the network divided by the total number of nodes (leaving the current node in consideration). The shortest path length is the least number of hopes to reach from current node to desired node in the network.

The general mathematical formula to calculate the closeness centrality C_C is as follows:

$$C_C(i) = \frac{1}{\sum_j d(i,j)} \tag{14}$$

As seen the closeness centrality is the reciprocal of sum of shortest path length d, that means greater the shortest path length smaller is the closeness centrality and vice versa.

The normalized value of closeness centrality C_C^* is used when comparing two networks and is calculated as follows:

$$C_C^*(i) = (n-1)C_C(i) = \frac{n-1}{\sum_j d(i,j)} \tag{15}$$

For directed graphs, the closeness centrality calculation is a bit different as it depends on the direction of path being used. So, in most cases to calculate the closeness centrality the directionality is dropped of the network and compute the same way as undirected graph.

$$C_C(i) = \frac{n-1}{\sum_j d(i,j)} \tag{16}$$

In the case of disconnected networks, there is usage of harmonic centrality which is same as closeness centrality by used for computing closeness centrality for disconnected networks.

$$C_H(i) = \sum_j \frac{1}{d(i,j)} \tag{17}$$

In Figure 11, the node blue is at two steps away from white nodes and one step away from the pink nodes. Hence blue nodes have the highest closeness centrality. This signifies that nodes with high closeness centrality are the geometric center of the network. [7]

1.4.1.3 Betweenness Centrality

Betweenness centrality C_B is used to measure the significance of a node *i* when there is a transfer of information between the networks. It is measured by the number of shortest paths passing through the node. Let $\sigma_{st}$is the total number of shortest paths from node *s* to node *t* and $\sigma_{st}(i)$ is the number of those paths that pass-through *i.*

The mathematical formula gives betweenness centrality is given by equation (18).

$$C_B(i) = \sum_{s \neq t \neq i} \frac{\sigma_{st}(i)}{\sigma_{st}} \tag{18}$$

Normalized value of betweenness centrality C_B^* is given by:

$$C_B^*(i) = \frac{2}{(n-1)(n-2)} C_B(i) = \frac{2}{(n-1)(n-2)} \sum_{s \neq t \neq i} \frac{\sigma_{st}(i)}{\sigma_{st}} \tag{19}$$

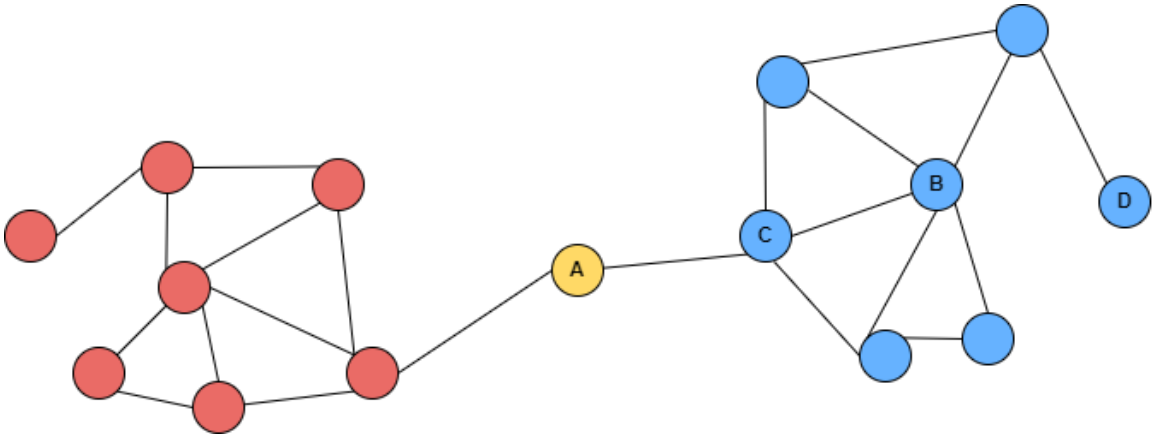


Figure 12: Sample graph explaining betweenness centrality.

High betweenness centrality indicates that nodes lies on many shortest paths. As seen in the above *Figure 12* node A lies in many shortest paths between the nodes in the network and hence is of high importance. [7]

1.4.1.4 Eigenvector Centrality

In simple terms, Eigenvector centrality states that a node is important if it is connected to other nodes which are important. If one makes an analogy with respect to social networks like Facebook, a person who is a friend of a celebrity is considered more important than another user with many non-famous users. The Eigenvector centrality is calculated using matrix calculation to find the *principal eigenvector* using an adjacency matrix. [7]

Mathematically, let's consider a simple case where there is a graph with N nodes and an adjacency matrix A (where A_{ij} = 1 if there is a link from node *i* to node *j*, and 0 otherwise). The eigenvector centrality *x* of node *i* can be defined as:

$$x_i = \frac{1}{\lambda} \sum_{j=1}^{n} A_{ij} x_j \tag{20}$$

Where: the sum is over all nodes *j*, x_j is the centrality of node *j*, λ is a constant (the eigen value). In matrix form, this can be written as:

$$Ax = \lambda x \tag{21}$$

where x is the vector of centralities for each node in the network, A is the network's adjacency matrix, and is the eigenvalue connected to eigenvector x. The centrality scores are the components of the eigenvector of the adjacency matrix A, which is the vector x that satisfies this equation. The greatest eigenvalue's corresponding eigenvector determines the eigenvector centrality scores. The centrality of the associated node in the network is then determined by each component of this eigenvector. [14]

2 HEURISTIC ALGORITHMS

This section discusses theoretical background about heuristics and metaheuristic algorithms. The sub-section related to evolutionary algorithms with general working principles, Particle Swarm Optimization explains the detailed working of algorithm which is later used in the practical part of the thesis.

2.1 Heuristics – meaning and definition.

The word heuristics has dictionary meaning as follows: *a way of solving problems by discovering things yourself and learning from your own experience.* [15]

Heuristic algorithms are used to find quick solutions to a time-consuming problem, otherwise with traditional computational approach needs more time or even impossible to get a solution at all. Heuristics algorithms are used to solve problems which are too complex (both in time and nature) or require large memory usage. Examples of these problems include Hamiltonian-cycle problem, Travelling Salesman problem and so on [16]. These problems are time-consuming in nature and hence need a relatively faster solution. But there are certain trade-offs for working with heuristics like accuracy (what is the threshold level for error), and optimality (reasonably optimal solution over best solution). [17]

Heuristics algorithms are basically problem dependent and change from one problem to another. Meta-heuristics refers to a class of algorithms that can be applied to a broad range of problems, meta-heuristics are high-level optimization algorithms and they do not have specific knowledge about the problem being solved [18].

2.2 Evolutionary algorithms

Evolutionary Algorithms (EA) or evolutionary techniques is sub-field of artificial intelligence inspired by the Darwinian principles of natural selection and Mendel's law of heredity. Even though significant amount of research is done on EA, these algorithms still face the hurdle of No Free Lunch Theorem [19] [20] and this theorem states that there is no single universal algorithm that can be used to solve all the problems. Evolutionary Algorithms (EA) are class of heuristics that are concerned with population of solutions rather than a single solution and this population is evolved over time to get optimal solution for the problem.

The evolutionary algorithms follow following sequence of operations:

Initial population of solutions is generated randomly;

Calculate objective (fitness) value for each solution in the population;

while stopping criterion not met

Select solutions with best cost values for reproduction (parents);

Recombination of parents (crossover);

Mutation the resulting offspring;

Evaluate cost value for the new solutions;

Replace the solutions with lower cost value in the population with the new solutions;

end while;

At the start of the algorithm the population is initialized randomly. The population is usually initialized using standard deviation. However, in some cases the population is initialized with solution or knowledge from previous runs and this initialization is known as *warm start* [21]. During the evolution of population certain individual solutions may cross the boundary of the search space, to overcome this situation boundary strategies are used, these strategies are called *Random Clipping, Reflection, Periodic.* [22] [23] [24] and based on the strategies used individual solution is either modified or replaced with another random solution.

In this thesis the focus is only on one algorithm and especially swarm intelligence algorithm like is Particle Swarm Optimization (PSO) algorithm is selected, due to the fact, that it "moves" above search space, i.e., the process is dynamic and can imitate the dynamical changes in social (complex network) and mainly is modelled by "movement" equation. (i.e., dynamics of PSO-created network can be also described by equation). The PSO also has proven record of usage in various fields like cloud computing [25], routing problems [26] [27], medical diagnosis [28] and many more. The next sub-section introduces the working and implementation of PSO.

2.3 Particle Swarm Optimization Algorithm

In 1995 *Kennedy* and *Eberhart* suggested an algorithm named *Particle Swarm Optimization* (PSO) that is motivated by the behavior of some living being in a swarm, like fishes in school and birds in a flock [29]. Swarm intelligence algorithms show the

phenomenon called collective intelligence in which swarm acts as a whole without having any central control.

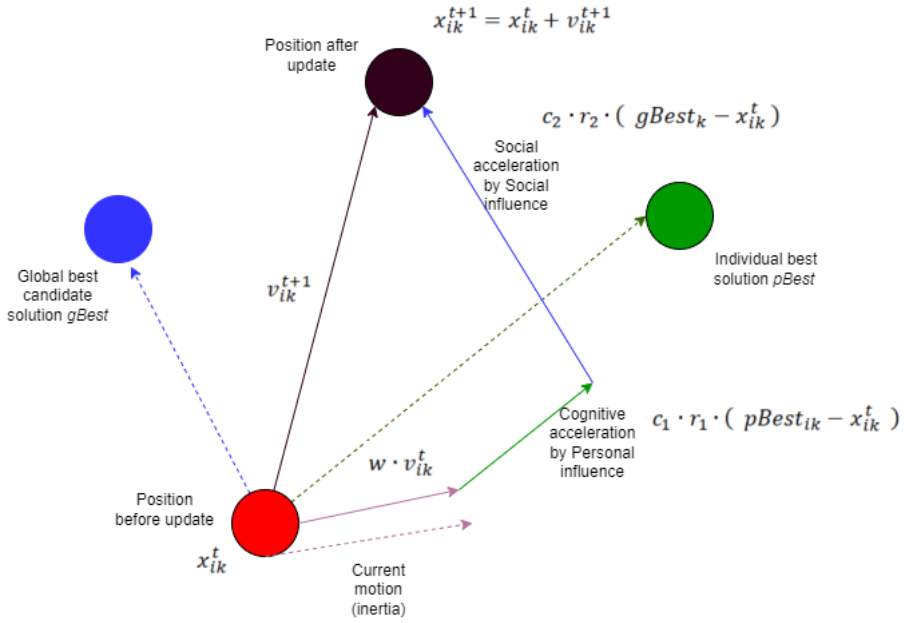


Figure 13: Particle Motion in PSO [30]

The above *Figure 13* shows the particle's motion in PSO. PSO is a population-based stochastic optimization algorithm strategy which means the initial population is maintained and evolved over time to get the best solution. The particle position update is affected by two factors: cognitive influence and social influence. Cognitive influence is due to the particle's personal best (*pBest*) position, and social influence is due to the best-found (*gBest*) solution in the swarm. [30]

The Standard PSO algorithm usually has few input parameters like *NP* (number of individuals in the population), *D* (dimension of each particle in the population), *CF* (cost (objective) function to check the fitness of each particle), *maxIt* (maximum iterations allowed for the algorithm), *w* (inertia which controls the influence of velocity), c_1 (cognitive coefficient) and c_2 (social coefficient).

