Multiprotocol Label Switching Network Optimization by Metaheuristic Algorithms

By

Mohsin Masood

A thesis submitted for the degree of

Doctor of Philosophy

Department of Electronic and Electrical Engineering

University of Strathclyde

Declaration

This thesis is the result of the author's original research. It has been composed by the author and has not been previously submitted for examination which has led to the award of a degree.

Signed: Date:

Examination Committee:

Dr. Helena Gleskova, Convener

Dr. Vladimir Stankovic, Internal Examiner

Prof. Milan Dado, External Examiner

Dedication

To my beloved wife, Rabia Khan, my two boys and daughter: Rayyan, Hamza, and Pareezay, and my mum; Tasneeem Kausar.

And in memory of my Dad: Mohammad Masood Saqiq Awan (Late)

Acknowledgements

First and foremost I would like to thank the Almighty ALLAH, the giver and sustainer of life for the grace to complete my PhD thesis.

PhD was a dream of my father, M. Masood Sadiq (late). Whatever I am today is just because of his constant support, motivation and cherish talk. He was a man who always supported me during hard time of my life. I still remember his words *"Mohsin do you work hard, you will always find me standing beside you to support you"*. His phrases of joy and encouragement always whisper around me. I am also very thankful to my wife, Rabia Khan. During the period of PhD, she continuously supported me and encouraged me to complete my PhD research journey. Whenever I am in trouble, she come up with her peaceful face and look at me with confidence and says *"Mohsin I know, you can do it"*. Her words of motivation always place soul of trust into my body. She gave me three wonderful children who added joy and beauty in my life. I am always grateful to my mother, Tasneem Kausar, for her prayers and my brother, Faisal Masood, for his full support.

I would like to express my profound gratitude to my supervisor Prof. Ivan Glesk, for the opportunity to undertake this work and for his advice, direction and comments when I needed them the most. Thank you for your patience and kind disposition towards me. I am indebted to you. After my father, I found him as my big brother, my uncle and my well-wisher who always supported me. I remember that whenever I get any problem, I used to visit his office and he was a man who can really guide me. The most impressive thing about Prof. Ivan Glesk is that his office doors are always open for his PhD students. We used to have at least one weekly meeting with him and beside this, he always give me time whenever I required. During my research, I learnt many skills from him such as proofreading, presentations, paper writing, how to present your research ideas to others and make research collaborations. I am thankful to my co-supervisor Prof. Ivan Andonovic for guiding me about my research plans.

I am also grateful for my friends for their unconditional friendship. Thank you Mohamed Abuhelala for lots of positive and encouraging statements, like he always used to say; *"Mohsin you are genius man, I know you can do it"*. And, many thanks to Dr. Mohamed Mostafa Fauad

for creating a pleasant and convivial environment for research and making research collaborations. I am also very thankful to Dr. Helana for her absolute friendship.

Abstract

From the network management approach, the term network efficiency signifies the effective utilization of network resources. The critical aspect of managing the Multi-Protocol Label Switch (MPLS) networks is to compute the best routes across the network that guarantees the cohesive traffic flow with the effective use of network resources. Considering the optimal routes in multiple switching based MPLS networks, comprised of multiple domains, serves as a complex and challenging task. Technically, the computation of optimal routes that can be depending on multiple objectives along with constraints introduces the concept of multi-objective subject to multiple constraints (MCOP) optimization problem in the field of optimization, which is considered as a computationally complex optimization problem. Metaheuristic optimization algorithms have raised as a mainstream approach for solving MCOP based complex optimization problems. However, metaheuristic algorithms can generate sub-optimal solutions because rooted problems within algorithms that badly disturbs the algorithm`s performance. Therefore, extensive research on the improvement of algorithms has become necessary. This thesis investigates the particle swarm optimization (PSO), bat, and dolphin echolocation (DEA) algorithms, highlights the problems in the algorithms and offers novel versions of these algorithms as a proposed methodology for the MPLS optimization problem. For MPLS optimization, the offers the MCOP based optimization models which consist of multiple objective functions and are mathematically formulated for experimental setups. For the considered optimization problems, the new metaheuristic algorithms are suggested as the modified and hybrid versions of PSO, Bat, and DEA algorithms. The numbers of experiments are conducted along with extensive results analysis, which demonstrates the performance of

presented algorithms for MPLS optimization, and to validate these algorithm performances, an exclusive comparative analysis is established with other familiar metaheuristic algorithms.

Table of Contents

Acronyms

ABAT	Adaptive Bat Algorithm
AELS	Adaptive Elitist learning Strategy
APSO	Adaptive PSO
ABAT	Adjustable Bat Algorithm
ACO	Ant Colony Optimization
AI	Artificial Intelligence
A-PSO	Ameliorated PSO
ATM	Asynchronous Transport Mode
BA	Bat Algorithm
BBA	Binary Bat Algorithm
BBO	Biogeography Based Optimization
BIH	Bat Intelligent Hunting
CBA	Chaotic Bat Algorithm
CDMA	Code Division Multiple Access
CF	Convergence Factor

CS	Cuckoo Search
DLBA	Differential Operator and Levy Flight Bat Algorithm
DNA	Deoxyribonucleic Acid
DE	Differential Evolution
DECSA	Dolphin Echolocation and Crow Search Algorithm
DA	Dolphin Algorithm
DEA	Dolphin Echolocation Algorithm
DEO	Dolphin Echolocation Optimization
ELS	Elitist Learning Strategy
ESE	Evolutionary State Estimation
FDR-PSO	Fitness Distance Ratio-based PSO
FLBA	Fuzzy Logic Bat Algorithm
GMPLS	Generalized Multi-Protocol Label Switching Network
GA	Genetic Algorithms
GWO	Grey-Wolf Optimizer
HBD	Hybrid Bat Dolphin Echolocation
IBA	Improved Bat Algorithm

IP	Internet Protocol
KMBA	K-Mean Bat algorithm
KH	Krill Herd
LER	Label Edge Router
LFIB	Label Forwarding Information Base
LMMSE	Linear Minimum Mean Square Error
LSP	Label Switch Path
LSR	Label Switch Routers
LO	Loudness Optimizer
MB cost	Minimum Load Balancing Cost
MR cost	Minimum Routing Cost
MPLS-TE	MPLS paired with the Traffic Engineering
MOP	Multi-Objectives Optimization
MCOP	Multi-Objectives with Multiple Constrained-based Optimization
MPLS	Multi-Protocol Label Switching
NFL	No Free Lunch
NP	Non-Polynomial

OSPF	Open Shortest Path First
PMLG-PSO	Pareto-based Modified Local Global Particle Swarm Optimization
PSO	Particle Swarm Optimization
QoS	Quality-of-Service)
SA	Simulated Annealing
SD	Standard Deviation
SI	Swarm Intelligence
TDM	Time Division Multiplexing
TE	Traffic Engineering
UAV	Unmanned Aerial Vehicles
WOA	Whale Optimization Algorithm

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Background

During recent years, the telecommunication networks are integrated with advance and diverse applications which highlighted the status of efficient utilization of resources and network optimization. Conventional IP (Internet Protocol) based networks face major issues for the incorporation and operation of miscellaneous applications that have various traffic characteristics that require exceptional quality of services [1] – [2]. The common problem with IP systems is to attach IP address to each packet across the network, which in return influence network management. Even if all packets are forwarded towards the same destination, each packet would need to be individually reviewed from the routing table to decide the next routing stage in the route towards that destination. This has increased the demand for strategies to manage the congestion and reduce network delays [3] – [5].

Over the past few years, MPLS (Multi-Protocol Label Switched) networks were introduced as an advanced strategy that uses short path labels between network nodes for communication [6]. MPLS networks have maintained a remarkable reputation because they supported combined switching techniques and providing several QoS (Quality-of-Service) parameters with efficient utilization of network resources. The major benefit that MPLS offers is the improved speed of packets across the network [6] – [7]. This higher flow of packets is because of the utilization of labels for setting up a virtual connection between nodes rather than decelerate checking the content of each packet for deciding its destination. For such mechanism of packets forwarding in MPLS networks, the LSR (Label Switch Routers) are employed to forward packets over defined routes. The packets that have the same destination moved from ingress router to its destined egress router without the lookup of routing tables [7] – [10]. The primary approach of TE (Traffic Engineering) is associated with the effective usage of network resources, decreasing the routing costs, improve the network reliability, and fast recovery of unexpected link failures across the network [11] – [13]. Modern applications may require large

data transfer that may produce network congestion, packet losses, and cause poor service provision. For this reason, to structure efficient routing, the optimization techniques are offered to packet forwarding mechanism in the MPLS network domain. The efficient utilization of network resources and performance optimization has become a complex task for network vendors [14] – [16].

In-network phrasing, network efficiency term is used for the effective utilization of network resources. Along these lines, the goal is therefore to determine the optimal path in the network that guarantees the effective usage of network resources. To compute the optimal paths, the objective functions need to be defined that ensures the optimization of the network [17] – [18]. In network optimization, the objective function will be the mathematical optimization function that needs to be optimized (minimized or maximized) according to the optimization model of the network. For the computation of optimal paths in the network, the optimization algorithms are engaged [19] – [20]. Advanced applications based network optimization models are usually dependent on multiple objective functions along with multiple constraints, which introduces the concept of MCOP (Multi-Objectives with Multiple Constrained-based Optimization) problems, which is considered as highly complex optimization problems [20] – [22].

Recently, metaheuristic optimization algorithms have shown up as a popular approach to manage this kind of complex optimization problems. These metaheuristic algorithms are applied for those applications where achieving the optimal solutions are either not possible or computationally too complicated to be accomplished [20], [23]. Metaheuristic algorithms are classified into various groups, in which one of the noticeable classes is the nature-inspired algorithms. Several nature-inspired algorithms are being introduced in the last few years, in which some of them have got noteworthy attention due to presenting the impressive solutions for the optimization problem in various applications [24]. However, there is a complex problem that exists in these algorithms that stimulate researchers to offer either new models or modified models of metaheuristic optimization algorithms. Metaheuristic algorithms have to maintain a balance between their exploring and exploiting the solutions in searching space of the optimization problem domain. The failure of maintaining this balance causes the algorithm to produce sub-optimal solutions for any complex optimization problem [25] – [26]. This challenge

introduces an open field of research to propose such adjustments in the algorithms that can support an algorithm`s performance in terms of producing optimal solutions.

1.2 Research Objectives

When the metaheuristic optimization algorithm suffers from exploration and exploitation problem, then the significance of the algorithm becomes ineffective as it comes with an unproductive methodology that can only offer sub-optimal solutions for the optimization problem. Therefore, before targeting the optimization problem, there is a need to focus on the problems of metaheuristic optimization algorithms and try to understand their important parameters. The parameters used in each algorithm have the vital role of controlling the algorithm`s performance, which requires comprehensive review and then modifies them to improve the algorithm. The improved algorithm then can become a feasible method for the optimization problem.

With the presented scenario, this doctoral dissertation is therefore focused on the development of new or modified/hybrid models of metaheuristic algorithms, which will be applied over the presented optimization models of MPLS networks. For this research, the number of metaheuristic algorithms is considered and is proposed with their modified versions as a methodology for the given MPLS optimization problems. More specifically, the main objectives of the thesis are;

1. To offer MCOP based optimization models for MPLS networks and offer the novel metaheuristic algorithms for the given optimization problem.

2. PSO algorithm suffers with local and global optima problem, which makes it an unacceptable technique for optimization. The research is to develop the new version of the PSO for MPLS optimization problem and generate better results compare to other techniques.

3. Bat algorithm has a slow convergence problem and thus fails to generate optimal solutions for various optimization problems. The objective is to investigate the essential parameters in the Bat algorithm, which can play a vital role for improvement in algorithm and then offered a new bat algorithm for the MPLS optimization.

4. Bat and DEA (Dolphin Echolocation) algorithms, both are dependent on their echolocation feature for solution searching in the optimization domain. Thus, the objective is to propose the hybrid model of the Bat and DEA which can generate better solutions for MPLS optimization compare to traditional techniques.

5. To investigate the offered algorithms performance with other metaheuristic algorithms with comprehensive comparative analysis by implementing several experiments in different scenarios.

1.3 Main Contributions and Research Publications

During the investigation of PSO, Bat, and DEA algorithms, the improved versions of these algorithms are projected as novel algorithms (methodology) for the proposed MCOP based optimization problems of MPLS networks. The work presented in this thesis resulted in several contributions that complement the state of the art in the field of metaheuristic algorithms as well as for MPLS network optimization. Specifically, this research has contributed as;

1. Developed the MCOP based optimization models for MPLS networks, in which multiple objective functions are considered which includes; III). Resource reservation costs, II). Traffic load balancing costs, III). Routing delay costs, IV). Load balancing cost, V). Energy consumption costs and, VI). Constrained based routing costs objective functions.

2. Pioneered a novel version of the PSO algorithm as *PMLG-PSO* (*Pareto based Modified Local Global PSO*) algorithm for the developed model for the MPLS network. The *PMLG-PSO* has been proposed with the Pareto front approach for conflicted based objective function in the optimization model and thus, offered not only non-dominated solutions but also optimal solutions. The proposed method is verified with a comprehensive comparative analysis while applied to the MPLS optimization problem.

3. Investigated the significance of the loudness parameter in bat algorithm, highlight its importance for the improvement of algorithm, and thus offered a new improved version as *ABAT* (*Adjustable Bat Algorithm*). The *LO* (*Loudness Optimizer*) is offered as a new parameter that supports the improvement in the convergence of the algorithms and produced optimal solutions compared to Bat and other familiar metaheuristic algorithms.

4. Proposed a novel hybrid model of the Bat and DEA algorithm, named as *HBD* (*Hybrid Bat Dolphin Echolocation Algorithm*). The *HBD* has provided much improved solutions compared to Bat and DEA algorithms for the developed optimization model of the MPLS network. The proposed algorithm also showed its superiority for mathematical optimization functions in terms of generating optimal solutions. The offered algorithm is briefly discussed in comparative analysis with Bat, DEA, and other well-known algorithms for both mathematical as well as MPLS optimization problems.

The research work presented in this thesis has been published while some of the work is under review in publications. The details of these publications are;

JOURNAL PAPERS

1. **Masood, M.,** Abuhelala, M & Glesk, I 2016, 'A comprehensive study of routing protocols performance with topological changes in standard networks' International Journal of Electronics, Electrical and Computational System, vol 5, no. 8, 6R7K3T, pp. 31-40. IF: 2.56

2. **Masood, M**., Mostafa Eltaweel, M., Kamal, R., Glesk, I. and Khan, I (2019). 'An Improved Particle Swarm Algorithm for Multi-Objectives based Optimization in MPLS/GMPLS Networks', IF: 4.98, IEEE Open-Access Journal'. 10.1109/ACCESS.2019.2934946

3. **Masood, M.,** Mostafa Eltaweel, M. Kamal, R. and Glesk, I. (2020). 'Adjustable Bat Algorithm based on Loudness Optimizer for Network Traffic Optimization', IF: 4.98, IEEE Open-Access Journal. [Under Review]

4. **Masood, M**., Mostafa Eltaweel, and Glesk, I. (2020). 'Echolocation based Hybrid Bat-Dolphin Algorithm for complex optimization', Scientific Report (Nature)'. IF: 4.83 [Under Review]

CONFERENCE PAPERS

1. **Masood, M**, Abu Helala, M & Glesk, I 2016, Detailed analysis of routing protocols with different network limitations. in J Müllerová, D Senderáková, L Ladányi & Ľ Scholtz (eds), Proceedings of the 20th Slovak-Czech-Polish Optical Conference on Wave and Quantum Aspects of Contemporary Optics. vol. 10142, 101421T, Bellingham WA, 20th Slovak-Czech-

Polish on Wave and Quantum Aspects of Contemporary Optics, Jasna, Slovakia, 5-9 September. DOI: 10.1117/12.2256834.