Each particle is associated with position ***x*** and velocity ***v***. The position of each particle is randomly initialized within the specified boundary. The fitness is calculated for each initialized position. The initial velocity is set to zero for all particles as suggested in [31]. The initial personal best (*pBest*) of each individual is fitness at the initial position. The swarm's initial global best (*gBest*) is calculated from all the personal best values.

Once the initialization is complete, the algorithm is repeated for a certain number of iterations *maxIt*. During each iteration, the velocity of each individual is calculated using the following equation:

$$v_{ik}^{t+1} = w \cdot v_{ik}^{t} + c_1 \cdot r_1 \cdot (\ pBest_{ik} - x_{ik}^{t} \ ) + c_2 \cdot r_2 \cdot (\ gBest_k - x_{ik}^{t} \ ) \tag{22}$$

here r_1 and r_2 are random variables in the range from 0 to 1. Using these random variables ensures that the particle changes its path slightly, searching for more possibilities in the search space. The initial velocity is set to 0 according to authors in [32]

The position of each particle is updated using the equation:

$$x_{ik}^{t+1} = x_{ik}^{t} + v_{ik}^{t+1} \tag{23}$$

where v_{ik}^{t+1} represents the updated velocity of the particle and is calculated using current position of the particle and the updated velocity.

The *Figure 13* gives a graphical representation of the particle moving in the search space. [30]

The pseudo-code for standard PSO is as follows.

Table 1: Pseudo code for PSO algorithm

```
1:   Initialization of particle position
2:   Calculate fitness of each particle
3:   Calculate gBest
4:   while it < max_iterations do
5:        for i = 1 to NP do
6:             calculate v_i
7:             calculate x_i
8:             if fitness(x_i) < fitness(pBest_i)then
9:                       pBest_i = x_i
10:            if fitness(pBest_i) < fitness(gBest)then
11:            gBest = pBest_i
12:            end if
13:          end if
14:       end for
15:  end while
```

General PSO working with simplified flow-chart is shown in *Figure 14*. The algorithm starts with random initialization of the population within the boundary specified for each particle. Calculate fitness for each particle and corresponding *pBest* and *gBest.* Based on the number of iterations the algorithm's inner loop is repeated for maximum number of iterations mentioned initially. For each iteration the particle's velocity, position and *pBest, gBest* are updated. The algorithm ends after executing required number of iterations. [33]

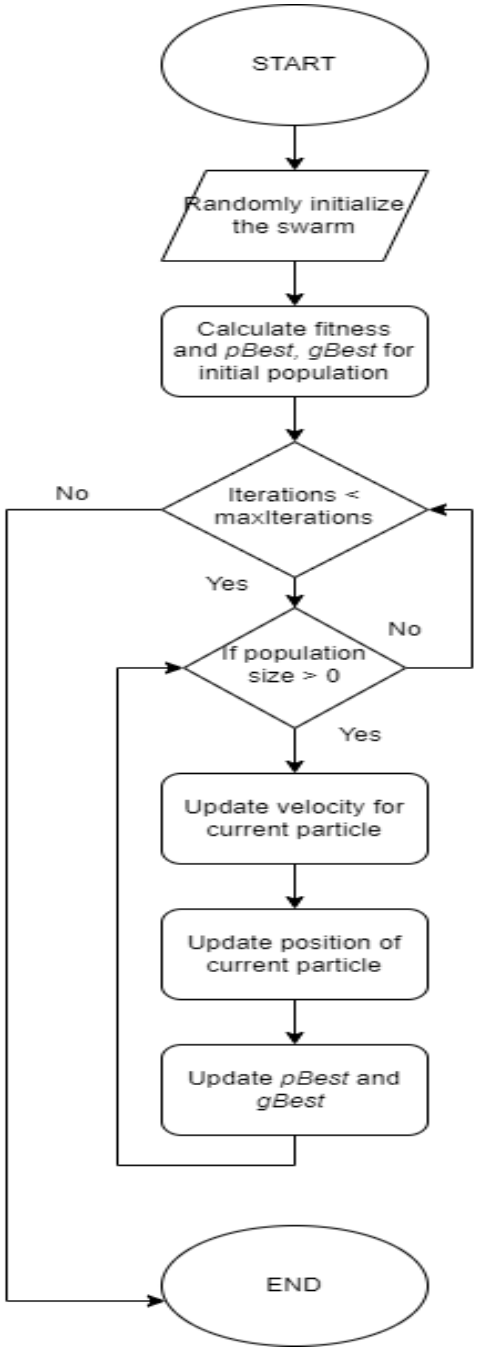


Figure 14: Flow chart for working of PSO [33]

2.4 Hyper-Heuristics

Numerous problem-independent algorithmic frameworks (Heuristics and Metaheuristics) are used for decision and optimization problems. Still, applying the currently available algorithms to newly countered issues or even to new occurrences of an existing problem might take time and effort. Generally, tweaking parameters requires a lengthy process. The parameters are frequently inaccurately defined and prevent a reasonable resolution of the problem (producing acceptable solutions). To tackle specific circumstances, one must often create new algorithms. There have been some recent attempts to automate the process of tuning the parameters by designing algorithms. The fundamental concept is to design an approach that is general and adaptable to minor changes and learns which suitable changes should be made at each phase of solving procedure. Hyper-heuristics are suitable for these types of scenarios. [34]

The term Hyper-heuristics [35] [36] [37] [38] was coined by Peter Cowlingin. The hyper-heuristic approach does not need problem-specific information apart from a bunch of simple-to-build heuristics. The hyper-heuristics can select between low-level heuristics using performance indicators; these performance indicators are independent of the problem domain and hence used by hyper-heuristics to decide whenever a low-level heuristic needs to be called for particular instance in the space. The communication between low-level heuristics and hyper-heuristics uses typical problem-independent interface architecture. Hyper-heuristics can examine the results returned by low-level heuristics used, or to change the current solution itself. The hyper-heuristics may also provide basic data to the low-level heuristics, like duration allowed to run. When hyper-heuristics are called low-level heuristics, it returns important information like solution quality and total CPU execution time. It is to note that hyper-heuristic has limited information about the objective function (just whether it is a minimization or maximization function), the architectural barrier facilitates the information transfer between the low-level and the hyper-heuristic. Domain knowledge is not allowed to cross the barrier. [34]

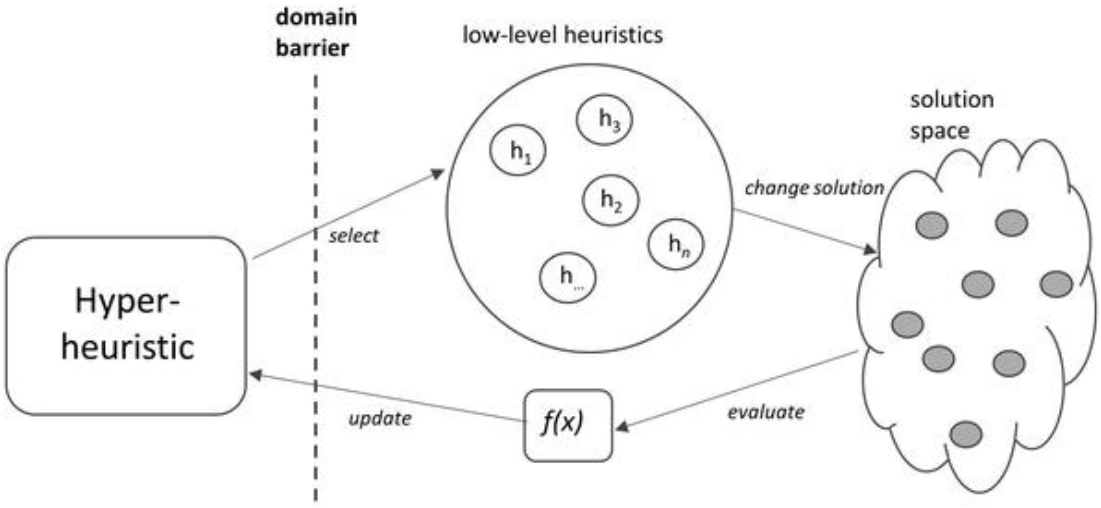


Figure 15: General scheme of how hyper-heuristics work [34]

A typical hyper-heuristic framework is composed of two levels of heuristics algorithms, hyper-heuristic and low-level heuristics, separated by a domain barrier. The hyper-heuristic algorithm monitors the performance of low-level heuristics and selects the appropriate low-level heuristics to either change the current solution itself or get performance metrics. The performance is measured using several factors like the objective function value (increase or decrease) for the given problem domain, CPU computation time, or last-time usage of particular heuristics. Single-point solutions and multi-point solutions are also possible using hyper-heuristics. In a single-point solution case, the initial solution goes through several steps to output the final solution. In multi-point solutions, a few solutions from a population are processed in parallel. The acceptance mechanism, the decision to accept or reject a solution from low-level heuristics, is an essential step in hyper-heuristics. The acceptance mechanism can be deterministic (same all the time) or non-deterministic (sometimes different, like decision-based on time elapsed to compute solution). The hyper-heuristic algorithm iteratively repeats this process of selecting the low-level heuristics from the available set of heuristics using the acceptance mechanism mentioned above. Once a low-level heuristic is selected, it is applied to the solution until the stopping criterion is achieved. As opposed to meta-heuristics (which work on the solution space), hyper-heuristics operates only on the search space and has no information about the problem being solved. Typical scheme for hyper-heuristics work is shown in above *Figure 15*. [34]

Tunning and controlling the parameters of another evolutionary algorithm is also alternative used to define Hyper-heuristic technique [39]. Since PSO is the only algorithm used for this thesis work, the hyper-heuristic technique to tune the parameters of PSO is employed.