2. **Masood, M**, Fouad, MM & Glesk, I 2017, 'A Pareto based approach with elitist learning strategy for MPLS/GMPS networks' 9th Computer Science & Electronic Engineering Conference, Colchester, United Kingdom, 27/09/17 - 29/09/17. DOI: 10.1109/CEEC.2017.8101602

3. **Masood, M**, Fouad, MM & Glesk, I 2017, 'Proposing bat inspired heuristic algorithm for the optimization of GMPLS networks' 25th Telecommunications Forum TELFOR 2017, Belgrade, Serbia, 21/11/17 - 22/11/17. DOI: 10.1109/TELFOR.2017.8249295

4. **Masood M.,** Fouad M.M., Glesk I. (2018) Pareto Based Bat Algorithm for Multi Objectives Multiple Constraints Optimization in GMPLS Networks. In: Hassanien A., Tolba M., Elhoseny M., Mostafa M. (eds) The International Conference on Advanced Machine Learning Technologies and Applications (AMLTA2018). AMLTA 2018. Advances in Intelligent Systems and Computing, vol 723. Springer, Cham. DOI: https://doi.org/10.1007/978-3-319-74690-6_4

5. **Masood, M.,** Mostafa Eltaweel, M. and Glesk, I. (2018). Analysis of Artificial Intelligence-Based Metaheuristic Algorithm for MPLS Network Optimization. In: 20th Anniversary International Conference on transparent Optical Networks (ICTON), Romania, IEEE. DOI: 10.1109/ICTON.2018.8473751

6. **Masood, M.,** Mostafa Eltaweel, M. and Glesk, I. (2018). Analysis of Artificial Intelligence-Based Metaheuristic Algorithm for MPLS Network Optimization. In: TRANSCOM 2019, International Scientific conference on sustainable modern and safe transport, High Tatras, Novy Smokovec - Slovak Republic. DOI: 10.1016/j.trpro.2019.07.205

1.4 Organization of the Thesis

This thesis is structured into seven chapters, **chapter one** provides a summarized background of the research, research aims, and objectives, research contributions along with a list of publications and thesis outline.

Chapter two introduces the MPLS networks and their structure. The chapter provides a basic understanding of MPLS using label switched paths over label switching routers based network. The importance of traffic management is discussed in the chapter in the form of traffic engineering based MPLS networks and highlighted the multi-objective based optimization problem in MPLS networks in the discussion section.

Chapter three briefly explained the proposed methodology that will be used for the optimization of MPLS networks. The chapter can be categorized into two groups, where the first group describes the concept of optimization, its types such as single objective and multi-objective based optimization, and computational complexity. The second section of the chapter presents nature inspired metaheuristic algorithms as an approach for MPLS optimization problem. The problems in metaheuristic algorithms are debated here and provide the platform for research in the field of metaheuristic algorithms.

Chapter four provides the optimization model for MPLS networks in the form of a multi-objective based optimization problem. The chapter proposed a novel *PMLG-PSO* algorithm for network optimization and the experiments are conducted for presenting non-dominant Pareto solutions as well as optimal solutions for each objective function. The Pareto front is presented with various versions of *PMLG-PSO* and the results of proposed algorithms are also compared with other metaheuristic algorithms to validate the efficiency of the algorithm.

Chapter five highlights the significance of the *loudness parameter* by implementing the various versions of the proposed *ABAT* algorithm. The experimental setup is defined for the optimization model of MPLS, which consists of load balancing costs and routing costs objective functions. For MCOP based MPLS network optimization problem, the proposed *ABAT* is applied for various experimental setups and the results are discussed in the chapter in terms of *ABAT* performance for MPLS optimization as well as with comparative analysis.

Chapter six introduced a new hybrid model of *HBD* algorithm for the proposed MCOP optimization problem of MPLS networks which consists of energy consumption costs and constrained based routing costs objective functions. The proposed *HBD* algorithm is applied for the MPLS optimization model with various numbers of experiments and the results are discussed in the form of Pareto front and comparative analysis (including convergence rate and statistical parameters). Furthermore, to validate the *HBD* superior performance in terms of convergence and generating optimal solutions, the algorithm is also tested in the experiments of mathematical optimization functions and compared with other famous metaheuristic optimization algorithms.

Finally, **chapter seven** is a summary of the thrust of the findings of all research aspects presented in this thesis, conclusion, direction, and recommendations for future work.

Chapter 2
Research Background

2.1 Introduction to MPLS Networks

Traditional IP switching is based on the method of transmitting data packets over the network where the packets are embedded with an identifier or a tag [5], [27]. In some aspects, standard IP routing is considered to be a type of packet switching where each packet is loaded with the destination IP address. This destination IP address is used for looking up information in the routing table to decide on the next hop en-route to the destination [28]. However, IP routing has had issues regarding scalability and speed. This has led researchers to investigate alternative methods to data packet switching. Added to the aforementioned concerns is the need to promote additional functions such as traffic engineering and traffic aggregation [29].

Multiprotocol Label Switching based networking has been used for years now. MPLS is famous for its unique networking methodology in which the labels used are attached to the packets to forward them within the network. It is defined as the set of protocols used in sophisticated telecommunication networks in which the data packets are tagged with unique labels in the path from the source to the destination node. Instead of using a destination address, labels are used as an identifier of the short path from one node to another. In MPLS, the labels are advertised across the routers in the network to build label-to-label mapping. IP packets that are tagged with labels enable routers to transmit the traffic across the network by identifying their labels. Thus in MPLS-based networking, the packets are routed by the label switching instead of IP switching. MPLS networks further assist the frame relay, ATM (Asynchronous Transport Mode), and internet protocols. For instance, an LSP (Label Switch Path) can be established for any of the aforementioned protocols. With the growing prevalence of MPLS networks, the concept of the MPLS network is now extended. It not only includes lambda (photonic or wavelength) switching but also TDM (Time Division Multiplexing) switching, Layer 2 switching, fiber 8 switching, and the current packet of cell and frame switching technologies. Lambda switching based MPLS is used in optical networking to switch

wavelengths of light onto separate paths. In TDM based MPLS networking, two TDM networks or devices are connected using MPLS domain. This idea gives rise to the introduction of Generalized Multiprotocol Label Switched (GMPLS/ Generalized MPLS) networks that support all of the aforementioned technologies [30] – [33].

2.2 MPLS Architecture

MPLS-based networks do not use IP destination addresses in their routing mechanism. The routers in this type of network are independent of traditional IP address look-ups in terms of their routing table. Therefore an MPLS network uses a different mechanism for routing which depends on the different modules in its architecture [30] – [33]. These are discussed in the following sections.

2.2.1 Label Switch Router

A label switch router is an MPLS-supported router. It is used to recognize MPLS labels in terms of receiving and forwarding labeled packets across the network. There are three types of LSR used in MPLS networks [30] – [33]:

- **Ingress LSRs**:

An ingress LSR is the edge router of the MPLS network on the source side of the network, also known as an LER (Label Edge Router). At the ingress LSR, the packet received is not tagged yet. A label (stack) is inserted in the front of the packet at the ingress LSR and then it is forwarded to the data link.

- **Egress LSRs**

An egress LSR is an LER that is at the destination end of the LSP in the MPLS network. Labeled packets are received at the egress router which then removes the tagged label from the packet and forwards the packet to the destination node.

- **Intermediate LSRs**

Intermediate LSRs or LSRs are intermediate routers within the MPLS domain that receive labeled packets, conduct an action on it, switch the packet and then transfer it to the

appropriate data link or the next LSR in an established LSP. An LSR can execute three operations such as push, pop, and swap.

In the MPLS domain, every LSR is capable of popping/removing one or multiple labels from the top of the label stack before switching the packet. In the same way, every LSR must be able to push/ insert one or multiple labels into a packet before transmitting it to the MPLS network. If the packet is not labeled, then the LSR creates a label stack and pushes it onto the received packet before transmitting it to the next stop. If the packet received is already labeled, then the LSR pushes one or more labels into the label stack before switching out the packet [30] – [33].

An LSR pushes a label onto a packet that has not yet been tagged with a label. The LSR that pushes the label is also known as an *imposing* LSR. This is as it imposes the label onto the packet. Ingress LSRs come under the definition of an *imposing* LSR. This is ingress LSRs as the first LSR in the MPLS domain that imposes/push a label onto a packet. The LSR that disposes of all of the labels from the packet before switching it is considered to be a *disposing* LSR. From this perspective, an egress LSR is a *disposing* LSR [30], [32] – [33].

2.2.2 Label Switched Path

In an MPLS-based network, before transmitting the data, a path is created between the label edge routers (ingress and egress LSRs) through any intermediate LSRs. In other words, an LSP is an established path across the MPLS network that consists of a sequence of intermediate LSRs between the LERs used to transmit only labeled packets. The LSR that only pushes labels onto a packet such as ingress LSRs is the first LSR in the LSP. The egress LSRs are the last LSR in this path and it is the LSP that supports only labeled packet transmission [31], [33]. Fig. 2. 1 shows the unidirectional established LSP in the MPLS domain. In this figure, the labeled packet flow is from left to right while for the left to the right direction, another LSP can be established if required.

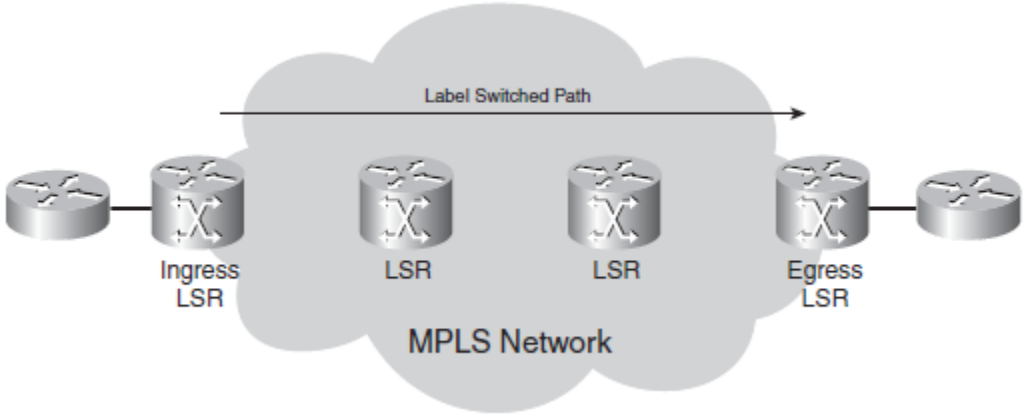


Fig. 2. 1: An LSP in MPLS Network [33]

Only ingress LSRs do not need to be the first LSR that supports LSP establishment. There is a chance that an LSP can be established between many intermediate LSRs. This is considered to be a *nested* LSP concept. An LSP inside another LSP is called a *nested* LSP, as shown in Fig. 2. 2. In Fig. 2. 2, an LSP is established from the second intermediate LSR (right side) to the fourth intermediate LSR within the MPLS domain. It is worth noting here that although the packet is already labeled from the ingress (first) LSR when the packet is received by the second LSR of the network, the LSR pushes another label onto the packet. This means that the label stack of the packet has two labels; the upper label is for the *nested* LSP while the bottom label is associated with the first LSP (that covers the entire MPLS network).

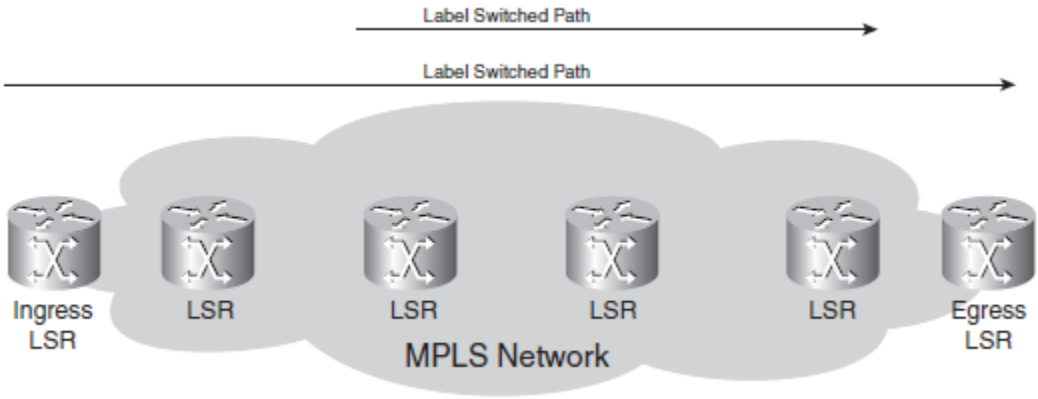


Fig. 2. 2: Nested LSP in MPLS Network [33]

2.2.3 Label Switching

The MPLS network relies on the label switching concept which can be described as a small and fixed-format label tagging a packet so then the packet can transmit in the MPLS domain. In other words, each packet/cell or frame needs an identifier so then the packet can be forwarded to the network. Within the MPLS network, at each hop, the packet transmission is dependent on the information tagged as an incoming label. It is then transmitted onward based on the new label. In the network at each intermediate LSR, the incoming label is swapped and the data is switched and supported based on the label value. This concept comes with two new terms; label swapping and label switching. In an MPLS-based network, the label is pushed onto a packet with a supplementary piece of information, known as the shim header [31], [33]. The shim header is inserted between the network header and IP header, as shown in Fig. 2. 3.

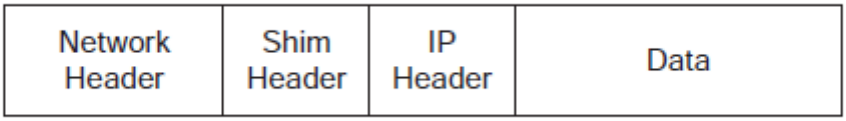


Fig. 2. 3: The shim header between the network header and IP header [31]

The shim header contains a 20-bit label and this label is used for path identification followed by a packet. Each intermediate LSR maintains a look-up table. This is known as the LFIB (Label Forwarding Information Base). The look-up table (LFIB) is used by the LSR for the identification of the next hop in the data transmission taking place in the network. Principally, the LFIB is comprised of the mapping of incoming labels and the incoming interface through to the outgoing label and outgoing interface. This means that when the LSR receives an incoming packet, it detects the incoming interface from which the packet is received and it finds the label in the packet shim header. The LSR then looks up the label values in the LFIB and finds the outgoing interface for where the packet needs to be sent. The new label value is placed in the shim header and the packet is then sent to the next hop. This procedure continues for each

intermediate LSR of the network and the path followed by this procedure is called a label switch path. Once the LSP is established from the LER/ingress LSR to the LER/ egress LSR, every intermediate LSR maintains this LFIB as its look-up table. The data follows this particular LSP path-based technique. However, the only complication occurs at the LER/ ingress LSR where the packet must be categorized according to its destination and the service offered. This may depend on the form of application or the level of service requested before being allocated to a particular path/ LSP [31], [33].