2.5 Regression models

This section concerns the theoretical part for the regression model. Two model are considered linear regression model and multiple regression model.

2.5.1 Linear regression

Linear regression is a statistical model that is used to predict or analyze the relationship between two variables i.e., dependent, and independent variable. For linear regression the equation is given by

$$y = m * x + c \tag{24}$$

Where *y* is the dependent variable, *x* is the independent variable, *c* is the intercept on *y-axis, m* is the slope of the regression line.

2.5.2 Multiple Linear Regression

Multiple linear regression is a statistical model that is used to predict or analyze the relationship between two or more independent variables and one dependent variable. The equation for multiple regression is given by.

$$y = c + m1 * x1 + m2 * x2 + \cdots + m_n x_n \tag{25}$$

Where c is the intercept on y-axis, m1 and m2 are the coefficients of the predictor variables x1 and x2 respectively. $m_i$estimates the expected change in the dependent variable y for one-unit change in the corresponding predictor x_i, holding all other predictors constant.

II. ANALYSIS

3 COMPLEX NETWORK CREATION BASED ON REAL DATA

This section is related to the practical part of building a complex network based on real data. The citation network was used for this study, and the data was taken considering citations between the faculty members of the Department of Artificial Intelligence, *Tomas Bata University in Zlín (UTB), FAI.*

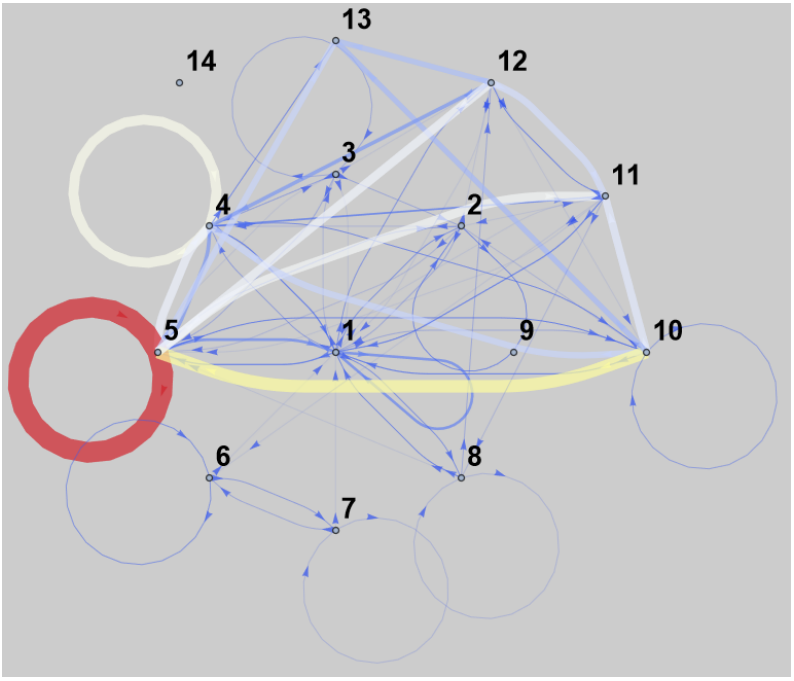


Figure 16: Visualization of the citation network based on real data at the end of 2020.

The citation network in *Figure 16* is drawn using the data collected from the Scopus [40] database. The data is processed to get citation information from 2015-2020. The processed data in form of an excel format can be found in the repository [1]. The tables *Table 2* to Table *5* represent the processed data in adjacent matrix form.

The nodes in the citation network represent the faculty members of the department of informatics and artificial intelligence, FAI, UTB and the edges represent the paper citation between the authors. The thickness of the edge (weight of the edge) represents the number of citation references between the authors or the author citing himself or herself. Citation data for 14 authors was collected. Each table's row indicates an author who referenced another author, while the column represents the cited author. For instance, in *Table 2*, by the

[1] https://shorturl.at/lHR36

end of 2015, the number 3 in the first row means that the author with ID 1 referenced the author with ID 10 three times. *Figure 16* shows the graphical visualization of the citation network towards the end of 2020. The circular loop or self-loops around some of the nodes represents authors who cited themselves in their previous work. The color and thickness of edges represent the weights of the edges, thinner edges indicated less citation between the authors and darker and thicker edges indicated relatively heavy interaction between the authors. There is one singleton node in the graph that represents there was no citation information gathered for that particular author over a specific period of time.

The captured data was visualized using software Mathematica [41] and code is available in the repository[2] . As seen from the *Figure 16* the most self-citation was recorded for the author with node 5 followed by authors with node 4. There was highest citation between the authors nodes 5 with 10, followed by relatively weak interaction between nodes 5 with 12, 5 with 11 and 5 with 4.

Table 2: Data at the end of year 2015

ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	9	2	2	4	4	0	0	3	0	3	0	1	0	0
2	0	2	0	0	0	0	0	0	0	0	0	0	0	0
3	0	1	1	1	1	0	0	0	0	0	0	0	0	0
4	5	0	0	11	5	0	0	0	0	3	0	0	0	0
5	6	0	0	12	18	0	0	0	0	3	0	0	0	0
6	1	0	0	0	0	2	3	0	0	0	0	0	0	0
7	1	0	0	0	0	0	1	0	0	0	0	0	0	0
8	1	0	0	0	0	0	0	0	0	0	0	1	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	1	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3: Data until the year 2016

ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	13	4	2	4	5	0	0	4	0	3	0	2	0	0
2	1	3	0	0	0	0	0	0	0	0	0	0	0	0
3	0	1	2	3	2	0	0	0	0	0	0	0	0	0
4	6	0	0	18	10	0	0	0	0	4	0	0	0	0
5	11	3	1	32	51	0	0	0	0	5	0	0	0	0
6	1	0	0	0	0	4	3	0	0	0	0	0	0	0
7	1	0	0	0	0	2	1	0	0	0	0	0	0	0
8	3	0	0	0	0	0	0	1	0	0	0	2	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	2	1	0	12	19	0	0	0	0	11	0	0	0	0
12	1	0	0	0	0	0	0	1	0	0	0	2	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0

[2] https://shorturl.at/lHR36

Table 4: Data until the year 2017.

ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	14	4	2	4	5	0	0	5	0	3	0	4	0	0
2	1	3	0	0	0	0	0	0	0	0	0	0	0	0
3	0	2	3	4	3	0	0	0	0	0	0	0	0	0
4	7	0	0	23	13	0	0	0	0	5	2	0	0	0
5	12	4	1	32	72	0	0	0	0	6	12	0	0	0
6	1	0	0	0	0	5	4	0	0	0	0	0	0	0
7	1	0	0	0	0	3	2	0	0	0	0	0	0	0
8	4	0	0	0	0	0	0	2	0	0	0	3	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	3	2	0	23	39	0	0	0	0	3	11	0	0	0
12	2	0	0	0	0	0	0	2	0	0	0	5	0	0
13	1	0	0	7	14	0	0	0	0	1	7	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5: Data until the year 2018.

ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	16	5	2	4	6	0	0	5	0	3	0	5	0	0
2	2	4	0	1	1	0	0	0	0	0	0	1	0	0
3	1	3	3	4	3	0	0	0	0	0	0	1	0	0
4	9	0	0	30	20	0	0	0	0	6	8	0	3	0
5	14	4	1	57	51	103	0	0	0	7	37	0	18	0
6	1	0	0	0	0	5	4	0	0	0	0	0	0	0
7	1	0	0	0	0	3	2	0	0	0	0	0	0	0
8	5	0	0	0	0	0	0	2	0	0	0	3	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	5	2	0	35	62	0	0	0	0	4	34	0	17	0
12	4	1	0	0	0	1	0	2	0	0	0	6	0	0
13	1	0	0	17	36	0	0	0	0	1	29	0	17	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6: Data until year 2019.

ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	18	5	2	4	8	0	0	5	0	3	0	5	0	0
2	3	6	0	2	2	0	0	0	0	0	0	1	0	0
3	1	3	3	4	3	0	0	0	0	0	0	1	0	0
4	9	0	0	36	25	0	0	0	0	6	12	1	6	0
5	15	4	1	57	122	0	0	0	0	7	52	0	32	0
6	1	0	0	0	0	5	4	0	0	0	0	0	0	0
7	1	0	0	0	0	3	2	0	0	0	0	0	0	0
8	6	0	0	0	1	0	0	2	0	0	0	3	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	5	2	0	40	76	0	0	0	0	4	46	0	28	0
12	5	1	0	0	1	1	0	2	0	0	0	6	0	0
13	1	0	0	21	49	0	0	0	0	1	41	0	28	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 7: Data until the year 2020.

ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	18	5	2	4	9	0	0	5	0	3	1	5	0	0
2	3	6	0	2	2	0	0	0	0	0	0	1	0	0
3	1	3	3	4	3	0	0	0	0	0	0	1	0	0
4	9	0	0	68	25	0	0	0	0	6	12	0	6	0
5	19	4	1	57	139	0	0	0	0	7	62	0	44	0
6	1	0	0	0	0	5	4	0	0	0	0	0	0	0
7	1	0	0	0	0	3	2	0	0	0	0	0	0	0
8	6	0	0	0	1	0	0	2	0	0	0	3	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	8	2	0	44	85	0	0	0	0	4	53	0	35	0
12	8	1	0	0	1	1	0	2	0	0	0	6	0	0
13	1	0	0	25	59	0	0	0	0	1	47	0	37	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0

4 METHODOLOGY TO REPRESENT SWARM ALGORITHM DYNAMICS AS A COMPLEX NETWORK

This practical section is concerned with transforming the dynamics of the selected evolutionary algorithm (PSO in this case) and representing the captured information as a complex network.

PSO is employed for this experiment with settings $c_1 = c_2 = 1.49445$ and w=0.729. Test function used in this experiment is Schwefel's function. The swarm's population size was selected as $NP = 14$, with maximum iterations $maxIt$ =10,000 and dimension D=5. The resulting complex network created is depicted in *Figure 17* below.

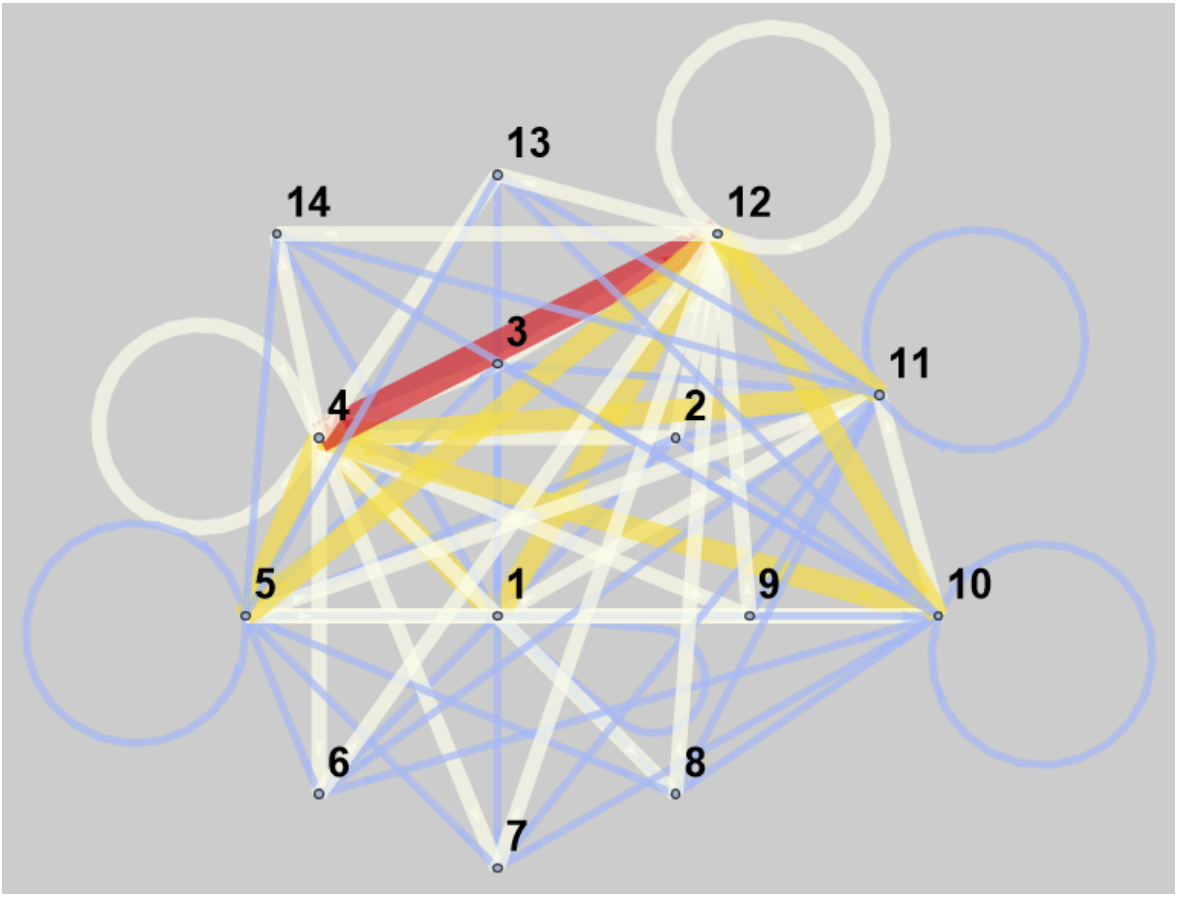


Figure 17: Complex network created by capturing the dynamics of PSO.

It is vital to identify the communication inside the swarm to establish a network related to the inner dynamic of PSO. It is evident (from PSO) that the only communication route within the swarm is shared knowledge of the global optimal solution's position (*gBest*). A few assumptions are used to capture the dynamics of the PSO. Firstly, each particle is represented

by position (or index) in the swarm. Second, the information is captured based on the position (or index) in the swarm and not the actual particle's position.

Based on this knowledge, two methods for capturing the dynamics of the PSO algorithm were considered. The first method captured just the communication between the particle which updates the new *gBest* and the particle which updated the previous *gBest*. Therefore, while creating the complex network, a link is created between the particle that previously updated the *gBest* and the particle that created the new *gBest,* self-loops were ignored. The inspiration for this method of complex network creation is from the article [42]. This method did not consider the weights of the edges between the particles and was unsuitable for this study, hence was ignored.

The second method, with a slightly different approach, was used. Two links were used for communication between the particles. The first link was created between the particle that updated the previous *gBest* and all the particles that updated their *pBest* because of this particle (the particle that updated the *gBest* previously). Furthermore, the second link was created between the particle which updated the new *gBest* and the particle which previously updated the *gBest*. This way of capturing the communication was inspired by the article [43]. This method was preferred for the study as it provided information about the weights of the edges between the nodes of the complex network created.

As observed in the *Figure 17*, there are 14 nodes in the network, and edges connect these nodes. The thickness of the edge signifies the weight of the edge or, in this case, the communication between the particles. The circular loop indicates self-loop means the particle at this position updated its *pBest* based on its previous *gBest* update.

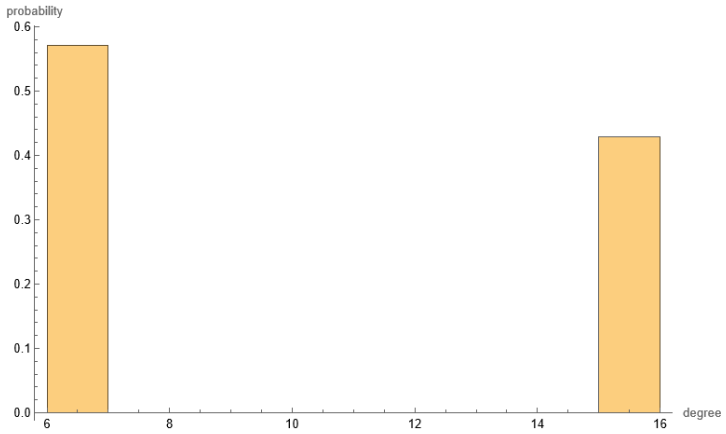


Figure 18: Degree distribution for the complex network based on captured dynamics.

As shown in the Figure 18 the degree distribution of the complex network created from the captured dynamics. The degree distribution indicates that the network does not follow power law. [41]

The complete code for this practical section can be found on the GitHub repository [3], for code considering self-loop for singleton nodes.

Another experimental setup was done for captured data without considering information about authors which cited themselves from the previous work and correspondingly in the PSO, particles which updated their own *pBest*. This work was neglected as the real network had self-citation information and was to use this network. The code can be found in the repository[4].

4.1 Calculating difference between the real and modelled network

The difference between the two networks (modelled network and the real network from data) is calculated using the difference in centralities for the two networks. This is implemented in the form of function in Mathematica with general equation as follows:

$$pts = \left[\left(\frac{DC_r - DC_m}{DC_r}\right)^2 + \left(\frac{BC_r - BC_m}{BC_r}\right)^2 + \left(\frac{CC_r - CC_m}{CC_r}\right)^2 + \cdots + \left(\frac{EC_r - EC_m}{EC_r}\right)^2\right] \cdot 1000 \tag{26}$$

The sub-scripts, r represents centralities for real network and m is for modelled network from the captured dynamics. The ideal or perfect value of *pts* in equation (26) is zero (meaning the modelled network is identical to the real network). The measured centralities in the equation (26) are abbreviations as follows:

- Degree Centrality (DC)
- Closeness Centrality (CC),
- Betweenness Centrality (BC),
- Eigenvector Centrality (EC).

[3] https://shorturl.at/klASZ

[4] https://shorturl.at/klASZ

5 HYPER-HEURISTICS MODEL

This practical section is concerned with the analysis and finding the optimal parameter settings for the EA so that the complex network created from the dynamics of the EA corresponds as close as possible to the actual network. The actual network was created from real data from the previous section 3 – Complex network creation based on real data.

As mentioned in the previous section, the EA used for the experiment to capture the dynamics is PSO. The important parameters that can be optimized to improve the performance of PSO are c_1 (cognitive component), c_2 (social component), and w (inertia), and test functions.

Two approaches were used to optimize the parameters of PSO:

- Method 1: Using another PSO to optimize the parameters for the PSO generating the dynamics.
- Method 2: Local search was added to the method 1 to improve performance.

5.1 Using metaheuristics to optimize the parameters for the PSO generating the dynamics of CN.

In this experiment, another PSO (outerPSO) was employed to find the optimal parameters for the PSO (innerPSO) generating the dynamics for Complex Network. The inspiration for this method of optimizing the parameters of PSO was taken from [44] .

The outerPSO has a population of particles (*c1, c2,* and *w*) that act as input to the innerPSO. These three parameters act as three dimensions of a particle for the outerPSO. So innerPSO uses each set of parameters from outerPSO and generates a random population, captures the dynamics, and calculates the difference with the real network.

The flow chart below shows the working of outerPSO and innerPSO *Figure 19*.