2.2.3.1 Packet Transmission in MPLS Networks

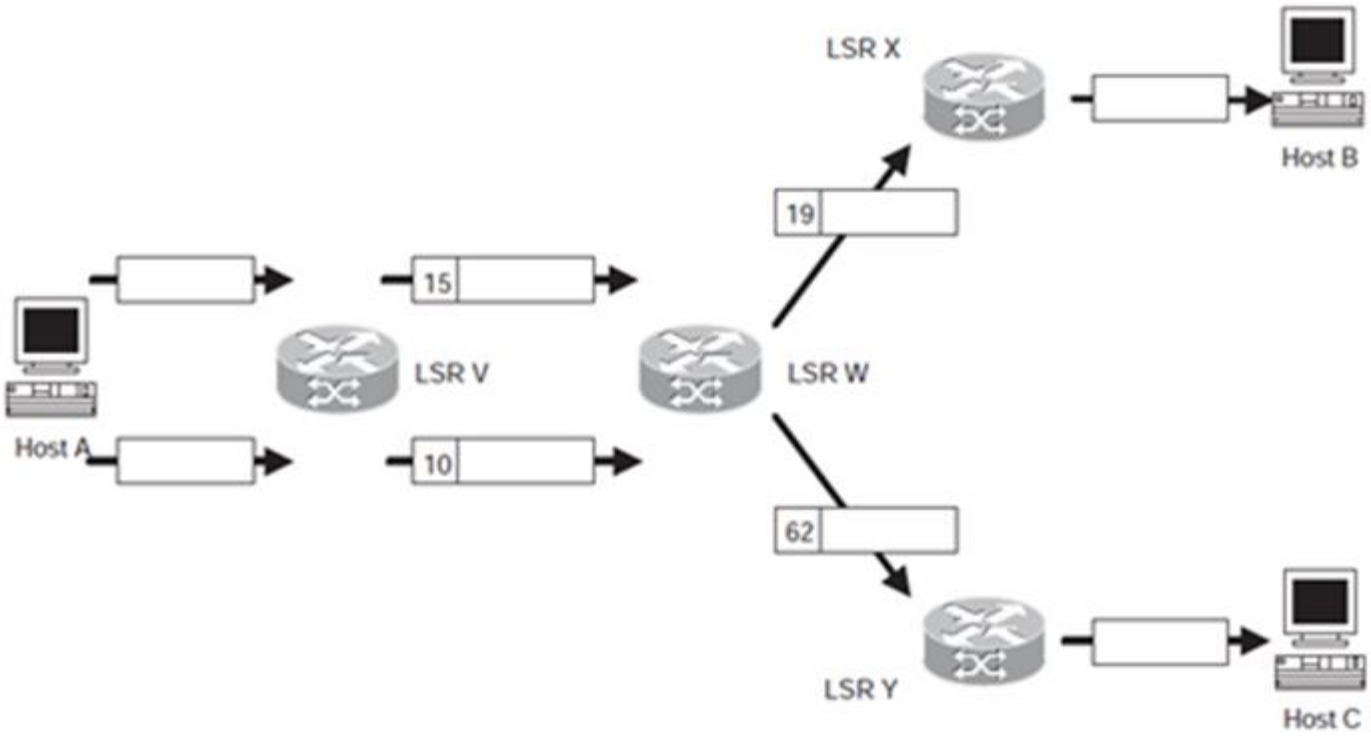


Fig. 2. 4: Packet Forwarding Mechanism in an MPLS Network [31]

The packet transmission mechanism in MPLS based network can be explained with help of an example as shown in Fig. 2. 4. In Fig. 2. 4, two LSPs are established. There is one LSP from Host A to Host B and another LSP from Host A to Host C. In the given example, there are four intermediate LSRs and they are used for packet forwarding in the MPLS domain. In the

beginning, with the support of the routing table or default route, Host A sends the IP packets to the next connected LSR V. In the MPLS network, LSR V works as ingress LSR, whose job is to categorize the packets according to their particular destination. The ingress LSR supports the LSP establishment towards the specific egress LSR. To do this, the ingress LSR allocates the LSP to a packet and tags them with labels by adding a shim header and set the label values. According to Fig. 2. 4, the packets that are targeted towards Host B are allocated to the upper LSP, called LSP 1, and the packets that are going towards Host C are assigned to the lower LSP, called LSP 2. At the ingress LSR, it can be seen that the LSP 1 packets are labeled with a value of 15 while the packets going over to LSP 2 are assigned with a label value of 10. Once the packets, according to their appropriate LSPs, are labeled, then they are forwarded out of their relevant interface towards the next intermediate LSR. As an intermediate LSR, LSR W's first job is to examine each labeled packet from the incoming interface for its incoming label. LSR W finds the outgoing label and appropriate outgoing interface for each packet using the look-up in the LFIB. The label values are swapped, such that the incoming label value is replaced with the outgoing label value. The packets are then forwarded out of their specified outgoing interfaces. In Fig. 2. 4, at LSR W, the old packet label value of 15 is replaced and forwarded towards the interface of LSR X with the swapped new label value of 19. Note that the transmission of this packet is done on LSP 1. Similarly, the packet label value of 10 received at the incoming interface of LSR X is replaced with a new label value of 62 before it is forwarded out to the interface of LSR Y over LSP 2. In the given MPLS network, LSR X and LSR Y are egress LSRs and they are specified as LSP 1 and LSP 2 respectively. As egress LSRs, the LSRs also use the look-up in the LFIB when they receive a labeled packet in their incoming interfaces. The egress LSRs (LSR X and LSR Y) remove the shim header along with the label values and forward the packet as a plain IP packet. The packet forwarding from the egress LSRs (LSR X and LSR Y) can be done with the support of a routine IP routing table. LFIB helps to optimize this method by providing the appropriate outgoing interface for the packet transmission. This means that there is no need to use a normal IP routing table at this stage. In the given example of Fig. 2. 4, it can be concluded that the packets are transmitted optimally within the MPLS domain using two separate LSPs according to their particular destination from a single sender host through the label terminology pathway.

2.4 Traffic Engineering

The fundamental concept of traffic engineering is the optimal utilization of network resources, particularly the links that are underutilized as they do not sit on the chosen path. This implies that TE will provide a way of directing traffic on these routes across the network. This is different from the chosen routes that are computed in the least-cost methodology provided by IP routing [33], [34]. In networking, the least-cost is considered to be the shortest path that is computed by dynamic routing protocols [33]. With the implementation of the TE concept in the MPLS network, the traffic can have a specific QoS flow from one point (say point A) to another point (say point B) through a path that will be different from the least-cost path. As a consequence, the traffic will be distributed more uniformly across the available links to allow for the better use of any underutilized links in the network.

MPLS with a TE enabled concept can be explained with the help of Fig. 2. 5. In Fig. 2. 5, using the TE concept, the traffic can be directed from Host A to destination Host B using the lower path (LSR P, LSR Q, and LSR S) instead of the upper path (based on LSR X and LSR Y). Note that the lower path (LSR P, LSR Q, and LSR S) will not be considered the shortest path as there are four hops compared to the three hops in the upper path. This may lead to a uniform distribution of any traffic over the underutilized links across the network.

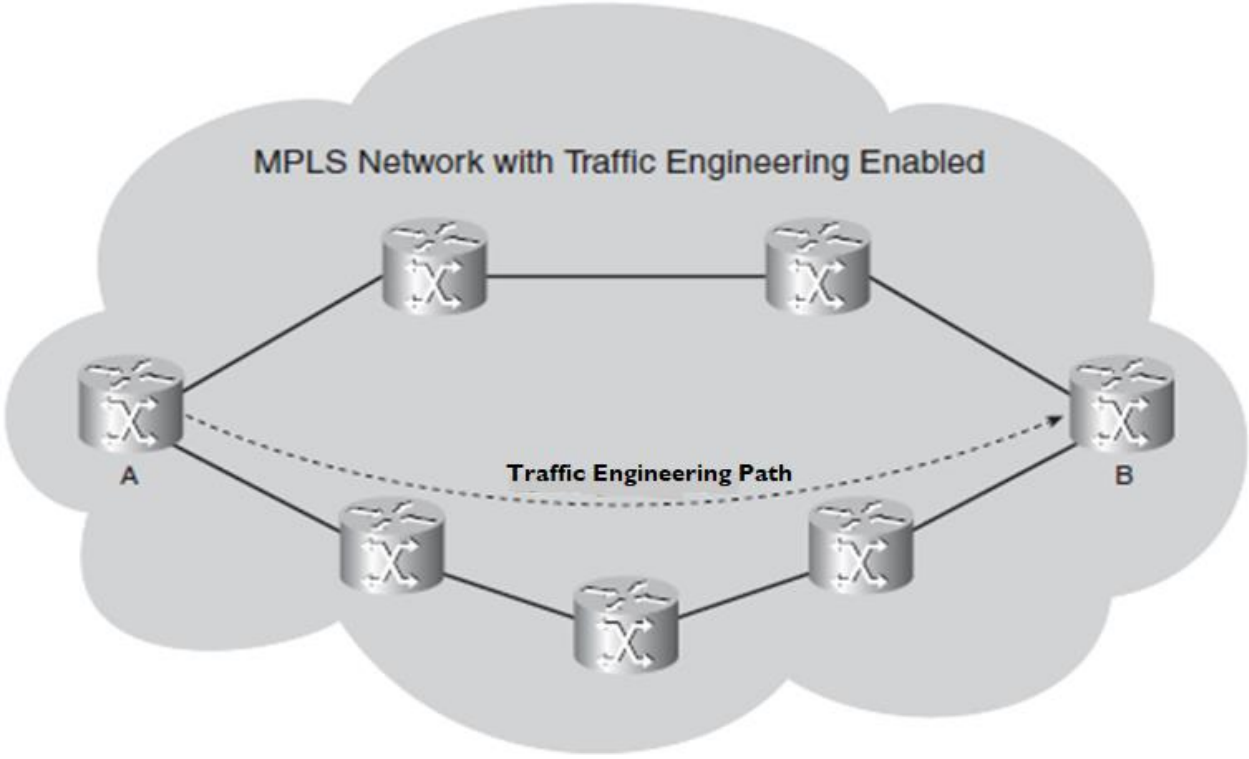


Fig. 2. 5: Traffic Engineering (TE) Enabled network [33]

2.5 MPLS Traffic Engineering

The function of TE is to direct traffic from one edge to another edge in the most optimal way across the network. Routing methodology supports the IP networks when they are routing the traffic within the network. The routing concept depends on the least-cost rule in which the shortest path is computed and then the traffic is routed through the computed least-cost routes. Each IP routing protocol has a cost related to the network links. The aggregation of each link cost is considered to be a path and it is used for the computation of the shortest cost path. The shortest path (based on the number of connected links with the least-cost) is computed using the shortest path algorithm. Once the shortest path is computed, the traffic is routed using this path across the network [33] – [36].

MPLS-TE (MPLS paired with the Traffic Engineering) concept employs the traffic engineering rule by establishing an optimal path (known as LSP) that is based on the optimal

route computation from the ingress LSR towards the egress LSR. For optimal path computation, the ingress router must contain information about the updated network topology. The ingress LSR must also contain information on the remaining bandwidth of all of the network links. In conclusion, LSPs can be established across the MPLS network. MPLS-TE provides source-based routing as a replacement for IP destination-based routing. This is because MPLS uses label switching. This is where MPLS forwards the packet in the data plane and matches the incoming label in the LFIB before swapping it with the outgoing label. This means that the head-end LSR of the LSP is the entity that is used to determine the routing of the labeled packet, where all LSRs agree on what label is used for what LSP [33] – [34].

Any network that has label switch routers (LSRs) can use the MPLS-TE methodology. However, it is also noteworthy that the head-end LSR of the LSP must-have information on the link bandwidth and other attributes. In addition to this, the link-state routing protocols are used between the head end and tail-end LSRs in the deployed MPLS-TE network. The reason for using link-state routing protocols in an MPLS-TE based network is that each LSR maintains the information of the current state of its connected links which is then flooded as a form of information across the LSRs of the network. In other words, in an MPLS-TE-based network, all LSRs have updated information about the network topology available to them [33] – [34].

In summary, MPLS-TE provides the following benefits [33]:

- MPLS-TE guarantees the effective distribution of traffic across the network by avoiding over-utilized and under-utilized links in the network.
- MPLS-TE considers many link attributes (for example, delay or jitter) when computing paths.
- MPLS-TE has the capability of automatically changing the link attributes and bandwidth.
- Source-based routing supports traffic engineering in MPLS as opposed to IP destination-based routing.

2.6 MPLS Network Optimization

In recent years, there has been a significant development in terms of information systems and communication networks dealing with complex optimization problems in the fields of network architecture and routing. Advanced network routing technologies deal with multiple QoS requirements. This leads to designing routing models to compute the number of network routes while satisfying the quality of service constraints and optimizing the route-related objective functions. Routing problems in communication networks can be defined as the computation of a series of network routes or paths that are computed by the optimization of some of the objective functions while satisfying the constraints in the network. Various routing decisions related to performance can be measured using route-related selected metrics. The potential benefit of formulating routing problems in communication networks as multiple objective-based optimization problems is the creation of trade-offs between the different network costs and metrics that are conflicting to each other. This allows them to be analyzed in a coherent manner [37] – [40]. For MPLS networks, various routing models dependent on multiple objective functions have been proposed in the context of emerging routing problems in MPLS networks. The optimization problems in MPLS networks can be formulated as a combinatorial optimization problem that involves the computation of optimal paths/routes with either single or multiple objective functions subject to the multiple constraints within a discrete set of feasible paths in the network. MPLS networks, which offer advanced and diverse services, deal with routing problems related to path computation that are considered to be NP-hard (Non-Polynomial) class optimization problems. This means that an optimal path computation that is dependent on multiple objectives and multiple network constraints in MPLS networks cannot be solved optimally in a feasible computational time. For this situation, optimization approaches have been adopted by researchers to solve the MPLS optimization problems encountered with approximate optimal solutions [20], [41] – [43].

2.7 Conclusion

Network service providers and network vendors usually prefer to offer MPLS-based networking to their customers as it offers advanced and miscellaneous applications to the end-users without sacrificing the QoS requirements. An MPLS network uses labels instead of the IP destination address. This reduces the complexity of the look-up table and routing table at each router. This method reduces the complexity of the network, thus it uses a label switch path to forward the data traffic within the network. MPLS networks can also be loaded with a traffic engineering entity that is used for traffic management. Offering multi-objectives-based routing in an MPLS network can be considered an optimization problem in an MPLS network. This means that while taking on the routing decision in an MPLS network, the network considers multiple objective functions and multiple constraints in route computation. Most of the objective functions are conflicted with each other and also dependent on multiple constraints. This makes it an NP-hard class optimization problem. Therefore the idea is to find an approach that can not only offer an optimization model for MPLS networks but also generate optimal routes using an appropriate and effective path computation algorithm.

Chapter 3

Network Optimization

3.1 Optimization

Optimization is related to the field of statistical analysis and applied mathematics. Problems related to achieving better solutions in the searching domain of different fields such as engineering sciences, economics, and life can be designed as an optimization problem. Optimization is the problem of finding the optimal solutions from all feasible solutions. Technically, it is the process of finding the decision variables that result in a maximization or minimization of an objective function. Several terms are used when designing optimization problems such as constants, parameters, constraints, and objectives that need to be achieved. The job of the problem designer is to craft the problem and to quantify the benefits of the proposed design. The designer also presents the initial design or baseline line for the optimization algorithm (used for the optimization problem) [20], [43]. The optimization process can be illustrated in the form of a figure designed by an engineer, as shown in Fig. 3. 1. In Fig. 3. 1, it can be seen that the designer describes the optimization process as including modules such as the initial design, evaluating performance, a condition, and a change in the design. All modules mentioned in the optimization process work with the support of the optimization algorithm that helps to optimize the solution in the problem searching space.