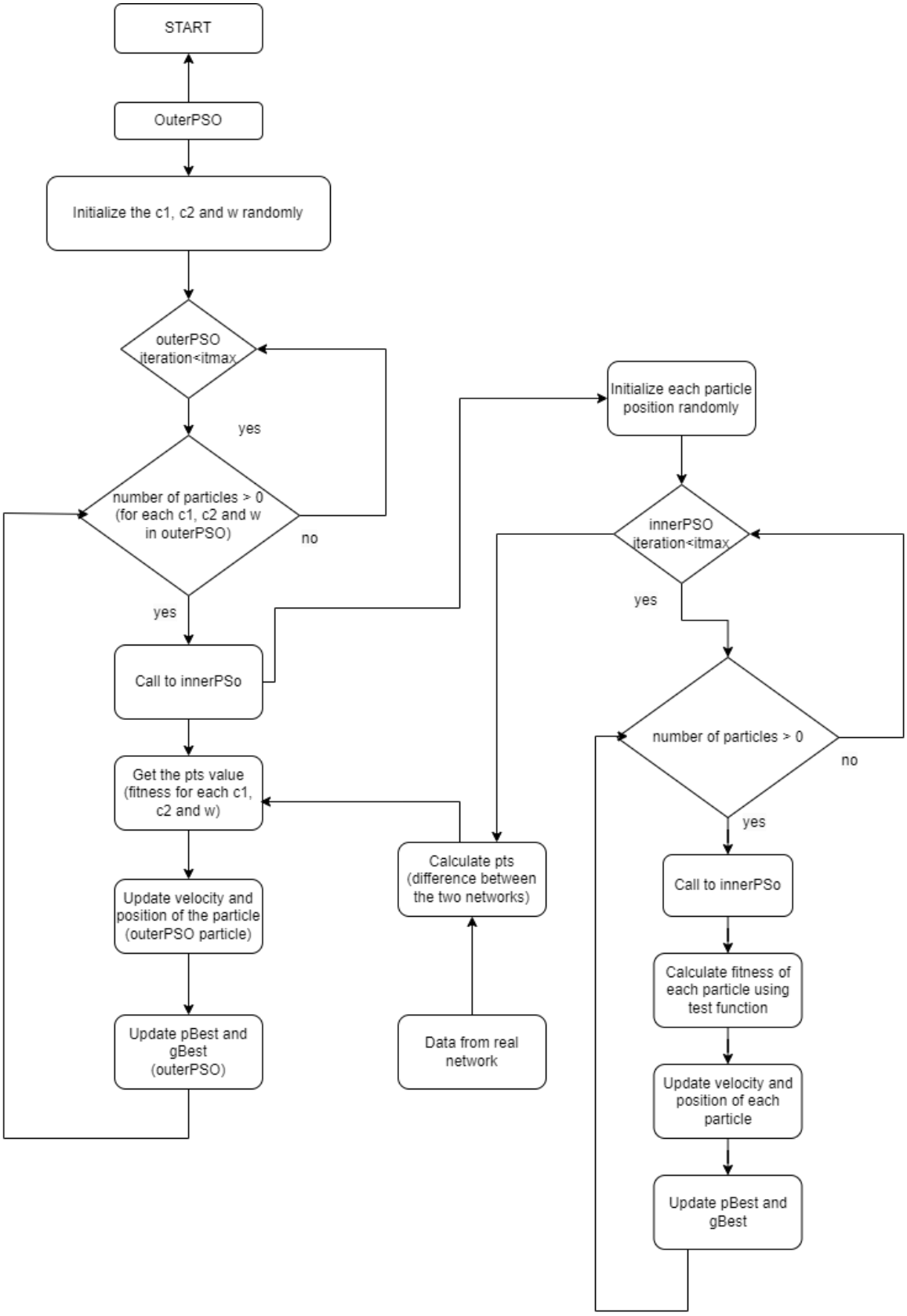


Figure 19: Optimizing the parameters of PSO using another PSO

The code for the experiment mentioned in *Figure 19* was written in Mathematica and is available in the repository[5]. The workflow shows that a random set of population is generated for the outerPSO, this population contains only the three parameters (c_1, c_2, *and w*). For each combination of c_1, c_2, *and w,* a call to the innerPSO is made. The innerPSO takes these parameters, generates random population, and performs further functions of PSO to update the population. At the end of iterations for innerPSO, the dynamics of the innerPSO are captured, and the difference from the real network is calculated. This difference between the real network and network created from the dynamics acts as a cost function for the outerPSO, which further evolves its population (c_1, c_2, and *w*) to achieve minimum difference between the two networks.

The cost function for the innerPSO is initially used as Schwefel's Function. And the cost function for the outerPSO is the difference between the two networks.

The experiment was done with the following details, the same details can also be found in the code with the variable *outputFromOuterPSO*. The experiment was repeated several times to make comparison and the results are present in the repository as mentioned above.

For outerPSO:

- Control parameters: $c_1 = c_2 = 1.49445$, *w=0.729,*
- Population dimension: *D* = 3 (since only three parameters c_1, c_2, and *w*),
- Population size: *NP=30,*
- Max iterations: *iterationsForOuterPSO*= 1000,
- Cost function: difference between the real network and the modelled network generated from the dynamics of PSO.

For innerPSO:

- Control parameters: c_1, c_2, and *w, comes from* outerPSO,
- Population dimension: *D* =5,
- Population size: *NP=14,*
- Max iterations: *iterationsForInnerPSO*= 50,

[5] https://shorturl.at/efmAC

- Starting Cost function: *f*= Schwefel's function.

With further improvements to method 1, logic to find the best test function for the innerPSO was implemented. The innerPSO is employed on seven different test functions and their results are compared to find the test function which produced the minimum difference between the two networks. These test functions include *Rosenbrock's Saddle, Sphere, 3rd De Jong, Schwefel's, Rastrigin, Ackley's 1,* and *Ackley's 2*.

5.1.1 Results for method 1

This sub-section is concerned with the results for method 1 mentioned in section 5.1 - Using another PSO to optimize the parameters for the PSO generating the dynamics of CN.

The setting for the experiment is already explained for both innerPSO and outerPSO above. The results for method 1 aiming to find best test function and parameter setup for each run are both summarized in the *Table 8* below. This table shows the best obtained results from seven independent runs. The results along with the implementation code are available in the repository[6].

Table 8: Results for optimizing the parameters of PSO using method 1 along with test function selection.

Experiment serial number-output File Name	*C1*	*C2*	*W*	Test function	*Pts* (difference between real and modelled network)
1, output.log	1.34	1.42	0.114	3rd De Jong	1114
2, output1.log	1.37	1.13	0.14	3rd De Jong	3764.6
3, output2.log	1.24	1.30	1.58	Rosenbrock's Saddle	1418.32
4, output3.log	1.440	1.47	0.75	Rastrigin	4126.38
5, output4.log	1.216	1.022	0.32	Ackley's 2	4098.87
6, output5.log	1.266	1.165	0.378	3rd De Jong	3314.0
7, output6.log	1.21	1.21	1.23	Sphere	1152.73

[6] https://shorturl.at/tFJMS

The *Table 8* displays optimal values of c_1, c_2 and w, which produced minimum value of *pts* during the respective experiment run. As evident from the results table function *3rd De Jong* produced the minimum value for *pts and* was preferred in three out of seven runs. The value of *pts* showed large variations with small changes in c_1, c_2 and w. Due to the extreme time complexity of hyper-heuristic approach, only seven independent runs were carried out.

5.2 Local search addition to improve the performance of outerPSO.

A local search was added to improve the performance of the algorithm mentioned in *Figure 19*. The flow chart for the algorithm to find optimal parameters for PSO along with local search algorithm addition is shown in the *Figure 20* below. The inspiration for this method was taken from [45].

With the addition of local search, the algorithm now has the possibility to search for better solutions in a region with specified radius. Another change in the previous algorithm is the usage of *warm start* for initialization of population for the inner PSO. This means the population is initialized just once in the outer PSO and the inner PSO uses the same population.

The results for algorithm with outer PSO optimizing the parameters for inner PSO with few sample runs can be found on the repository[7] and the algorithm with local and warm start can be found in the same code repository as both methods are used together.

For outerPSO:

- Control parameters: $c_1 = c_2 = 1.49445$, *w=0.729,*
- Population dimension: *D* = 3 (since only three parameters c_1, c_2, and *w*),
- Population size: *NP=30,*
- Max iterations: *iterationsForOuterPSO*= 1000,
- Cost function: difference between the real network and the modelled network generated from the dynamics of PSO.
- Local search iterations: 50

[7] https://shorturl.at/tyJLZ

For innerPSO:

- Control parameters: c_1, c_2, and *w, comes from* outerPSO,
- Population dimension: *D* =5,
- Population size: *NP=14,*
- Max iterations: *iterationsForInnerPSO*= 100,
- Starting Cost function: f = Schwefel's function.