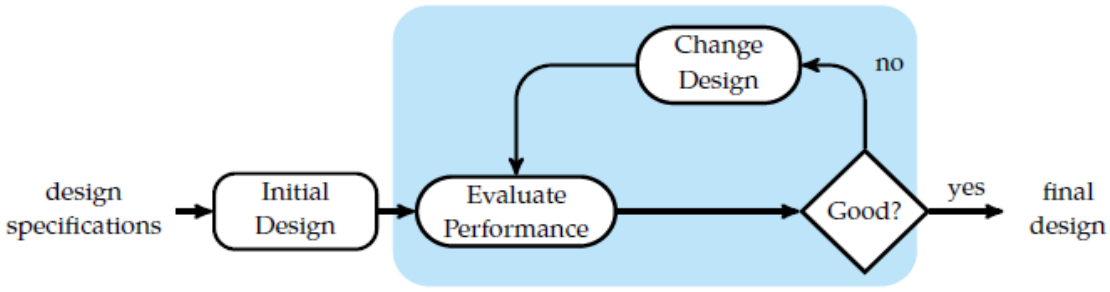


Fig. 3. 1: Optimization Model [43]

To find an optimal solution, optimization can comprise of a broad variety of problems. Consequently, the identification and classification of optimization problems can be done in different ways, and in general, the optimization approaches may often vary greatly from problem to problem. For this reason, a unitary solution is not feasible. This is why the nature of the optimization problem relies primarily on the practical aspects of its objectives, functions, and constraints [43].

Most optimization problems can be written in a general mathematical form as follows [43]:

$$\text{Objective function} = \underset{x \in \mathcal{R}^n}{\text{minimize}}\, f_i(x), \qquad (i = 1, 2, 3, \dots\ M) \tag{3.1}$$

$$Subject\ to\ \ \phi_j(x) = 0 \qquad (j = 1,2,3, \dots, J) \tag{3.2}$$

$$\psi_k(x) \leq 0, \qquad (k = 1,2,3, \dots, \mathrm{K}) \tag{3.3}$$

$$where\ f_i(x), \phi_j(x)\ and\ \psi_k(x)\ are\ functions\ of\ the\ design\ vector \tag{3.4}$$

$$x = (x_1, x_2, x_3, \dots, x_n)^T \tag{3.5}$$

In the aforementioned equations, there are important components presented that are used for the mathematical formulation of the optimization process. These include variables, the objective function(s), and constraints. In equation 3.5, x_i of x are the variables known as the decision variables and they can be in either discrete or continuous form. They may also be a mixture of these two forms. Eq. (3.1) symbolizes the function $f_i(x)$, which is the minimization of the function. This function is the objective function, which can be one objective function $if\ M = 1$. It can be two or more than two objective functions $if\ M > 1$. The objective function(s) can also be referred to as either a fitness function or a cost function. The space utilized by the decision variables is known as the searching space, denoted as $\mathcal{R}^n$ and the space that is formed by the objective function(s) is called the response space or solution space. The constraints linked to the objective function(s) can be classified into equality-based constraints and inequality constraints, which are denoted as $\phi_j(x)$ and $\psi_k(x)$ respectively.

It is should be noted here that in Fig. 3. 1, the minimization of the objective function is applied. This can be formulated as the maximization of the given function, and this depends on the optimization problem at hand. The decision variables help us to categorize the optimization problem formulation. This means that if the decision variables are discrete, then the optimization is known as a discrete optimization. If the decision variables are continuous, then the optimization problem is in the continuous optimization problem domain. Discrete optimization can also be categorized as a combinatorial optimization problem. A combinatorial optimization problem can be defined as finding the optimal solutions from within a feasible set of given solutions. Combinatorial optimization has gained significant popularity among researchers from various fields because this optimization class is linked with graph theory. It is used to solve different optimization problems such as those of traveling salesman, network routing, vehicle routing, airline scheduling, minimum spanning trees, and knapsack problems [43] – [44].

3.2 Computational Complexity of Optimization Problems

Computational complexity theory focuses on the classification of computational problems according to their underlying complexities and on connecting these groups. Computational problems refer to a job that needs to be solved by a computer with the support of a sequence of well-defined steps, known as algorithms. The problem complexity or difficulty depends on the number of resources used and whatever algorithm is used to solve the problem. In other words, the question referring to the reliability and usefulness of the algorithm depends on how complex the optimization problem is. This theory introduces the mathematical models of computations that are used to study the problems and measure their computational complexity such as the number of resources required to solve the problem. In the computational field, these resources are focused on time and memory requirements. An algorithm is required to run according to the number of resources available, which varies with the size of the input problem (n). [20], [43] – [45].

3.2.1 P, NP, NP-Hard and NP Compete

In computational complexity theory, there are some problems where the answer can either be "Yes" or "No". These problems are known as decision problems. The process of finding the answer to a decision problem with the help of an algorithm is known as the *decision procedure* of that problem. A complexity class contains a set of computational problems and these problems can only be solved using a specific amount of resources. There are four well-known classes of problem, explained below [20], [43] – [45]:

- **P- Class:**

P-class is the class of decision problems that can be solved by an algorithm in polynomial time. In other words, this class problem is solvable in polynomial time if an appropriate algorithm is used.

- **NP- Class:**

A decision class problem that gives a solution to the problem which can be verified in polynomial time is called an NP-class problem. It is a class of problems in which the solution computation in polynomial time is not necessary but the solution must be verifiable in polynomial time.

- **NP-Hard:**

A problem is considered to be an NP-hard class problem if it is at least as hard as the hardest problems in NP. For example, problem A is considered to be an NP-Hard if problem B in NP can be reduced to problem A in polynomial time.

- **NP-Complete:**

A problem is known as NP-complete if it is NP-hard as well as NP.

The relationship between the four aforementioned classes of problems is shown in Fig. 3. 2.

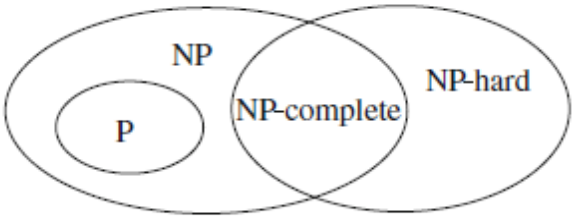


Fig. 3. 2: The Relationship between the P, NP, NP-Hard and NP-Complete Classes [45]

3.3 Multi-Objective Optimization

Optimization problems with a single objective function/fitness function are known as scalar optimization problems [103]. This is as the objective function of these problems always approaches a single value as the globally optimal solution (a scalar). As discussed before, optimization problems may have more than one objective function, known as multi-objective functions, which form a vector. This is known as vector optimization. In a MOP (Multi-Objectives Optimization) problem, the task is to find the variable vector x in the problem domain X that optimizes the objective vector $f(x)$. In simpler words, the MOP is the optimization of multiple conflicting objectives. Mathematically, the MOP can be expressed as follows [44] – [47]:

$$minimization \ \ f(x) = \left(f_1(x) f_2(x), f_3(x), \ldots, f_n(x)\right) \quad , \quad x \in X \tag{3.6}$$

$$subject\ to \qquad g_i(x) \leq 0 \quad , \quad i = 1, 2, 3, \ldots, M \tag{3.7}$$

$$h_i(x) = 0 \quad , \quad i = 1, 2, 3, \ldots, P \tag{3.8}$$

Where multiple objective functions are mentioned as $f(x), i = 1, 2, 3, \ldots n)$ and they are subject to multiple equality-based constraints and inequality-based constraints such as $h_i(x)$ and $g_i(x)$, respectively. In the case of conflict-based objective functions, the quality of one objective function will compromise or decrease compared to the other objective function. Therefore for conflict-based MCOP problems, there is no single optimal solution. Instead, there

is a balanced trade-off between the objective functions. Let us summarize the various optimization problems with multiple objectives below [44] – [47]:

- **MOP (Multi-Objectives Optimization) Problem:**

Computation of an optimal solution subject to multiple objective functions.

- **MCOP (Multi-Objectives with Multiple Constraints-based Optimization) Problem:**

Finding an optimal solution with multiple objective functions subject to multiple constraints.

To solve either a MOP or MCOP optimization problem, the Pareto method has significant popularity, which will be explained in the following sections.

3.1 Non-dominant Solutions and the Pareto Front for Multi-Objective Optimization

Pareto Front provides a set of solutions for multi-objective optimization problems in which there is a balanced tradeoff between conflict-based multiple objective functions. The Pareto approach is used for contrary objective functions where there is no way of improving one objective function without weakening another objective function. In multi-objective functions, the dominance concept is very useful for understanding the comparison between multiple solutions and their respective ranking. During the optimization process, the Pareto method stores the elements of the solution vector separately and then uses the methodology of dominance in the form of dominant and non-dominant solutions. Dominant solutions represent the set of optimal values that can be produced when one objective function increases with the reduction of another objective function in value. In simple words, one objective function value dominates the solutions to the other objective functions, which is a Pareto optimal condition. The set of optimal solutions is known as a Pareto optimal solution. The non-dominant alternative represents a solution in which one objective function is improved without the reduction of another objective function. This is known as a non-Pareto optimal solution. Therefore for the given state, a Pareto Front model is used that displays a set of non-dominant solutions for multi-objective functions [43], [45].

Understanding the Pareto Front concept with the help of an example can be seen in the form of Fig. 3. 3. Consider two multi-objective functions (opposed to each other) and the non-dominant solutions depicted in the two-dimensional space of Pareto-Front. The objective function is the minimization of two objective functions $f_1(x)$ and $f_2(x)$. The dominant solutions generated by both objective functions are represented as $p_7, p_8, \ldots, p_{21}$ while the non-dominant solutions are p_1, p_2, p_3, p_4, p_5 and p_6. The Pareto front is formed by the combination of the aforementioned non-dominant solutions, as shown in Fig. 3. 3.

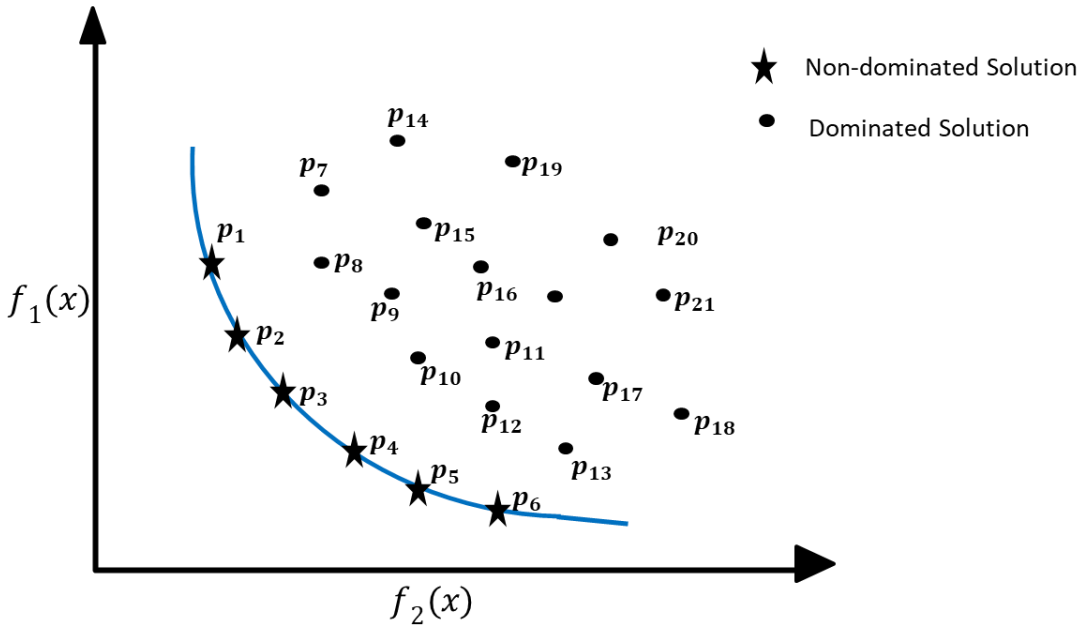


Fig. 3. 3: Demonstration of the Pareto Front

3.4 Nature-Inspired Computation

Optimization is the process of searching for optimal solutions in the problem searching space. Searching is an important concept used in AI. Searching spaces are used for searching for solutions with the assistance of intelligent agents from which originated the standards for computational intelligence. Computational intelligence is the branch of the AI field that explores the adaptive mechanisms used to assist intelligent actions in diverse environments [44] – [45]. Unlike AI which relies on human knowledge and intelligence mechanisms, computational

intelligence depends on the amount of data that is obtained. This field contains a collection of nature-inspired computational models. Some of the major subjects in computational intelligence encompass the fuzzy system (used for reasoning in uncertain situations), neural networks for the recognition of the patterns and evolutionary computation for stochastic optimization searches [48] – [50].

Nature is the key source of motivation for modern computational models. Wiener`s cybernetics, for example, is influenced by the feedback control operations that can be found in biological systems. Various changes in nature on either a microscopic scale or on a biological level can be viewed from a computational approach. Natural systems regularly maintain an ideal balance to reach an optimal equilibrium. These analogies may be used to consider practical approaches to search and optimization. Natural computing paradigms include the examples of SI (Swarm Intelligence), GA (Genetic Algorithms), SA (Simulated Annealing), quantum computing, artificial immune systems, artificial neural networks, DNA-based molecular computing, membrane computing, and cellular automata Biological species maintain social interactions varying from collaboration to rivalry. Mutual actions in a biological population originate from the concept of SI [44], [51] – [53].

Cooperative problem solving is a strategy that helps to accomplish a particular goal through the coordination of a community of independent bodies. In agent-oriented computing paradigms, cooperation mechanisms are popular, regardless of whether or not they are biologically dependent. Researchers in the field of biology, economics, and multi-agent systems are motivated by cooperative behavior. Game theory explores both competitive conditions and the level of cooperation between different groups. It has numerous uses in military strategy, culture, and social issues, in addition to animal behavior and democratic voting. The microscopic cell structure of life offers inspirational knowledge in the fields of evolutionary computing, membrane computing, and DNA computing. Evolutionary computation involves the evolutionary process in a community of individuals over the generations. It produces offspring through mutation and recombination and chooses the fittest to survive from each generation. Membrane computing and DNA computing are new molecular-level computational paradigms. Quantum computing is defined by the concepts of quantum physics merged with computational intelligence [44], [54] – [55].

3.5 Metaheuristic Algorithms

Exact optimization techniques are known for solving the optimization problems of low complexity, whereas most real-life optimization problems are very complex and therefore impossible to be solved through exact optimization approaches [56] – [57]. On the contrary, approximate algorithms have gained noteworthy attention for solving complex optimization problems that reside in the complexity class of NP-Hard. Approximate algorithms can be classified into heuristic and meta-heuristic techniques/algorithms. Heuristic signifies the art of finding new approaches. Heuristic techniques relate to the methods of problem-solving and learning that are dependent on knowledge. This approach provides an acceptable solution in a feasible amount of computational time, but the solution might not be optimal. Heuristic approaches are problem dependent; this means that they are designed to solve specific problems [45], [58] – [62].

The word $meta$ is derived from old Greek, meaning $upper\ level$. In 1986, Glover coined the word meta-heuristic to refer to the set of approaches that are conceptually graded above heuristic in the sense that they provide strategies that can be used to develop heuristic algorithms [133]. Metaheuristics are a high-level technique that is built to discover, produce or choose a lower-level (heuristic) technique that is in turn used to generate satisfactorily a better solution to an optimization problem. Compared to heuristic methods, metaheuristic methods generate improved solutions with less computational effort or fewer computer resources [45], [61] – [63]. Metaheuristics can be categorized as either a single-solution method or as a population-based strategy. Single-solution dependent meta-heuristics are focused on a single solution at any time and they include local-search dependent meta-heuristics algorithms such as tabu search, random search, iterated local search, solis-wets algorithm, variable neighborhood search and guided local search [64] – [69]. Population-based metaheuristics offer several solutions that can be improved over progressive iterations until the condition of termination is satisfied. Population-based metaheuristics are commonly classified into either swarm-based or evolutionary-based algorithms [70] – [71]. Single solution-based metaheuristics are known to be more exploitation-oriented methods, in which the algorithm is inclined towards the local searching space. In contrast, population-based metaheuristics are considered to be a more exploration-oriented approach, where the algorithm tends towards global searching. Furthermore,

metaheuristics are problem independent. This means that they can be applied to solve any complex optimization problem heuristic [45], [70], [72] – [73].

Metaheuristic algorithms are used as a choice of optimization approach for complex optimization problems where a heuristic algorithm fails to provide an acceptable solution. Several studies have discovered that metaheuristic algorithms have proven to be an efficient alternative with complex and large time-computational dependent problems [74] – [75]. Over the last few years, metaheuristic algorithms have obtained significant attention by solving engineering and communication network-based optimization problems. The significance of these algorithms relies on several factors such as (i) easy implementation and dependency on simple approaches, (ii) easy to escape from the local optima or the ability of more exploration, (iii) problem independent, hence they can be used for complex optimization problems, particularly NP-Hard class problems, (iv) and they can be used in various science disciplines [76] – [78]. Nature-inspired algorithms are one of the well-known classes of metaheuristic algorithms that imitate the biological behavior of animals. These algorithms have been using various engineering optimization problems. Many nature-inspired metaheuristic algorithms have been introduced and this area of research is still open to coining new algorithms of this type. Some of the nature-inspired algorithms are very popular among researchers from various fields because by using these algorithms, most common optimization problems have been solved. Some well-known algorithms are PSO, GA, bat algorithm, ACO (Ant Colony Optimization) algorithm, DEO (Dolphin Echolocation) algorithm, GWO (Grey-Wolf Optimizer) algorithm, CS (Cuckoo Search) algorithm and honey bee algorithm [79] – [86]. All of these algorithms have gained incredible admiration from various field researchers because they offered effective solutions to various complex optimization problems.

3.5.1 Exploration and Exploitation

In a real-life situation, there is no single methodology or technique which stands without having a problem or issue related to it. Although metaheuristics approaches have achieved noticeable admiration among researchers for solving a complex problem in numerous fields, these algorithms severely suffer from exploration and exploitation problems. Exploration and exploitation problems such as diversification and intensification are associated with their searching capability concerning optimal solutions in the problem searching space. By definition, exploration or diversification describes the capability of searching tending towards the global searching domain, thus exploring a wide range of different places in search of the space of the problem. Intensification or exploitation is the searching ability of the algorithm to search more in its local searching domain to obtain a high-quality solution. In metaheuristic algorithms, there should be a balanced tradeoff with the effective utilization of the algorithm, generating optimal solutions. Nevertheless, preserving this tactical balance between exploration and exploitation is a complex challenge [87]. In simple words, metaheuristic algorithms are either searching more in the global searching domain and they do not find solutions in their local searching area (known as n exploration problem) or they are mostly searching in the local searching domain and thus do not explore or search well in the problem searching space globally, known as an exploitation problem. In any case (exploration or exploitation), the algorithm fails to maintain the balance between exploration and exploitation. Failing to maintain this tradeoff balance between exploration and exploitation severely affects the capabilities of the metaheuristic algorithm and the algorithm can go on to produce sub-optimal solutions to the optimization problem at hand. In other words, if the algorithm cannot manage this balance, then the algorithm becomes useless when it comes to providing optimal solutions to optimization problems [87] – [91]. Researchers in the field of metaheuristic algorithms understand this problem and therefore various studies have been conducted offering new metaheuristic algorithms, hybrid models and modified models of metaheuristic algorithms. All of these attempts have been made to solve exploration and exploitation problems in metaheuristic algorithms and then to apply them to real-world optimization problems.

3.6 Discussion

In the updated/modified versions of metaheuristic algorithms, the work typically equips algorithms with various mathematical operators and stochastic approaches to enhance the algorithm's efficiency. In this group of developments, local searches, chaotic maps and evolutionary operators are the common models that are used. Another popular solution is to hybridize various metaheuristic algorithms. Some popular hybrid models of metaheuristic algorithms are PSO-ACO, PSO-DE, PSO-GA, KH-BBO, KH-CS, ACO-DE, GA-DE, and ACO-GA [92] – [99]. Many researchers also take an interest in proposing new metaheuristic algorithms. However, new algorithms are also inspired by nature, physical rules, swarm intelligence, evolutionary phenomenon, and some biological behaviors. With the existence of plenty of new, modified, and hybrid algorithms, the query emerges as to whether there is still a research space for more metaheuristic algorithms and the answer lies in the (*NFL*) *No-Free-Lunch* theorem. The *NFL* theorem states that no metaheuristic algorithm can solve all optimization problems [100] – [101]. This means that a metaheuristic algorithm can produce an optimal solution for a specific optimization problem but it may fail to generate optimal values for other optimization problems. Thus we should use either popular metaheuristic algorithms for every kind of optimization problem or introduce new or modified algorithms to produce optimal solutions [101]. The *NFL* theorem and exploration and exploitation problems encourage researchers to conduct research into metaheuristic algorithms and how to apply them to various optimization problems.

Network optimization terminology is the efficient utilization of network resources without compromising the quality of services provided. However, the network can only be resourceful when the routes used for the data flow are computed optimally without compromising the various network quality requirements. MPLS networks are dependent on advanced and multiple network constraints and the objective functions used for route computation are formulated as multiple objectives subject to multiple constrained-based optimization problems. These are considered to be NP-hard class optimization problems [102] – [104]. For this class of optimization problems, metaheuristic approaches have become a popular choice over the last few years. Researchers have offered various optimization models for communication networks along with new, modified, or hybrid models of metaheuristic

algorithms. The situation not only relates to the optimization of the MPLS networks but also to the offering of improved or new versions of metaheuristic algorithms. The reason for offering new metaheuristic algorithms is related to the discussed exploration and exploitation problem. As discussed before, metaheuristic algorithms suffer from either exploration or exploitation problems that can generate a sub-optimal solution. Thus, having a metaheuristic algorithm that generates a sub-optimal solution for the NP-hard optimization of an MPLS network will not solve the optimization problem. In other words, less efficient algorithms cannot solve network optimization problems. Therefore the research question arises to offer a new or modified version of a metaheuristic algorithm that can balance its searching capability as exploration and exploitation and generate optimal solutions for the optimization model of the MPLS network.

Chapter 4

Pareto based Modified Local Global Particle Swarm Optimization Algorithm for MPLS Optimization

MPLS optimization is NP hard problem, for which metaheuristic algorithms will be the feasible approach to be used. There are numerous metaheuristics that can be applied for the mentioned optimization problem. However, the selection of metaheuristic principally depends on the algorithm popularity for various applications, its approach of dealing with exploration and exploitation, and success of generating optimal solutions. PSO algorithm is one of the well-known metaheuristic, which is used for many applications but still have poor convergence issue while applying to various applications.

In this chapter, the basic introduction of PSO is given along with its pseudo code. The exploration and exploitation problem will also be briefly explained. The state of the art literature review of PSO algorithm for network optimization will be the part of the chapter. Then, the problem formulation of MPLS network optimization is given along with objective functions. For the given optimization problem of MPLS, an algorithm is proposed along with flow-chart and pseudo code. In the last section of the chapter, the results will be discussed with conclusion.

4.1 Introduction to the PSO Algorithm

Metaheuristics can be categorized using several aspects; one of which is classifying them according to population-based algorithms. PSO belongs to the group of metaheuristic algorithms, introduced by Kennedy, and Eberhart , that depend on the methodology of swarms or population searching concerning the optimal solutions in the problem searching domain. The PSO algorithm stimulates the technique practiced by synchronized bird flocking, bees swarming, and fish

schooling. The algorithm uses the concept of a swarm or population, where the searching agents are known as particles. The particles share their current position with the other particles in the swarm using social behavior. During iterations, each particle in the swarm moves towards the new position and adjusts its position to the P_{best} (Particle Best Position) through cognitive behavior. The P_{best} is computed based on the condition that if the particle's new or current position is better than the previous position, then the current position will be updated as the particle best position. Otherwise, the previous position will be taken as P_{best}. Once all of the particles in the swarm update their P_{best}, then the particles' P_{best} positions are compared with each other using social behavior to compute the G_{best} (Global Best Position) during the current iteration [79]. Compared to other population-based algorithms, the PSO algorithm is governed by basic mathematical operators and it has less computational complexity as it can be implemented using fewer lines of code. The properties of PSO make it a resourceful algorithm concerning the aspects of memory and speed. The simplicity of its implementation and the ability of a quick and effective convergence make PSO a prevalent choice for various optimization problems [79], [105] – [108].

4.1.1 Basic PSO Algorithm

The origin of the PSO algorithm depends on the socio-cognitive behavior of the particle`s own experience and the knowledge of the successful particle in the swarm. In the optimization problem, there is n number of particles $(i = 1,2,3, \dots, n)$ in the swarm, which are initially positioned randomly in $n - dimensional$ searching space of the problem as candidate solutions. During the first iteration t, each particle i^{th} has flown towards its random position and velocity, indicated as $x_{i_{(t)}}$ and $v_{i_{(t)}}$, respectively. During the next iteration $t + 1$, each particle i^{th} in the population moves in the searching space and update its trajectory as updated $x_{i_{(t+1)}}$ and $v_{i_{(t+1)}}$. In the swarm, particles have fitness values that are measured by the fitness function to be optimized. During every single iteration $t + 1$ of the algorithm, each particle i^{th} compares its current position $x_{i_{(t+1)}}$ with its previous position $x_{i_{(t)}}$ using computed fitness values. Based on the judgment of the position that has optimal fitness value, the particle progress its best position so far as P_{best}. After achieving the P_{best} values from each particle, the best position value is obtained known as the global best position G_{best}, which represents the optimal position obtained

so far by any particle in the swarm. During the next iteration $t+1$, particles in the population update their position $x_{i_{(t+1)}}$ and velocity $v_{i_{(t+1)}}$ by using following mathematical expression as [79], [105] – [108];

$$v_{i_{(t+1)}} = v_{i_{(t)}} + c_{1*}\, r_1 \left(P_{best} - x_{i_{(t)}}\right) + c_{2*}\, r_2 \left(G_{best} - x_{i_{(t)}}\right) \tag{4.1}$$

$$x_{i_{(t+1)}} = v_{i_{(t+1)}} + x_{i_{(t)}} \tag{4.2}$$

where the acceleration constants c_1 and c_2 represent the cognitive and social behavior. Both c_1 and c_2 have constant values of 2, whereas r_1 and r_2 are the random numbers with the range of $[0, 1]$[79], [105] – [108].

By using the social learning ability, all particles move towards the global optimum G_{best}, while cognitive learning makes particle to keep its information [79], [105] – [108]. The pseudo-code of the basic PSO algorithm is given in *Algorithm* 4. 1.

Algorithm 4. 1: PSO Algorithm Pseudo code

---

1. **Input**
2. initialize number of searching agents as particles with random positions
3. initialize velocities of initial positions
4. compute particles position values using objective function
5. take initial random positions as P_{best}
6. compute G_{best} based on present P_{best}
7. set iteration $t = 1$
8. **Repeat**
9. compute the fitness function for each particle position
10. if fitness function for particle i position is optimal than its best fitness value so far, update P_{best}
11. compute the best position among the particles having optimal fitness value, as G_{best}
12. update each particle velocity according to Eq. (4.1)
13. update particle position based on Eq. (4.2)

14. update iteration $t = t + 1$

15. **Until** the stopping criteria met

16. **Output**

17. optimal solutions from the optimization problem domain

4.1.2 Balancing Exploration and Exploitation in the PSO Algorithm

In the PSO algorithm, all of the particles in the swarm rely on two basic parameters such as the self-learning (cognitive) and the social-interaction capability of the particle with the other particles during the collective iterations. For each iteration of the algorithm, each particle in the population discovers its optimal position compared to its previous best position, which is theoretically acknowledged as particle`s local best position or local optima. The optimal position of every particle $P_{best}$is compared with the other particles' P_{best} in the swarm to determine the global best position as the global optima. This implies that cognitive behavior in the algorithm is used to discover the local optima and that the social-interaction parameter is considered for the global optima. Following the next iteration, the global optimum is treated as a reference position to be followed by the other particles in the population [79], [108], [109].

All of the particles in the swarm consider G_{best} as their global optimum or the targeting position during the current iteration even though the G_{best} maybe so far from the optimal position. This problem usually occurs in multimodal optimization problems where particles can easily get trapped by the local optimum. Furthermore, if a suboptimal solution is obtained in an early stage of the algorithm, then all of the particles can easily fly around the suboptimal position without continuing the exploration in the searching space. One of the common problems that occur in the PSO algorithm is the trapping of the particle into its local optima. This is a condition in which the particle lies or follows the same position without an exploration of the searching space during iterations. This problem is acknowledged as a local optimal problem. On the flip side, there is a state where the particles concentrate more on exploration in the searching space, thus compromising the local search and thus producing repetitive or suboptimal solutions during

several iterations. It should be noted that this local and global optima problem discussion is also under investigation for other well-known optimization algorithms such as ACO and GA. For PSO, there is a challenging conversation among researchers to maintain the balance between local search (local optima) and global search (global optima) when applying the algorithm to various optimization problems. When dealing with multimodal optimization problems, PSO converges quickly towards a local search while missing out on the exploration of other search areas. For this problem, techniques like non-global best neighborhoods have been introduced that enhance the exploration capability of the algorithm but at the cost of a slow convergence towards the optimal position in the algorithm overall [108], [110] – [113].

According to Dong *et al.*,[114] the PSO is not a suitable algorithm for higher dimension complex optimization problems as it produces premature or suboptimal solutions because of its local optima problem. In addition to this, the researcher stated that with a lack of global exploration capability, the PSO algorithm is a poor choice for complicated and complex optimization problems. As a consequence, optimization algorithm engineers concentrate on the fundamental enhancement of the PSO algorithm in terms of its local and global optima problems to maintain the balance between explorations locally and globally in the searching domain. One of the common strategies used to improve the searching capability of the PSO algorithm is to enhance search space diversity. Despite these problems, this algorithm has been widely recognized as the superior choice for solving various real-valued optimization problems but with some modifications either in the form of fine-tuning its parameters or hybridizing it with other algorithms [115] – [120].

Several studies have revealed that PSO has inherited a global searching capability but its local search varies on a case by case basis [121]. To balance the local and global searching capability of PSO, the research in [122] presented a time decreasing factor modification to the algorithm. Recently, the hybridization model of PSO with other algorithms is a well-liked approach to enhance the algorithm's performance. Comprehensive learning particle swarm optimization (CLPSO) has been introduced by Liang *et al.* [113], in which the particles in the population preserve each of the particle`s best position and then utilize it for velocity updating. Parsopoulos and Vrahatis [123] suggested the modification in PSO in the form of a combination of local and global searches, called a unified particle swarm optimizer (UPSO). A fully informed version of PSO was presented by Mendes and Kennedy, in which all of the neighboring

particles update their velocities instead of updating their local and global positions [124]. In simple terms, the particle`s fitness function is dependent on its neighbor's size. The distance of a particle from its near neighborhood is an interaction technique is offered by Peram et al. [125] by introducing a FDR-PSO (Fitness Distance Ratio-based PSO) algorithm. There is a long list of literature available in which numerous researchers have developed different versions of enhanced or hybridized PSO to improve the PSO's performance while proposing solutions to the exploration and/ or exploitation problem [126] – [130]. Furthermore, the improvement of PSO for various topologies is also an active research area of interest.