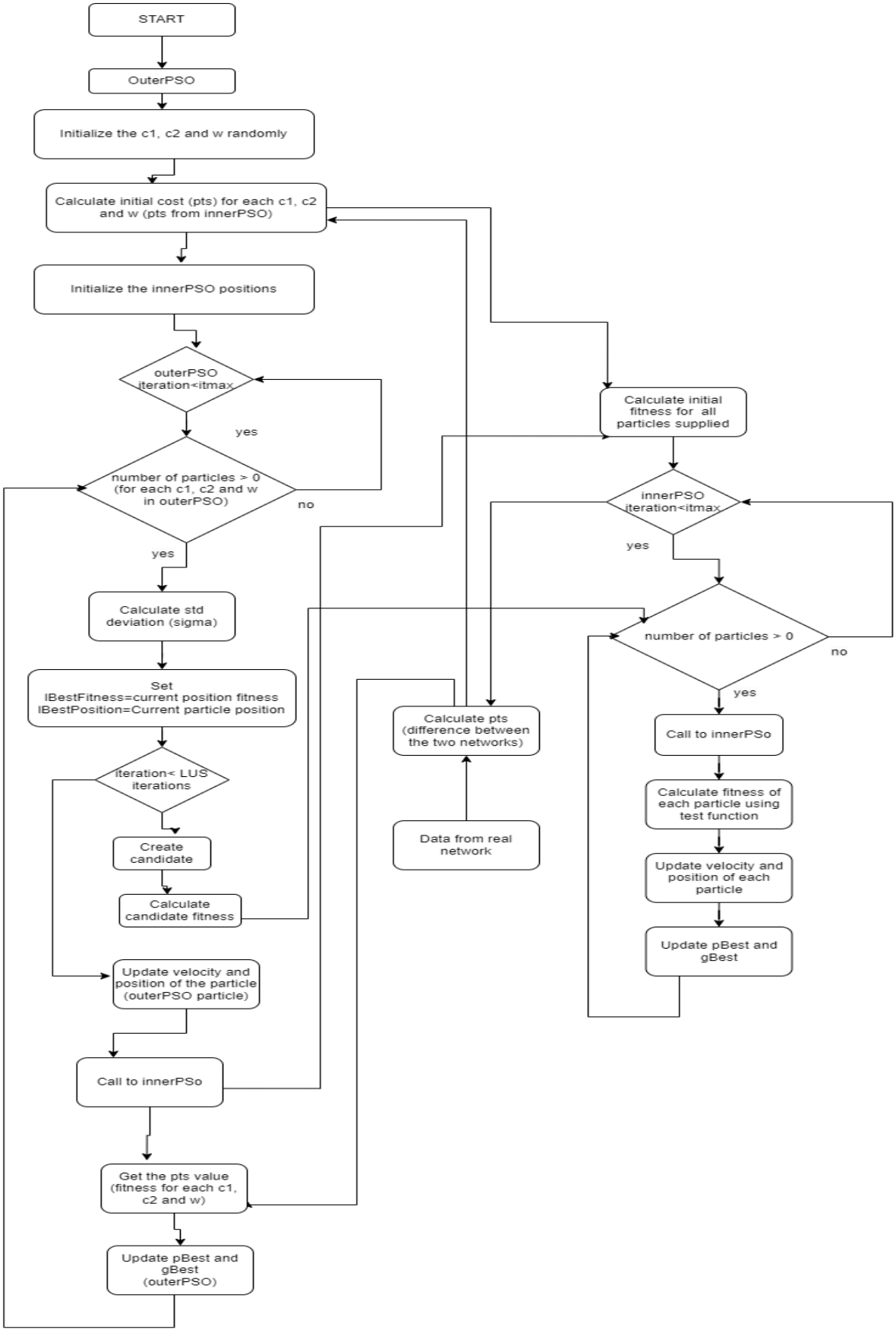


Figure 20: Optimizing the parameters of PSO by using another PSO along with Local Search algorithm.

5.2.1 Results for method 2

This sub-section is concerned with the results for method 2 done in section 5.2 – Local search addition to improve the performance of outerPSO.

Table 9: Results for optimizing the parameters of PSO using method 2

Experiment number – output file name	C1	C2	W	*pts* (difference between real and modelled network)
1, outputA1.log	1.224	1.38	0.49	1231
2, outputA3.log	1.545	1.027	0.568	1099
3, outputA4.log	1.49	1.962	0.822	1113.9
4, outputA5.log	1.60	1.619	0.462	1141
5, outputA6.log	1.69	1.52	0.824	1113

The results are displayed in *Table 9* for method 2. The table displays optimal values of c_1, c_2 and w which produced minimum value of *pts* during the respective experiment run (total five independent runs). As shown from the results from *Table 8* and *Table 9*, a conclusion can be made that method 2 with local search and warm start produced a relatively better result than the experiment with method 1. The value of *pts* was relatively stable in method 2 results than in method 1 results. Since the experiments were run with small number of iterations (because of high time complexity) with further increasing number of iterations better results are expected.

6 PREDICTING THE DYNAMICS OF A COMPLEX NETWORK

This practical section is concerned with predicting the dynamics of modelled network captured from the PSO. The methodology to capture the dynamics of PSO was done in section 4, and network created from real data was done in section 3. In section 3 real data was taken from the Scopus database with citations between the faculty members of AI department at FAI, UTB.

Table 10: Data until year 2019.

ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	18	5	2	4	8	0	0	5	0	3	0	5	0	0
2	3	6	0	2	2	0	0	0	0	0	0	1	0	0
3	1	3	3	4	3	0	0	0	0	0	0	1	0	0
4	9	0	0	36	25	0	0	0	0	6	12	1	6	0
5	15	4	1	57	122	0	0	0	0	7	52	0	32	0
6	1	0	0	0	0	5	4	0	0	0	0	0	0	0
7	1	0	0	0	0	3	2	0	0	0	0	0	0	0
8	6	0	0	0	1	0	0	2	0	0	0	3	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	5	2	0	40	76	0	0	0	0	4	46	0	28	0
12	5	1	0	0	1	1	0	2	0	0	0	6	0	0
13	1	0	0	21	49	0	0	0	0	1	41	0	28	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 11: Data until the year 2020.

ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	18	5	2	4	9	0	0	5	0	3	1	5	0	0
2	3	6	0	2	2	0	0	0	0	0	0	1	0	0
3	1	3	3	4	3	0	0	0	0	0	0	1	0	0
4	9	0	0	68	25	0	0	0	0	6	12	0	6	0
5	19	4	1	57	139	0	0	0	0	7	62	0	44	0
6	1	0	0	0	0	5	4	0	0	0	0	0	0	0
7	1	0	0	0	0	3	2	0	0	0	0	0	0	0
8	6	0	0	0	1	0	0	2	0	0	0	3	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	8	2	0	44	85	0	0	0	0	4	53	0	35	0
12	8	1	0	0	1	1	0	2	0	0	0	6	0	0
13	1	0	0	25	59	0	0	0	0	1	47	0	37	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0

For example, if we consider the data from section 3 until end of 2019 as shown in above *Table 10*. This table represents an adjacency matrix with weights representing the citation between the authors. So, predicting the dynamics of the complex network is nothing but predicting the weights between the nodes and hence *pts* value (equation (26)) of the modelled (captured) network is calculate using the centralities of the nodes.

Three separate methods were employed for prediction.

1. Using PSO-based regression.
2. Using Linear regression technique.
3. Using multiple linear regression techniques.

Each method is mentioned in detail in respective sections 6.1 – 6.3 with final evaluation in chapter 6.4.

6.1 Using PSO based regression.

This method of prediction uses the algorithm mentioned in section 5.2 as the baseline and continues to make predictions using PSO-based regression using just the control parameters from the outerPSO i.e., c_1, c_2 and w, then captures the dynamics from the innerPSO and calculates the difference between the real network and the network modelled using the dynamics of innerPSO which is *pts*.

Firstly, the PSO with tuned parameters is used to learn on data until 2019 mentioned in *Table 6* , the difference between the 2020 real data in *Table 7* and modelled network from the PSO (for 2020 prediction) was found to be 3713.9. The predicted network for this experiment is as follows:

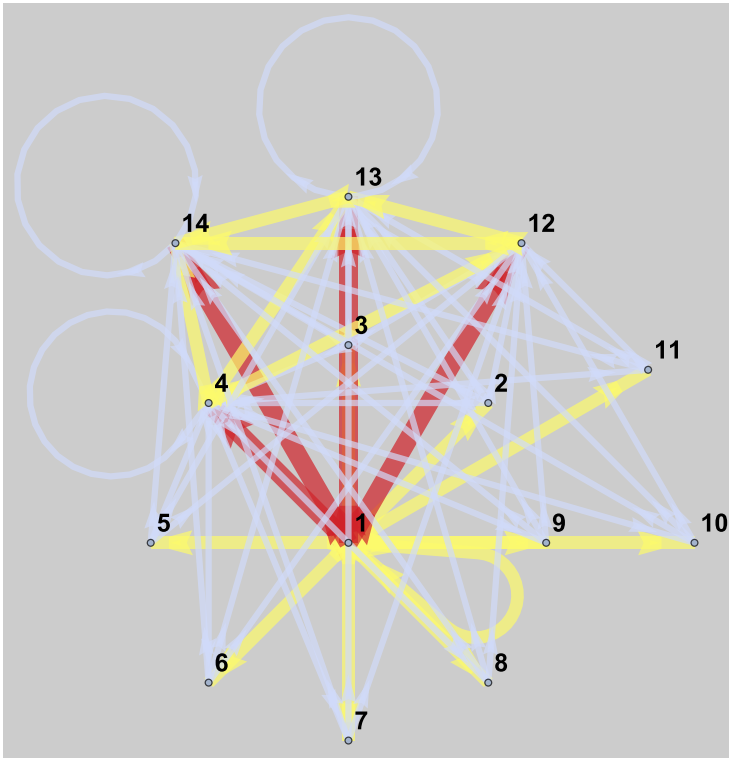


Figure 21: Result for prediction using PSO-based regression.

The code for this is available in the repository[8]. The results for this experiment are measured using the following:

[8] https://shorturl.at/mJKP3

6.2 Using Linear regression technique.

In this experiment a linear regression technique is employed to predict the dynamics of the modelled network. In this method the adjacency matrix data was used as mentioned in section 3 until year 2019 as test data and train the linear model. And the data consists of IDs of the authors, year of citation and the number of citations.

$$\{ID1, ID2, year, number\ of\ citations\} \tag{27}$$

The dataset for test is presented in **Error! Reference source not found.** where the ID r epresents the id for nodes in the network. Therefore, from equation (27) author with id 1 citing author with 2 in the year 2019 with number of citations as 5. The corresponding dataset for this example is {1, 2, 2019, 5}

The model was trained and tested on data from section 3 until the year 2020, which is the following year for training data. The resulting model resembles the real model as shown in Figure 22 below.

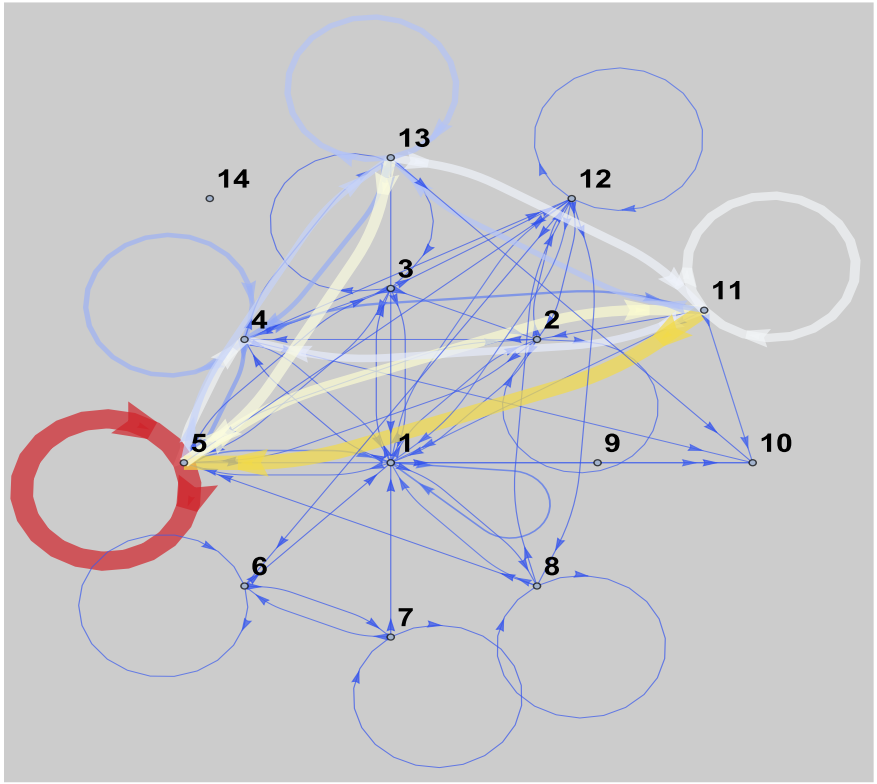


Figure 22: Predicted model using linear regression.