In real communication networks, network optimization usually relies on MCOP. This is considered to be a complex optimization problem, designated as NP-hard [37] – [40], [104], [108]. For the given complex network optimization problems, metaheuristic algorithms are an appealing approach as they offer acceptable solutions with less computational complexity. The PSO algorithm in its enhanced form has been applied to various MCOP network optimization problems [131] – [135].

4.2 Literature Review

The optimal route computation in MPLS networks, which are modern multiservice based networks, is complex optimization model and thus, belongs to the group of NP hard computational complexity problem. Routing problem in MPLS networks include the selection of a sequence of optimum routes (satisfying network resources), which seek the optimization of multiple objectives having multiple constraints. Formulation of routing problems in MPLS networks has potential advantages, as the trade-off between conflicted objectives functions can be analyzed in a consistent manner. Formulation for multi-objectives is also considered as global routing problem, in which simultaneous LSPs computation is taken [131] – [135]. In the context of MPLS networks, the stochastic multi-objectives optimization model was proposed by Craveirinha *et al* [136]. This was the meta-model for routing formulation in MPLS networks in which hierarchical multiple objectives were considered. In addition to this, a specialized metaheuristic strategy for computing optimal solutions related to multiple objectives was proposed by *Girao-Silva et al* [137]. A comprehensive review of multi-objectives based routing optimization models for modern communication networks are presented by Climaco *et al* [138].

Several classical exact methods are presented for multi-objective optimization models in [139] – [140]. However, exact techniques are too costly in terms of execution time and memory requirements. Recently, metaheuristics have got momentous attention for multi-objective optimization problems. For modern networks optimization problem, there are a vast number of bibliographical references for heuristic and metaheuristics. For instance, Kim *et al*. [141] modified simulated annealing algorithm for the allocation of nominal channels to the cells in the cellular system in which the objective functions was to minimize the average call blocking in the network. The transportation cost of the traffic flow across the network was done by utilizing simulated annealing in [142]. Two methods of greedy procedure and table search algorithm are developed by Amaldi *et al*. [143], in which the maximization of traffic covered by base station and installation costs are minimized. The optimization model of least-cost communication network was proposed by Cox and Sanchez [144], in which the network capacity according to traffic demand was maximized. Fink *et al*. [145] presented optimization model for ring network design problems by using simulated annealing and tabu search algorithms. Gendron *et al*. [146] proposed a heuristic technique for a network load balancing problem. For discrete cost multi-commodity network optimization problem, a metaheuristic optimization technique is offered by Gabrel *et al* [147]. Resend and Ribeiro [148] presented a group of heuristics algorithms for routing problem in private virtual communication networks. For route traffic on self-healing links (stack ring design problem), the genetic algorithm as metaheuristic approach was implemented by Armony et al. [149], in which the objective functions was to minimize the trade-off between equipment costs and traffic flow cost in ring topology. Hybrid genetic algorithm is presented by Buriol et al. [150] for solving the OSPF (Open Shortest Path First) weight setting problem in MPLS networks. Ant colony optimization algorithm was applied by Varela and Sinclair [151] for the routing and wavelength allocation problem in virtual wavelength path routing in transport networks. Wittner *et al*. [152] presented swarm intelligence based metaheuristic for the resource reservation optimization problem in communication networks. The modified version of PSO algorithm as parallel PSO is presented by Bo Li *et al*. [153] for improving the communication networks latency. PSO based multiuser detector for CDMA (Code Division Multiple Access) communication system was presented by Soo *et al.* [154], in which the modified PSO is applied for detecting the received data bit by optimizing an objective

function incorporating the linear system of the LMMSE (linear minimum mean square error) detector.

It is important to note here that all these metaheuristic algorithms are applied to communication networks optimization in their modified versions. In other words, the basic algorithms for each metaheuristic suffers with exploitation and/ or exploration and poor convergence problem while applying to NP hard base optimization problems. There are vast amount of literature available for the enhanced versions of each metaheuristic, but PSO is one of the famous metaheuristic algorithm. The reputation of PSO resides in its less complex parameters, easy implementation, less computational cost and flexibility for optimization problems for various applications. However, PSO severely suffers with premature convergence due to its exploration and/ or exploitation problem. Numerous attempts have been taken for solving mentioned problem in PSO. For instance, A. Rezaee Jordehi [155] presented ELPSO (Enhanced Leader PSO), which is manly based on five-staged successive mutation strategy for solving traditional PSO premature convergence problem. Khan *et al*. [156] proposed a GPSO (Global PSO) for the optimization of electromagnetic design problem. Yasir Mehmood [157] proposed ESPSO (Exploration Strategy PSO), which enhanced the exploitation capability of the basic PSO and the effectiveness of the proposed algorithm was proved over benchmark optimization function. A-PSO (Ameliorated PSO) is presented by Ke Chen *et al*. [158], in which the different strategy is applied for position and velocity updating in PSO and the algorithm performance is verified over 20 benchmark functions. Ekkarat *et al*. [159] presented the improved version of PSO in which the mechanism of global best position is modified and the algorithm is tested over 27 benchmark functions. A comprehensive survey is presented by Yasir mehmood *et al*. [160] for clustering-based modified PSO techniques to solve multimodal optimization problem. PSO in its enhanced forms are applied for various modern communication networks $i.e,$ a novel distributed and dynamic version of PSO was proposed for UAVs (Unmanned Aerial Vehicles) in [161]. PSO used for shortest path problem in communication networks literature is presented in [162]. For MPLS based optical networks, a modified version of PSO was proposed by Yu et al. [163], in which the author presented the global optimization model to achieve the optimal traffic node distribution.

4.3 Problem Formulation

Several studies made by researchers for the PSO algorithm revealed that the application of P_{best} is superior compare to G_{best}. This conveys that the P_{best} as local best has better exploration capability and enhance algorithm performance in terms of improved early convergence. On the contrary, the PSO version that more tends towards the G_{best} the model supports for unimodal optimization problems but can easily be trapped into local optima problem while dealing with multi-modal optimization problems. For multimodal optimization problems, P_{best} supported versions of PSO can be used as a better choice but offers poor convergence rate for unimodal optimization problems [108], -[164].

To address the above, this chapter proposes a modified version of PSO which has the support of both local-best and the global-best versions of the PSO algorithm. The method is entitled *"PMLG-PSO* (*Pareto based Modified Local Global Particle Swarm Optimization*)*"* algorithm embedded with the Pareto approach for solving the MPLS optimization problem. In the following sections, the optimization problem of the MPLS network is formulated in the form of presented multi-objective functions.

4.3.1 Objective Functions for MPLS Optimization

In recent days, network vendors and internet service providers prefer to deploy MPLS based routers in the networks. In the optimization model, the MPLS network is represented as a graph by using graph theory, where nodes/ routers are vertices and links signify edges. A route is a computed path between the LERs. Each path indicates the sequentially connected links between intermediate LSRs. Therefore, the MPLS network can be represented by *G*, which indicates the graph/ network. The set of routers (vertices) are denoted by R_{Set}, where each router r is the member of R_{Set} and L_{Set} is the set of connected links l. Each link has a fixed capacity for traffic denoted as l_c. In the experiments, several traffic requests will be arriving at ingress LSR, which computes the optimal path (s) based on the proposed algorithm between ingress and the egress LSRs.

In the experimental setup for the optimization model of MPLS networks; two objective functions are considered though out the optimization approach, such as; the optimal resource

provisioning and the load balancing within the constructed network along with constraints, since it is MCOP problem. The following subsection is a detailed explanation of the mentioned objective functions.

4.3.1.1 Resource Reservation Objective Function

In the experimental setup, the MPLS domain-based network is designed, which receives traffic demands at ingress LSR of the network. The objective function for the routing algorithm is to compute the optimal path(s), where resources can be provisioned optimally without compromising on QoS, to be granted to end-users. In the network, each link l has assigned a specific cost per unit of traffic flow symbolized as $C_{l,r}$, which is assigned by the service provider. Whereas C_e^p is the total routing cost of a path p for traffic requests over a selected link l. The bandwidth price for the connection between each router of the network is p_r^b and thus, the bandwidth B cost is $B \times p_r^b$ over the link l. Similarly, there is also a delay d_e on each link l for forwarding the data traffic in the network. Let x_{ij} be used as an indicator to show whether the link l is selected for forwarding traffic from i to j. $i_{e,l}^p$ used as an indicator that shows the traffic flow over the path; which will be 1 if there is a traffic flow.

$$X_{ij} = \begin{cases} 1, & if\ the\ link\ is\ selected \\ 0, & otherwise \end{cases} \tag{4.3}$$

$$Cost_{resource} = Cost_{bandwidth} + Cost_{delay} + Cost_{Routing} \tag{4.4}$$

$$Cost_{bandwidth} = \alpha \left(\sum_{l \in L_{set}} C_e i_{e,l}^p \right) \tag{4.5}$$

$$Cost_{delay} = \beta \left(\sum_{l \in L_{set}} d_e \right) \tag{4.6}$$

$$Cost_{Routing} = \gamma \left(\sum_{l \in L_{set}} C_{l,r} i_{e,l}^{p} \right) \tag{4.7}$$

$$\textbf{\textit{Objective Function}}: \quad min\,imization(Cost_{resource}) \tag{4.8}$$

Subject to following constraints;

$$\sum_{l \in L_{set}} x_{ij} = 1 \qquad for \quad x_e^p > 0 \qquad \text{Constraint (4.8a)}$$

$$\sum_{l \in L_{set}} x_{ij} = 1 \qquad for \quad i_{e,l}^{p} = 1 \qquad \text{Constraint (4.8b)}$$

$$\sum_{l \in L_{set}} x_e^p = v_{traffic} \qquad \text{Constraint (4.8c)}$$

$$\sum_{l}^{L_{set}} p_r^b \qquad where \quad p_r^b > 0, \quad p_r^b = \{10, 100\} \qquad \text{Constraint (4.8d)}$$

$$d_e > 0 \qquad for \quad \forall\, l \in L_{set} \qquad \text{Constraint (4.8e)}$$

$C_{l,r}$ is the total routing cost of a path p for traffic requests over a selected link l; $\alpha, \beta,$ and γ are the positive constants that are used to indicate the weight of each component. The objective function for resource provisioning problem is given by Eq. (4.8); Constraint (4.8a) ensures that all the links belonging to a set of links which can only be selected if $x_{ij} = 1$, and when the variable x_e^p for the selection of the path value is greater than 0. From Constraint (4.8b), the constraint identifies that the link l from L_{Set} can only be chosen if link l is used for the traffic request. Constraint (4.8c) verifies that all selected links for the computed path must accommodate all number of traffic requests in the given topology. Constraint (4.8d) as verification of bandwidth price is added on each selected link within limited bounds of $[10, 100]$. Constraint (4.8e) recognizes that traffic delay d_e must be present to all links.

4.3.1.2 Traffic Load Balancing Objective Function

There are many choices to accomplish the traffic load balancing metric and implement a piecewise approximation of a cost function as described by Fortz and Thorup . Each connecting link between routers in the network is associated with two parameters known as a link load (link utilization) l_u and a link capacity l_c. The Link utilization l_u represents traffic flows over the link. While the link capacity l_c can be defined as a capacity of the link to handle unit traffic. By having these two functions, the traffic load balancing φ_l the cost can be illustrated as the link utilization according to its capacity as following;

$$\varphi = \frac{link\ utilization\ (l_u)}{link\ capacity\ (l_c)} \tag{4.9}$$

To achieve a resourceful use of links for traffic flows in the network, the links load balancing can be used as an objective function that is to be minimized. This approach leads to efficient utilization of network resources as well as to minimum packets loss, bound delays, and jitter, etc. Consequently, this objective function can be described to minimize the sum of traffic load balancing costs for all the links in the network as demonstrated by Eq. (4.10).

$$\boldsymbol{Objective\ Function}: minimization\left(\sum_{l \in L_{set}} \varphi_l\right) \tag{4.10}$$

Subject to constraints:

$$l_u \leq l_c \qquad \text{Constraint (4.10a)}$$

$$\varphi_l \geq l_u \quad \forall \ \ l \in L_{total} \qquad \text{Constraint (4.10b)}$$

$$l_u = \sum_{e \in E} \sum_{l=1}^{L_{set}} i_e^p x_e^p \qquad \text{Constraint (4.10c)}$$

$$\varphi_l \geq 2.2\, l_u - 0.52\, l_c \qquad \text{Constraint (4.10d)}$$

$$\varphi_l \geq 5.1\, l_u - 2.33\, l_c \qquad \text{Constraint (4.10e)}$$

$$\varphi_l \geq 15.2\, l_u - 9.32\, l_c \qquad \text{Constraint (4.10f)}$$

$$\varphi_l \geq 60.2\, l_u - 45.35\, l_c \qquad \text{Constraint (4.10g)}$$

$$\varphi_l \geq 300.3\, l_u - 261.34\, l_c \qquad \text{Constraint (4.10h)}$$

$$x_e^p > 0 \quad \forall \quad e \in E \quad and \quad l \in L_{set} \qquad \text{Constraint (4.10i)}$$

$$r \in R_{set} \quad for \quad v_{traffic} \qquad \text{Constraint (4.10j)}$$

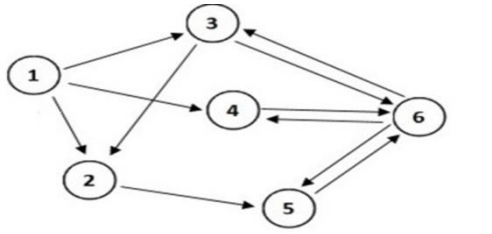
$$l > 0 \quad for \ \{r, r^*\} \qquad \text{Constraint (4.10k)}$$

Constraint (4.10a) ensures that the link utilization l_u in the network must have either the same or less than the link's capacity l_c. In modest wordings, the traffic flows over the link must not exceed the link's capacity. Constraint (4.10b) states that traffic load balancing φ_l must be greater than or equal to the link utilization l_u for all links. Further, this constraint guarantees that difference between the link utilization l_u and link capacity l_c must not be very high. Constraint (4.10c) precisely explains features of the link utilization l_u in the given topology. Link l can only be considered as utilized for a traffic flow, when the traffic flow is the member of the set of traffic demands, and the link l is also the member of the set in admissible links of the network. While x_e^p is the indicator which identifies the link l can be used for traffic flow over a path, such that a link l can be used for path computation. Constraint (4.10d) to Constraint (4.10h) define some specific linear load balancing cost function within a feasible region, in which the numbers multiplying with l_u and l_c define the constraints taken within the lower and upper bound of the searching domain.. Constraint (4.10i) represents the identifier of traffic flow over a path when it is more than 0 value. In other words, a path can be used for a traffic request. Moreover,

Constraint (4.10j) and Constraint (4.10k) identify the connected links and routers in the topology, which are used for the requested path.