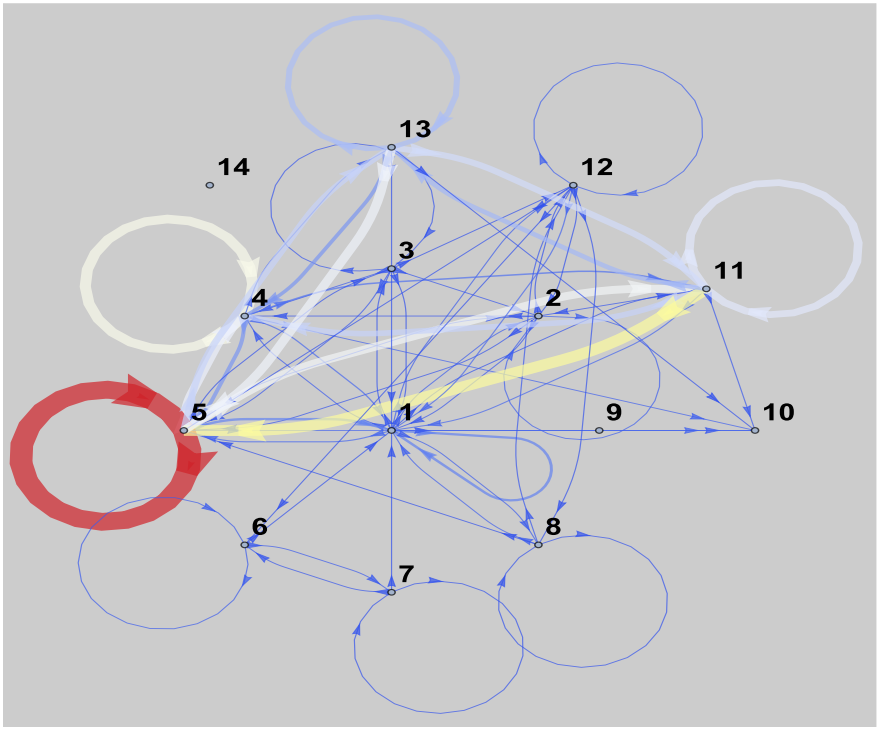


Figure 23: Real data network for comparison with predicted network mentioned above.

The difference between the networks is calculated using the pts equation (26) and is calculated as 997.25 which is closest among all the other prediction models used.

The code for this linear regression model in available on the repository[9].

6.3 Using multiple linear regression technique

In this section of the experiment multiple linear regression was employed. The dataset used for this network is the same as in the section 6.2 for testing and learning. The structure of the dataset is in equation (27). In this experiment there is python code used to create a multiple linear regression model because of its ease of use and libraries available. Code for the created model is available in the repository [10]. Predicted network and the difference from the real network is as shown *Figure 24* below.

[9] https://shorturl.at/DOR04

[10] https://shorturl.at/hs259

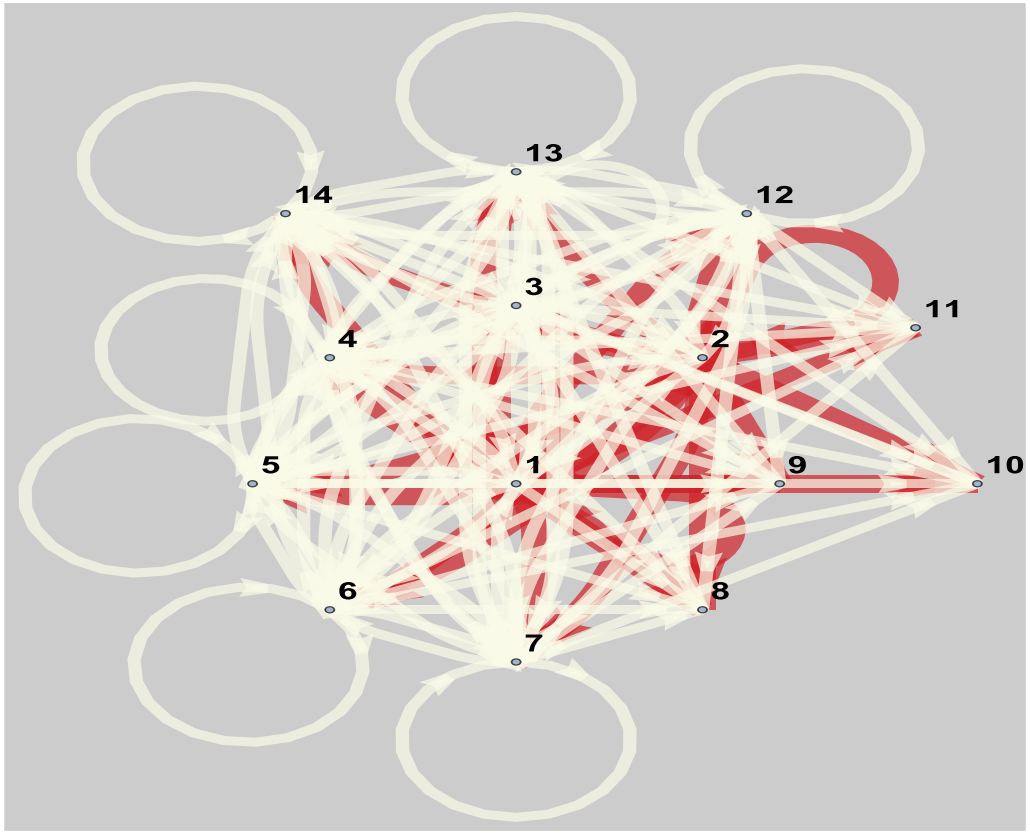


Figure 24: Model created for dynamics using multiple regression model.

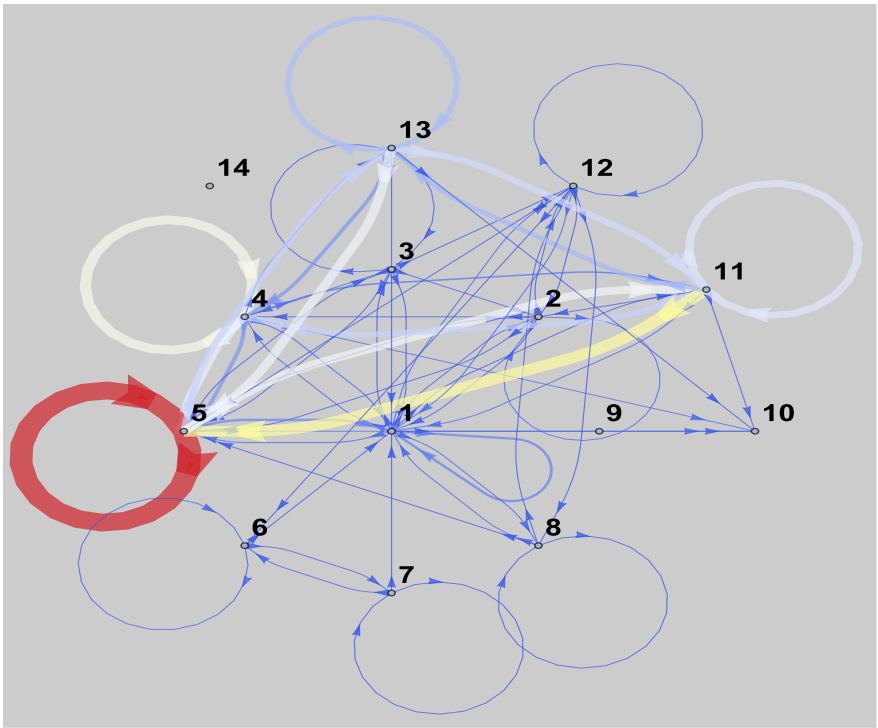


Figure 25: Real network for comparison with above mentioned network using multiple regression.

6.4 Results

This section is related to the results obtained from the experiments for section 6 – Predicting the dynamics of modelled network based on created model.

The experiment in section 6.1 i.e., using the PSO-based regression, the *pts* value of 3713.9 as the difference the real network and the one using the modelled data. This technique should results better than experiment involving multiple regression discussed in section 6.2.

Table 12: Results for predicting the dynamics of modelled network for sections 6.2 and 6.3.

Experiment section #	Algorithm/Model	*Pts* (difference between real and modelled network)
6.1	Hyper heuristic PSO model	3713.9
6.2	Linear regression	997.25
6.3	Multiple linear regression	26864.4

As shown from the *Table 12* the results from the experiments convey that experiment in section 6.2 was closest in predicting the dynamics of the modelled network. The experiment in section 6.3 produced the worst results. The result for hyper-heuristic approach seems to be promising (compared to the multiple linear regression) and surely open for further future investigations. The better results may be achieved by redesigning the workflow, parallelization and speeding up calculation towards higher effectiveness of whole methodology. It is necessary to mention that only canonical PSO was selected, out of plethora of available swarm algorithms.

CONCLUSION

The thesis initiates with theoretical study of network basics, history behind network and graph theory, network models which serves as model for comparison, important terminology related to network with associated equations, characteristics of complex networks and node centralities which helps us understand behavior of nodes in the network. Then literature study about heuristics was done, this includes general study about heuristics, evolutionary algorithms, and the basic principles on which evolutionary algorithms work and some population clipping and reflection concepts while building the algorithm were considered. The evolutionary algorithm, specifically PSO, was studied in detail with its working and implementation. Some regression models like linear regression and multiple regression were also studied.