4.4 Particle Encoding

A Particle encoding is the first step towards the further evaluation of the proposed algorithm. In the MPLS optimization network model, the particles travel over the two-dimensional matrix, where each row represents a *"from-node"* and each column describes the *"to-node"*, in a searching matrix. Each particle represents the traffic flow over each selected link for a given traffic request. To elaborate on this section, let us consider a 6-nodes network as shown in Fig. 4. 1, the adjacency and demand matrices, as shown in Fig. 4. 2 and Fig. 4. 3. In the given example of six nodes network (from node 1 to node 6)three traffic demands (packets per unit sec) are considered as $N_1 = (1, 6, 30), N_2 = (3, 5, 10)$, and $N_3 = (1, 4, 20)$. For example, in $N_1 = (1, 6, 30)$ case, the traffic flow must be from node 1 to node 6 with 30 packets. For the first traffic demand t_1, the traffic will be split over three paths $i.e.$, first path as (1, 3, 6), 2nd path (1, 4, 6) and (1, 2, 5, 6) as 3rd path. This means, for t_1 the traffic flow is from node 1 to node 6 via node 3, for t_2 the traffic flow is from node 1 to 6 via node 4 and similarly, for t_3 the packets will flow from node 1 to 6 through node 2 and node 5. For N_2, traffic splits over two paths such as $path_1: (3, 2, 5)$, and $path_2: (3, 6, 5)$. Similarly for N_3, the traffic will be split over two equal paths as $path_1: (1, 4)$ and $path_2: (1, 3, 6, 4)$. This can be seen in Fig. 4. 3, for example in the case of $N_1 = (1, 6, 30)$, the traffic of total 30 packets splits into 10 packets from node 1 to node 2 and node 3, from node 2 to node 5, node 3 to 6, node 4 to 6 and node 5 to 6.



$Network = (source, destination, traffic)$

Fig. 4. 1: Network Model

From Node \ To Node	1	2	3	4	5	6
1		1	1	1		
2					1	
3		1				1
4						1
5						1
6			1	1	1	

Fig. 4. 1: Adjacency Matrix for Traffic Model

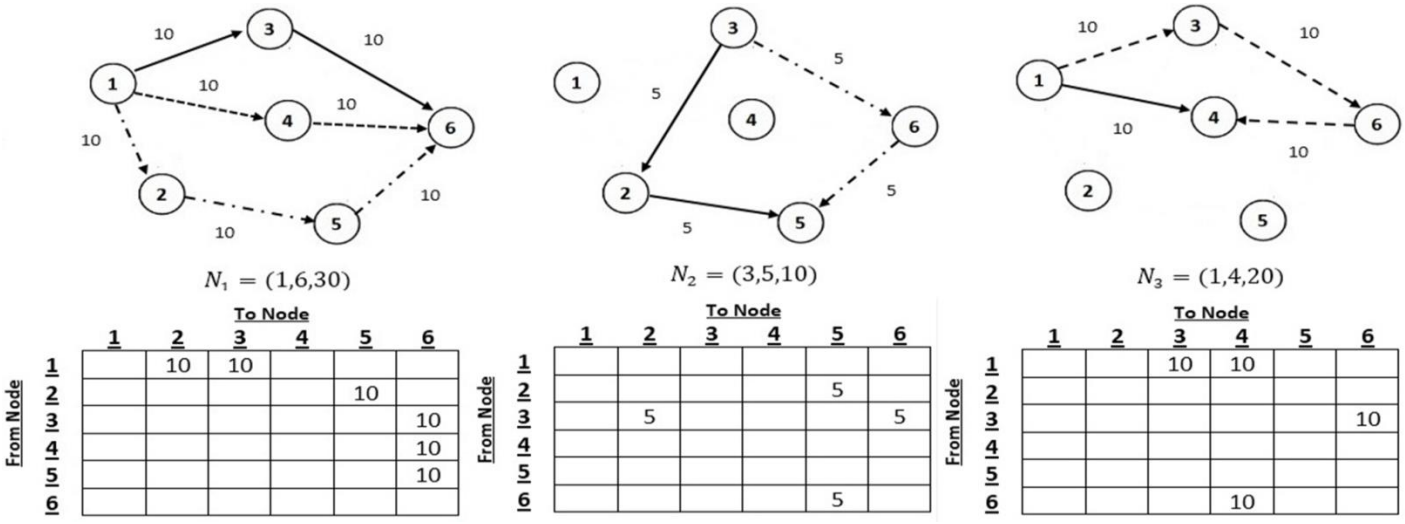


Fig. 4. 2: Demand Matrices

4.5 Proposed *PMLG-PSO* Approach for Network Optimization

This section proposes a novel meta-heuristic approach, entitled *PMLG-PSO* algorithm for the MCOP optimization problem in MPLS networks. The flow chart of the proposed *PMLG-PSO* algorithm is shown in Fig. 4. 4. The flow chart is segmented into three main phases; initialization phase, processing phase, and post-processing phase. Each of the phases will be described with pseudo code along with an explanation in the following subsections.

4.5.1 Initial Phase

This phase discusses the initial stage or pre-processing stage of the algorithm, where the searching agents (particles), searching space (matrix/ graph), and the algorithm's initial operations are engaged. In the pre-processing stage, the constraints have been applied to limit the searching space and as a result, obtain a skeleton matrix. A Skeleton matrix is used as a searching constrained dependent graph, where particles can hunt for feasible solutions. In the experiments, the skeleton matrix is applied as an MPLS network that is the collection of permissible links between connected nodes and can be used for traffic requests. Furthermore, the particles are initialized with random initial positions in the searching matrix for both fitness functions (resource provisioning and traffic load balancing). This section explores the case for a resource provisioning fitness function along with its constraints, where the random initial positions of particles are loaded as $init_1$.

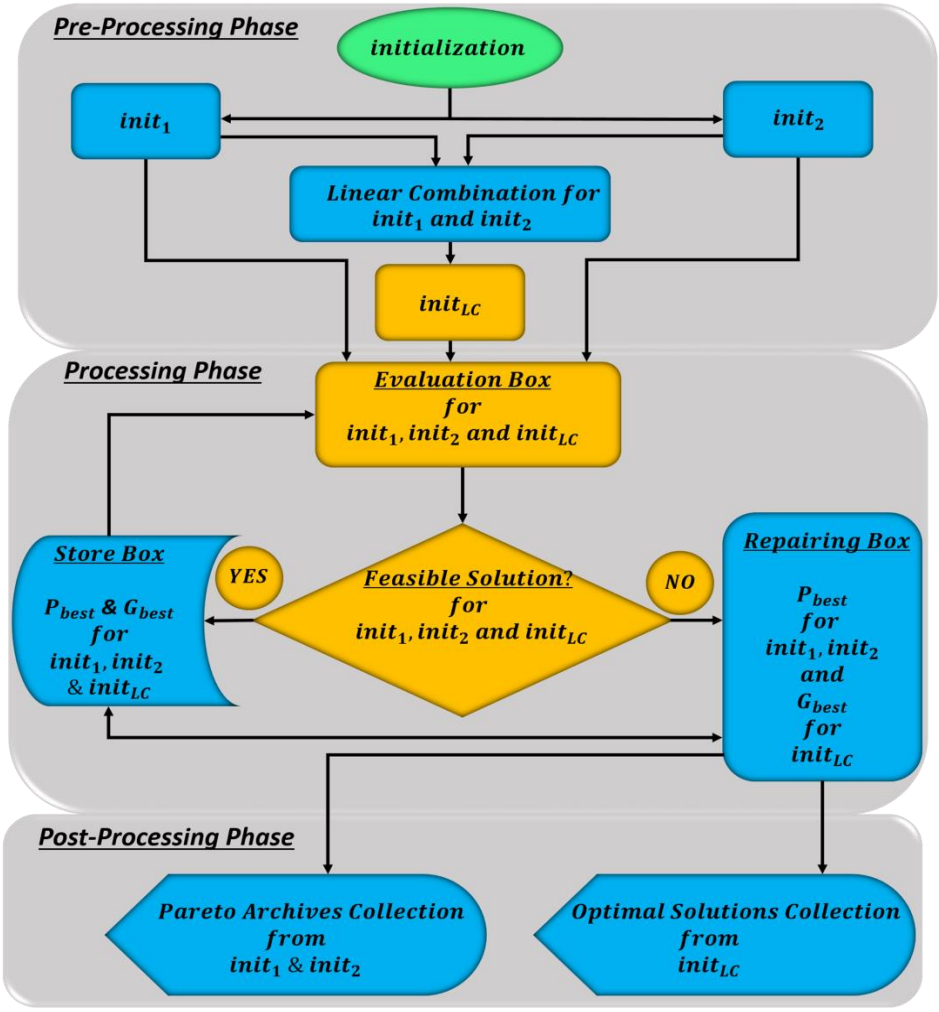


Fig. 4. 3: Proposed PMLG-PSO Algorithm Flow Chart

Whereas, for traffic load balancing fitness functions with its constraints, the initial particle positions are loaded in $init_2$.

A Particle best position for a resource provision function is P_{best_r}. The particle best position for a load balancing function is P_{best_l}. Global-best-positions G_{best_r} and G_{best_l} that is achieved by P_{best} (P_{best_r} and P_{best_l}) group. Solutions for (G_{best_r} and G_{best_l}) are collected as non-dominant solutions from both objective functions for generating the graph of Pareto front. However, the algorithm's optimal solutions are obtained as global-best-positions G_{best} from the foundation of the linear combination fitness function $init_{LC}$. According to [108], [165]– [166], for optimal solutions collection various tactics can be used for MCOP optimization problems. They are criteria-based, aggregation-based, Pareto-based dominant ranking, and linear

combination-based methods. In this chapter, the parametric linear combination of fitness functions for the computation of optimal solutions G_{best}) is applied, which can mathematically derive as;

$$Fitness\ Function_{LC} = \alpha\ (Cost_{resource}) + (1 - \alpha) \left(\sum_{l \in L_{set}} \varphi_l \right) + \Phi \tag{4.11}$$

In Eq. (4.11), α is a weight variable and Φ is plenty. G_{best} of the linear combination fitness function ($Fitness\ Function_{LC}\ or\ init_{LC}$) will be considered as the optimal solution. The pseudo-code for the initialization phase is given below as *Algorithm* 4. 2.

Algorithm 4. 2: Initialization Stage of *PMLG-PSO* Algorithm

INPUT:

1: initialize number of particles N in the population with respect to objective functions $f_{x_{i_r}}$ and $f_{x_{i_l}}$

2: w : initialize weight inertia parameter $f_{x_{i_r}}$ and $f_{x_{i_l}}$ with the value of 2

3: c_1 , c_2 : initialize with the initial values of 2 for both

REPEAT

4: ***for*** $i = 1{:}\,N$ population ***do***

5: x_{i_r} : initialize for $init_1$ subject to constraints using Eq. (4. 1)

6: x_{i_l} : initialize for $init_2$ subject to constraints using Eq. (4. 1)

7: v_{i_r} : initialize for $init_1$ using Eq. (4. 2)

8: v_{i_l} : initialize for $init_2$ using Eq. (4. 2)

9: $f_{x_{i_r}}$: compute $fitness\ function_{Cost_{resource}}$ according to Eq. (4. 8)

10: $f_{x_{i_l}}$: compute $fitness\ function_{Cost_{traffic\ load\ balancing}}$ according to Eq. (4. 10)

11: P_{best_r} : update particle best position based on $f_{x_{i_r}}$

12: P_{best_l} : update particle best position based on $f_{x_{i_l}}$

13: ***end***

14: G_{best_r} : update global best position for $f_{x_{i_r}}$

15: G_{best_l} : update global best position for $f_{x_{i_l}}$

16: ***for*** $i = 1: N$ population ***do***

17: compute $fitness\ function_{LC}$ using Eq. (4. 11)

18: $P_{best_{LC}}$: update particle best position based on $fitness\ function_{LC}$

19: ***end***

OUTPUT

20: $G_{best_{LC}}$: update global best position for $fitness\ function_{LC}$

4.5.2 Processing Phase

The processing phase contains all those key components which help the proposed *PMLG-PSO* algorithm to resolve the traditional PSO algorithm's discussed problems and thus, discover the optimal solutions for MCOP optimization in the MPLS network. During the processing stage, the algorithm's traditional parameters such as w, c_1 and c_2 will contribute to the algorithm's convergence activity for $init_1, init_2$ and $init_{LC}$. The updated positions and velocities for $init_1, init_2$ and $init_{LC}$ are evaluated concerning their fitness functions separately. The feasible solutions are referred to here as feasible paths in the network. A condition is applied that if the updated fitness function used for path computation is optimal than the previous solution, then update the particle's position (solution) as the particle's best solution P_{best}. This exercise occurs in $init_1, init_2$ as P_{best_r} and P_{best_l} respectively. This step ensures that the updated position is better than the previous position. If the latest position of the particle is not optimal then particle keeps the previous position as its best position and from P_{best} positions in the updated swarm choose the global best solution G_{best}. The algorithm uses ESE (Evolutionary State Estimation) technique, which is dependent on the particle's distribution information in the swarm. The evolutionary factor is developed on the fuzzy classification method and is directed

by evolutionary factor f . The steps that have been followed for ESE terminology can be described as following [108],[167]:

1. Find each particle's mean distance d_i from all other particles in the swarm, using the following expression;

$$d_i = \frac{1}{N-1} \sum_{j=1, j \neq x_i}^{N} \sqrt{\sum_{k=1}^{D} \left(x_i^k - x_j^k\right)^2} \tag{4.12}$$

where j =another particle position from x_i particle, $k =$ each dimension in the matrix, and $D =$ total dimensions in the matrix.

2. Compare all computed distances (for each particle) for both $init_1$ and $init_2$ matrices and define the maximum distance d_{max}, minimum distance d_{min}, and distance of the computed global best position of the particle d_g.

3. Now, evaluate the evolutionary factor f by the following expression;

$$f = \frac{d_g - d_{min}}{d_{max} - d_{min}} \in [0\,,1] \tag{4.13}$$

4. After computing evolutionary factor f, calculate the w as follow

$$w = \frac{1}{(1 + 1.5\, e^{-2.6f})} \in [0.4\,,0.9]\ ,\forall\ f\ [0\,,1] \tag{4.14}$$

5. Evaluate c_1 and c_1 with the following expression;

$$c_i^{t+1} = c_i^t \pm \delta \tag{4.15}$$

Where $i = 1\,,2$ for c_1 and c_2, but c_1, c_2 values must lie within the range of [1.5: 2.5] which decrements during each iteration. An extensive research has been conducted for specifying the values range for w and for c_1, c_2, and it is observed that these variables give support the algorithm`s convergence when having these values [108]. APSO (Adaptive PSO) used ELS (Elitist Learning Strategy) method to target the global optima problem in the PSO technique. In APSO based ELS model the random dimension, from the particle's global historical best positions, is selected and denoted as p^d. Thereafter, find the maximum dimension

value x_{max}^{d} and minimum dimension value x_{min}^{d} from the dimension ranges. From the values, compute the dimension value P^{D} based on the following equation:

$$P^{D} = p^{d} + \left(x_{max}^{d} - x_{min}^{d}\right) Gaussian\,(\mu, \sigma^{2}) \tag{4.16}$$

Where $= \sigma_{max}\,(\sigma_{max} - \sigma_{min})$, $\sigma_{max} = 1$ and $\sigma_{min} = 0.1$. $Gaussian$ represents here as Gaussian distribution with mean value of $\mu = 0$ and σ is the time varying standard deviation value. σ_{max} and σ_{min} values based on empirical studies made in [108], [167].

Regardless of the use of an ELS model in APSO, the algorithm can still suffer intense exploration problem, as the local-optima problem still exist during number of iterations. Such as, if only global optima problem is supervised with the assistance of ELS model (as proposed in [167]), then the local optima problem can still occur. This situation disturbs the overall performance of the PSO algorithm with poor convergence and thus, generates sub-optimal solutions, which will be monitored in the results section. Furthermore, during iterations the increasing number of particles undergoing local optima may results the ELS technique as ineffective model. In simple words, the ELS model that targets only global optima problem, does not guarantee the provision of optimal solutions. The research contribution work presented in this paper proposes an algorithm that targets both local and global optima problems simultaneously, and to certify the algorithm's rationality on real-world application, and is applied for MCOP optimization in MPLS networks. The pseudo code for processing stage of *PMLG-PSO algorithm* is given as *Algorithm* 4. 3: Pseudo code for Processing Stage of *PMLG-PSO* Algorithm.