The practical part was done by implementing most of the concepts learned in the theoretical section. In section 3 of the practical part, a citation network was created with real data collected from the online database. This network created in section 3 of the practical part was used as reference for further experiments. This was followed by implementation of PSO algorithm in Mathematica. In section 4 of the practical part a methodology was implemented to capture the dynamics from the PSO algorithm and transform the dynamics into a network. In section 5, the hyper-heuristics strategy was implemented which used one PSO to optimize the parameters for another PSO. Section 5 also used some variation like selecting the optimal function, local search algorithm and warm start. These variations produced a better result for parameter optimization. The last part of the practical section dealing with predicting the dynamics of the EA (PSO) was implemented. The prediction was done using three techniques, one technique was using the control parameters from the outer PSO as independent variables and difference between the networks as dependent variable. The second technique was implemented using linear regression and the last technique of multiple linear regression was implemented.

During the work on this thesis topic, a better understanding of the evolutionary algorithms and complex network was achieved. The wide area of usage of complex networks and their importance in understanding evolutionary algorithms was realized.

BIBLIOGRAPHY

[1] YONGAN ZHANG, Yiyuan Zhou. Bibliometrics Analysis of Complex Networks Research, July 2017. Available from: https://www.semanticscholar.org/paper/Bibliometrics-Analysis-of-Complex-Networks-Research-Zhang-Zhou/ec59e27fad9d291004e8d8b142131fde7547da3a.

[2] NEWMAN, MEJ. The Structure and Function of Complex Networks.

[3] *Cambridge Dictionary.* Available from: https://dictionary.cambridge.org/dictionary/english/network.

[4] ESTRADA, Ernesto. *The Structure of Complex Networks: Theory and Applications.* Oxford University Press, 2012.

[5] SACHS, Horst; STIEBITZ, Michaeland WILSON, Robin J. An Historical Note: Euler's Königsberg Letters. *Journal of Graph Theory*, 1988, vol. 12, no. 1. pp. 133-139.

[6] *Seven Bridges of Königsberg. (2023, March 28). in Wikipedia.* Available from:https://en.wikipedia.org/wiki/Seven_Bridges_of_K%C3%B6nigsberg.

[7] GOLBECK, Jennifer. Analyzing the Social Web. GOLBECK, Jennifer ed., Boston: Morgan Kaufmann, 2013. *Chapter 2 - Nodes, Edges, and Network Measures*, pp. 9-23. Available from https://www.sciencedirect.com/science/article/pii/B978012405531500002X. ISBN 9780124055315.

[8] ERDŐS, Paul; and RÉNYI, Alfréd. On the Evolution of Random Graphs. *Publ.Math.Inst.Hung.Acad.Sci*, 1960, vol. 5, no. 1. pp. 17-60.

[9] BARABASI, Albert -. L. *Random Networks, Small World Network, Scale Free Network. . Degree, Average Degree and Degree Distribution.* Available from: http://networksciencebook.com/.

[10] SHIHUA ZHANG, Xue-Mei Ning, Xiang-Sun Zhang. Graph Kernels, Hierarchical Clustering, and Network Community Structure: Experiments and Comparative Analysis, 2007. Available from: https://www.semanticscholar.org/paper/Graph-kernels%2C-hierarchical-clustering%2C-and-network-Zhang-Ning/d01d5a1c3444fda99be781914202639d237bacef.

[11] ZHONG, Lin-Feng, et al. Identifying the Influential Nodes Via Eigen-Centrality from the Differences and Similarities of Structure. *Physica A: Statistical Mechanics and its Applications*, 2018, vol. 510. pp. 77-82. Available from https://www.sciencedirect.com/science/article/pii/S0378437118308392. ISSN 0378-4371.

[12] *Cambridge Dictionary - Heuristics.* Available from:https://dictionary.cambridge.org/dictionary/english/heuristics.

[13] MICHALEWICZ, Zbigniew; and FOGEL, David B. *How to Solve it: Modern Heuristics.* Springer Science & Business Media, 2013.

[14] KOKASH, Natallia. An Introduction to Heuristic Algorithms. *Department of Informatics and Telecommunications*, 2005. pp. 1-8.

[15] VIJENDRA KUMAR, S. M. Yadav. A State-of-the-Art Review of Heuristic and Metaheuristic Optimization Techniques for the Management of Water Resources., 1 April 2022. Available from https://doi.org/10.2166/ws.2022.010.

[16] GRIFFITHS, Evan J.; and ORPONEN, Pekka. Optimization, Block Designs and no Free Lunch Theorems. *Information Processing Letters*, 2005, vol. 94, no. 2. pp. 55-61. Available from https://www.sciencedirect.com/science/article/pii/S0020019004003837. ISSN 0020-0190.

[18] SERVICE, Travis C. A no Free Lunch Theorem for Multi-Objective Optimization. *Information Processing Letters*, 2010, vol. 110, no. 21. pp. 917-923. Available from https://www.sciencedirect.com/science/article/pii/S0020019010002449. ISSN 0020-0190.

[19] FORGET, Nicolas; GADEGAARD, Sune Lauthand NIELSEN, Lars Relund. Warm-Starting Lower Bound Set Computations for Branch-and-Bound Algorithms for Multi Objective Integer Linear Programs. *European Journal of Operational Research*, 2022, vol. 302, no. 3. pp. 909-924. Available from https://www.sciencedirect.com/science/article/pii/S0377221722000868. ISSN 0377-2217.

[20] XU, Shenheng; and RAHMAT-SAMII, Yahya. Boundary Conditions in Particle Swarm Optimization Revisited. *IEEE Transactions on Antennas and Propagation*, 2007, vol. 55, no. 3. pp. 760-765.

[21] HELWIG, Sabine; BRANKE, Juergenand MOSTAGHIM, Sanaz. Experimental Analysis of Bound Handling Techniques in Particle Swarm Optimization. *IEEE Transactions on Evolutionary Computation*, 2012, vol. 17, no. 2. pp. 259-271.

[22] ZHANG, Wen-Jun; XIE, Xiao-Fengand BI, De-Chun. *Handling Boundary Constraints for Numerical Optimization by Particle Swarm Flying in Periodic Search Space.* IEEE, 2004.

[23] MASDARI, Mohammad, et al. A Survey of PSO-Based Scheduling Algorithms in Cloud Computing. *Journal of Network and Systems Management*, 2017, vol. 25, no. 1. pp. 122-158.

[24] Loau Tawfak Al-BahraniJagdish Chandra Patra. Orthogonal PSO Algorithm for Economic Dispatch of Thermal Generating Units Under various Power Constraints in Smart Power Grid, May, 2017. Available from https://www.researchgate.net/publication/316655843_Orthogonal_PSO_Algorithm_for_Economic_Dispatch_of_Thermal_Generating_Units_Under_Various_Power_Constraints_in_Smart_Power_Grid.

[25] BHASKAR KANNA, Sn Singh. Towards Reactive Power Dispatch within a Wind Farm using Hybrid PSO, July, 2015. Available from

https://www.researchgate.net/publication/272200684_Towards_reactive_power_dispatch_within_a_wind_farm_using_hybrid_PSO.

[26] SRISUKKHAM, Worawut, et al. Intelligent Leukaemia Diagnosis with Bare-Bones PSO Based Feature Optimization. *Applied Soft Computing*, 2017, vol. 56. pp. 405-419. Available from https://www.sciencedirect.com/science/article/pii/S1568494617301485. ISSN 1568-4946.

[27] J. Kennedy; and R. Eberhart. *Particle Swarm Optimization.* , 1995. ISBN NULL-.

[28] Sanjay Saini, Dayang Rohaya Bt Awang Rambli, M. Nordin B. Zakaria, Suziah Bt Sulaiman. A Review on Particle Swarm Optimization Algorithm and its Variants to Human Motion Tracking, 2014. Available from https://doi.org/10.1155/2014/704861.

[29] EBERHART, Russell; and KENNEDY, James. *A New Optimizer using Particle Swarm Theory.* Ieee, 1995.

[30] EBERHART, Russell; and KENNEDY, James. *A New Optimizer using Particle Swarm Theory.* Ieee, 1995.

[31] BHANDARI, Ashish Kumar, et al. Performance Study of Evolutionary Algorithm for Different Wavelet Filters for Satellite Image Denoising using Sub-Band Adaptive Threshold. *Journal of Experimental & Theoretical Artificial Intelligence*, 2016, vol. 28, no. 1-2. pp. 71-95. Available from https://doi.org/10.1080/0952813X.2015.1020518. ISSN 0952-813X.

[32] LORENTE, Javier D. S. *Heuristics and Hyper-Heuristics: Principles and Applications.* BoD–Books on Demand, 2017.

[33] BURKE, Edmund, et al. Hyper-Heuristics: An Emerging Direction in Modern Search Technology. *Handbook of Metaheuristics*, 2003. pp. 457-474.

[34] BURKE, Edmund K., et al. Hyper-Heuristics: A Survey of the State of the Art. *Journal of the Operational Research Society*, 2013, vol. 64. pp. 1695-1724.

[35] BURKE, Edmund K., et al. A Classification of Hyper-Heuristic Approaches: Revisited. *Handbook of Metaheuristics*, 2019. pp. 453-477.

[36] DRAKE, John H., et al. Recent Advances in Selection Hyper-Heuristics. *European Journal of Operational Research*, 2020, vol. 285, no. 2. pp. 405-428.

[37] DEUGO, Dwight; and FERGUSON, Darrell. *Evolution to the Xtreme: Evolving Evolutionary Strategies using a Meta-Level Approach.* IEEE, 2004.

[38] *Scopus.* Available from:<https://shorturl.at/eyzJP>.

[39] *Wolfram Mathematica.* Available from: https://www.wolfram.com/mathematica/.

[40] Michal Pluhacek, Roman Šenkeřík, Adam Viktorin & Tomas Kadavy. *Complex Networks in Particle Swarm.* , November, 2017.

[41] Ivan Zelinka & Roman Šenkeřík. *On Relation between Swarm and Evolutionary Dynamics and Complex Networks.* , June, 2019.

[42] S. Doctor; G. K. Venayagamoorthyand V. G. Gudise. *Optimal PSO for Collective Robotic Search Applications.* , 2004. ISBN NULL-.

[43] M. E. H. Pedersen. *Tuning & Simplifying Heuristical Optimization.* , 2010.

LIST OF ABBREVIATIONS

PSO – Particle Swarm Optimization algorithm

EA – Evolutionary Algorithm

CN – Complex Network

LIST OF FIGURES

LIST OF TABLES