Algorithm 4. 3: Pseudo code for Processing Stage of *PMLG-PSO* Algorithm

INPUT:

1: ***for*** objective functions $f_{x_{i_r}}$ and $f_{x_{i_l}}$

REPEAT:

2: ***for*** $i = 1: N$ population ***do***
3: ***if*** $f_{xi_{present}} < f_{xi_{previous}}$
4: update $P_{best} : x_{i_r}, x_{i_l} \rightarrow P_{best_r}, P_{best_l}$
5: store p^d register (P_{best}) = empty
6: store P^D register (P_{best}) = empty
7: ***else***
8: $P_{best_{previous}}$ = update $P_{best} \rightarrow P_{best_r}, P_{best_l}$
9: update $P_{best} \rightarrow Repairing\ Box$
10: $Repairing\ Box$ $(AELS)$ method for P_{best}
11: ***if*** $p^d = p^d_{previous}$ and $P^D = P^D_{previous}$ ***do***
12: initialize random p^d from P_{best} dimensions
13: initialize x^d_{max} and x^d_{min} from P_{best} dimensions
14: compute P^D using Eq. (4.16)
15: ***else***
16: initialize random p^d from p^d Register
17: initialize random P^D from P^D Register
18: ***end if***
19: update p^d register (P_{best})

update P^D register (P_{best})

update $P_{best} \rightarrow P_{best_r}$, P_{best_l} register

end if

stored $P_{best} \rightarrow P_{best_r}, P_{best_l}$

update $G_{best} \rightarrow G_{best_r}, G_{best_l}$

update G_{best} as $P_{best}\left(P_{best_r}, P_{best_l}\right) \rightarrow G_{best}\left(G_{best_r}, G_{best_l}\right)$

if fitness function of $G_{best_{present}} < G_{best_{previous}}$ ***do***

update $G_{best}\left(G_{best_r}, G_{best_l}\right)$

store $G_{best}\left(G_{best_r}, G_{best_l}\right)$ in G_{best} Register

store $p^d\ (G_{best})$ = empty

store $P^D\ (G_{best})$ = empty

else

evaluate $G_{best} \rightarrow$ Repairing Box

Repairing Box ($AELS$ method for G_{best})

if $p^d = p^d_{previous}$ and $P^D = P^D_{previous}$ ***do***

initialize random p^d from G_{best} dimensions

initialize x^d_{max} and x^d_{min} from G_{best} dimensions

compute P^D using Eq. (4.16)

else

initialize random p^d from G_{best} Register

initialize x^d_{max} and x^d_{min} from G_{best} dimensions

compute P^D using Eq. (4.16)

end if

update p^d register for G_{best}

update P^D register for G_{best}

update $G_{best}\left(G_{best_r}, G_{best_l}\right)$ register

end if

end for

stored $G_{best}\left(G_{best_r}, G_{best_l}\right)$ register

end for

OUTPUT:

Optimal Solutions for $f_{x_{i_r}}$ and $f_{x_{i_l}}$

4.5.2.1 Registers used in *PMLG-PSO* Algorithm

In the proposed *PMLG-PSO* algorithm, a modified model of ELS is offered, which is loaded with so called *"Repairing Box"*. Once the algorithm identifies the exploration problem at any stage, it goes to its repairing box (implanted with six registers). The repairing box is used to drag out the particle or group of particles which is either stacked in its local optima or its global optima. This repairing process of the local/ global optima problem within the repairing box is briefly explained in the pseudo code along with description. However, before solving the problem, it is important to vigilantly detect the local/optima problem. Having such goal, six registers are implanted into the algorithm. Each register is specifically used for the detection of local and global optima problem during different stages of the algorithm. The purposes and names of the registers are mentioned in Table 4. 1.

Table 4. 1: Registers used for the detection of exploration problem

Register Names		Registers Working
i. P_{best} $Register$	→	Register for each particle's local best position(P_{best})
ii. P_{best} : p^d $Register$	→	Register of P_{best} $random$ p^d values
iii. P_{best} : P^D $Register$	→	Register for P_{best} P^D values
iv. G_{best} $Register$	→	Storing register for global best position(G_{best})
v. G_{best} : p^d $Register$	→	Register of global best position p^d values
vi. G_{best} : P^D $Register$	→	Stored register for G_{best} P^D values

Registers presented in Table 4. 1 worked as a memory registers that store previous values thus help the algorithm to spot the exploration problem. For instance, during algorithm's implementation, each particle i in the swarm keeps storing its P_{best} values in the P_{best} register. If particle`s best position P_{best} fell into its local optima, then P_{best} register identifies this problem with the support of its repetitive value/ position for P_{best} . In addition to this, it is also possible that those P_{best}: p^d register and P_{best}: P^D register may also help to detect the local optima problem. Similar situations can be reflected in G_{best} register, G_{best}: p^d register and G_{best}: P^D register for the identification of global optima problem during algorithm's processing stages. Once the local and/ or global optima problem is identified at any stage with the help of aforementioned registers, and then the algorithm jumps towards the repairing box. Repairing box contains the modified version of ELS model and is entitled as *AELS* (*Adaptive Elitist learning Strategy*) model. In the repairing box, the offered *AELS* model will be used to fix the local/ global optima problem.

For example, if the P_{best} register discovered a local optima problem, this information will be send to repairing box in which this issue will be resolved with the support of *AELS* model. If

in some way the $P_{best}: p^d$ register couldn't resolve the problem in its 1st iteration and records still show the local optima problem, then the proposed *AELS* technique will further choose another random p^dvalue and then computes the final P^D value. And, if the problem still occurs, the algorithm will repeat the random replacement of other p^d values until the condition is not satisfied. In worst-case scenario, when p^d can also be trapped with repetitive fixed value, then the new p^dwill be fetched from P_{best} register. Global optima problems can be exposed in G_{best}, $G_{best}: p^d$ and $G_{best}: P^D$ registers and henceforth, can be resolved with the help of *AELS* model in the repairing box. Once the exploration and / or exploitation problem is resolved, the updated solution is sent back to the processing stage of the algorithm.

Similar procedure of the proposed repairing box (along with *AELS*) and the algorithm will be followed for the linear combination function $init_{LC}$. If there is no local/ global optima problem raised, then Pareto based feasible solutions are stored the in the appropriate storage boxes of $init_1$, $init_2$ and optimal solutions in $init_{LC}$. The pseudo code in the processing stage of the *PMLG-PSO* algorithm is given as follow.

4.5.3 Post processing Phase

In the algorithm, two separate initializations $init_1$ and $init_2$ are used to collect non-dominated solutions, in order to create the archive for the Pareto front. Whereas the algorithm`s optimal solutions are stored in the form of G_{best} solutions from $init_{LC}$. These optimal solutions are a collection of connected links (paths). The G_{best} solutions are preserved in the form of an archive and stored in the G_{best} storing box, as a number of available optimal solutions, as it is shown in *Algorithm* 4. 5.

Algorithm 4. 4: Pseudo code for Post Processing Stage of the *PMLG-PSO* Algorithm

---

Input : G_{best} from objective functions $f_{x_{i_r}}$ and $f_{x_{i_l}}$

Output : Optimal path (s) for the traffic requests

1: ***for*** $i = 1$: maximum iterations for G_{best} ***do***

2: ***for*** $i = 1 : N$ ***do***

3: G_{best} representing solutions of the problem

4: select link with minimum (optimal) value in G_{best} matrix

5: initialize path from the selected links

6: path(s) for requested traffic

7: ***end for***

8: achieve the selected links as optimal

9: ***end for***

4.6 Empirical Assessment

To assess the effectiveness of the proposed *PMLG-PSO* algorithm, the two brief performance analysis investigations are made. The first evaluation appears in the outcomes of solving the MCOP optimization problem in MPLS networks by using the Pareto Front graphs. The second investigation is a comparative study analysis; where the offered algorithm performance is compared against renowned metaheuristic algorithms such as PSO, APSO, Bat, and DA (Dolphin Algorithm) [108]. Both researches focus on an algorithm convergence ratio (particularly the trapping in local/ global optima problem domain) to test how successfully the proposed algorithm fixed the exploration problem when compared to the other optimization algorithms. The MPLS network-based topology is used for the experiments, where objective functions; resource provisioning costs and traffic load balancing costs are considered, as described in the problem formulation section. Demonstration of the conducted experiments along with results breakdown are discussed in the proceeding sections.

4.6.1 Experiments for Pareto Front and Result Analysis

For conducting simulations, the *MATLAB R2016a* package is used to develop the Pareto based solutions as well as optimal solutions based on described algorithms. The experiments were scheduled in such a manner that the proposed *PMLG-PSO* algorithm was categorized into six versions, such as $PMLG\_PSO_n$, where $n = 1, 2, 3, 4, 5, and\ 6$. In each version, various approaches are applied in terms of the number of particles or number of algorithm iterations as it

is categorized in Table 4. 2. These classes of *PMLG-PSO* are implemented for a predefined number of nodes for MPLS networks such as $P = 80, 90, 100,$ and 110 nodes. The purpose of making these variations is to inspect the success of the algorithm on different network size for generating optimal solutions. The basic structure of the algorithm (described in *Algorithm* 4. 3, *Algorithm* 4. 4, and *Algorithm* 4. 5) towards obtaining a minimization of the objective functions, is the same for all network's settings. Fig. 4. 5 shows the results of the Pareto archive that has been collected in the post-processing stage, which produced Pareto front-based solutions from $PMLG - PSO_1$ to $PMLG - PSO_6$ for a different scale of MPLS networks.

Table 4. 2: PMLG-PSO Algorithm Versions

Version	**Number of particles**	**Number of Iterations**
$PMLG - PSO_1$	20	20
$PMLG - PSO_2$	30	
$PMLG - PSO_3$	40	
$PMLG - PSO_4$	20	40
$PMLG - PSO_5$	20	60
$PMLG - PSO_6$	40	

A set of Pareto solutions from the Pareto archive, for two objective functions, are collected and presented in Fig. 4. 5 (a) − (d). As formerly explained, for each $PMLG - PSO_n$ ($PMLG - PSO_1$ to $PMLG - PSO_6$) class, dissimilar values are presented as Pareto solutions. For the ease of understanding, the obtained non-dominated solutions are presented in separate figures for each case and are linked by lines to highlight different $PMLG - PSO_n$ classes' results. Fig. 4. 5 indicate that the proposed algorithm is eligible to provide feasible Pareto based solutions with the contrast objective functions. Fig. 4. 5 also confirms that once the traffic load balancing costs increases, the resource provisioning costs decreases and vice versa.

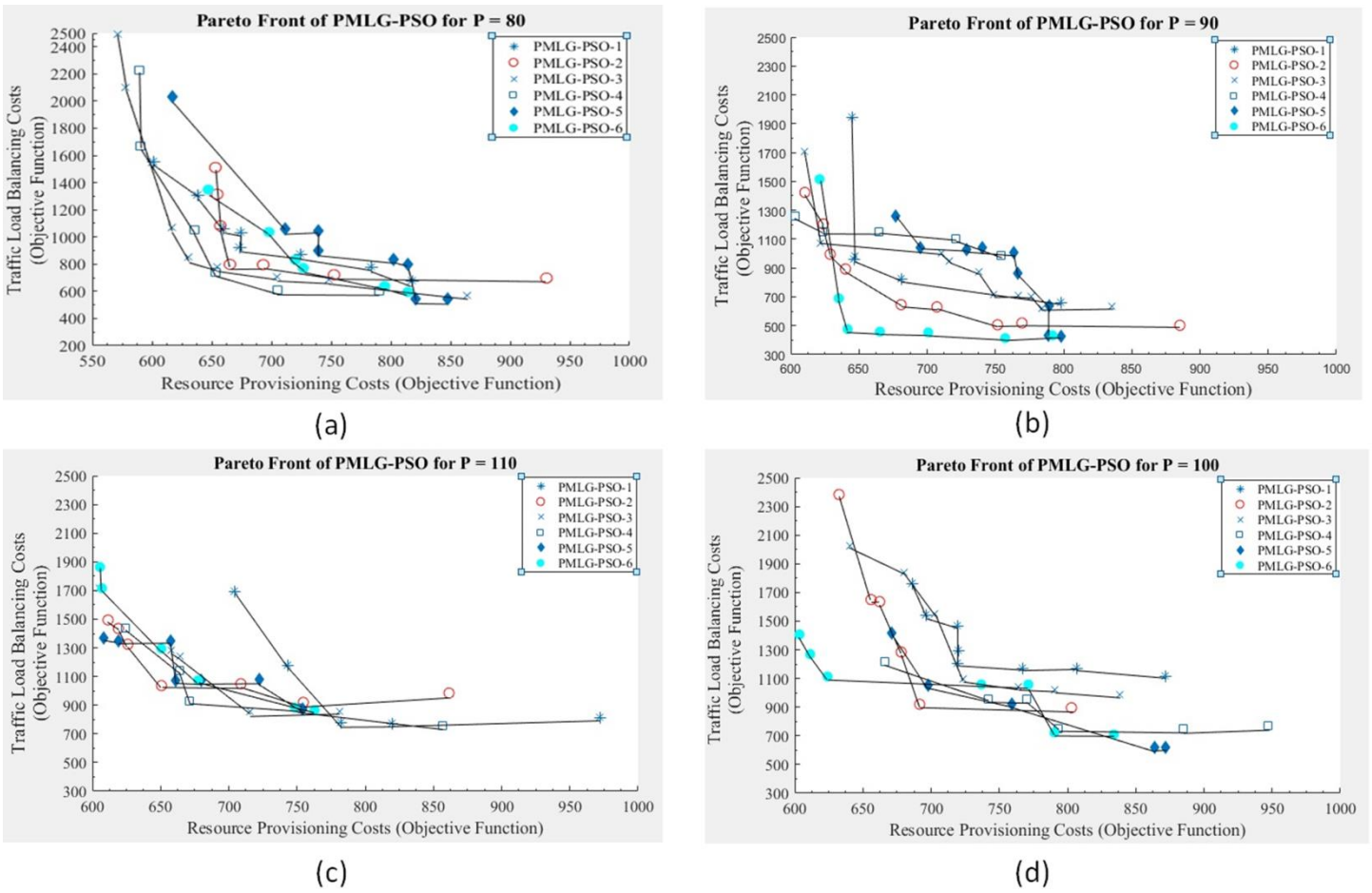


Fig. 4. 4: Pareto-Front Graphs, P = 80, 90, 100, and 110 nodes Network

4.6.2 *PMLG-PSO* Algorithm`s Convergence Investigation

To investigate the *PMLG-PSO* success ratio in terms of monitoring its convergence, the results are composed as the optimal solutions from the linear combination fitness function $init_{LC}$. To discover the algorithm convergence activity with various network scale, the algorithm has been run on 30, 60, 90 and 120 nodes network (see Fig. 4. 6 (a, b, c and d)). The purpose of this investigation is to discover the *PMLG-PSO* convergence capability while approaching towards the optimal solutions without any interruption of the exploration/ exploitation problem and creating optimal solutions during iterations. The results shown in Fig. 4. 6 (a, b, c and d) illustrates the convergence activity of *PMLG-PSO* and portray the local/ global optima problem

(if one existed) during the convergence. In the experiments, each version of $PMLG - PSO_n$ is simulated using various network sizes. From the results obtained in Fig. 4. 6, it is found that the algorithm does not experience a local/ global problem at any stage and thus, smoothly converges towards its optimal solutions. This concludes that the algorithm successfully overcomes the exploration and / or exploitation problem and produces optimal solutions with the coherent convergence. In its nature *PMLG-PSO* is a stochastic algorithm, therefore the Pareto curves change.

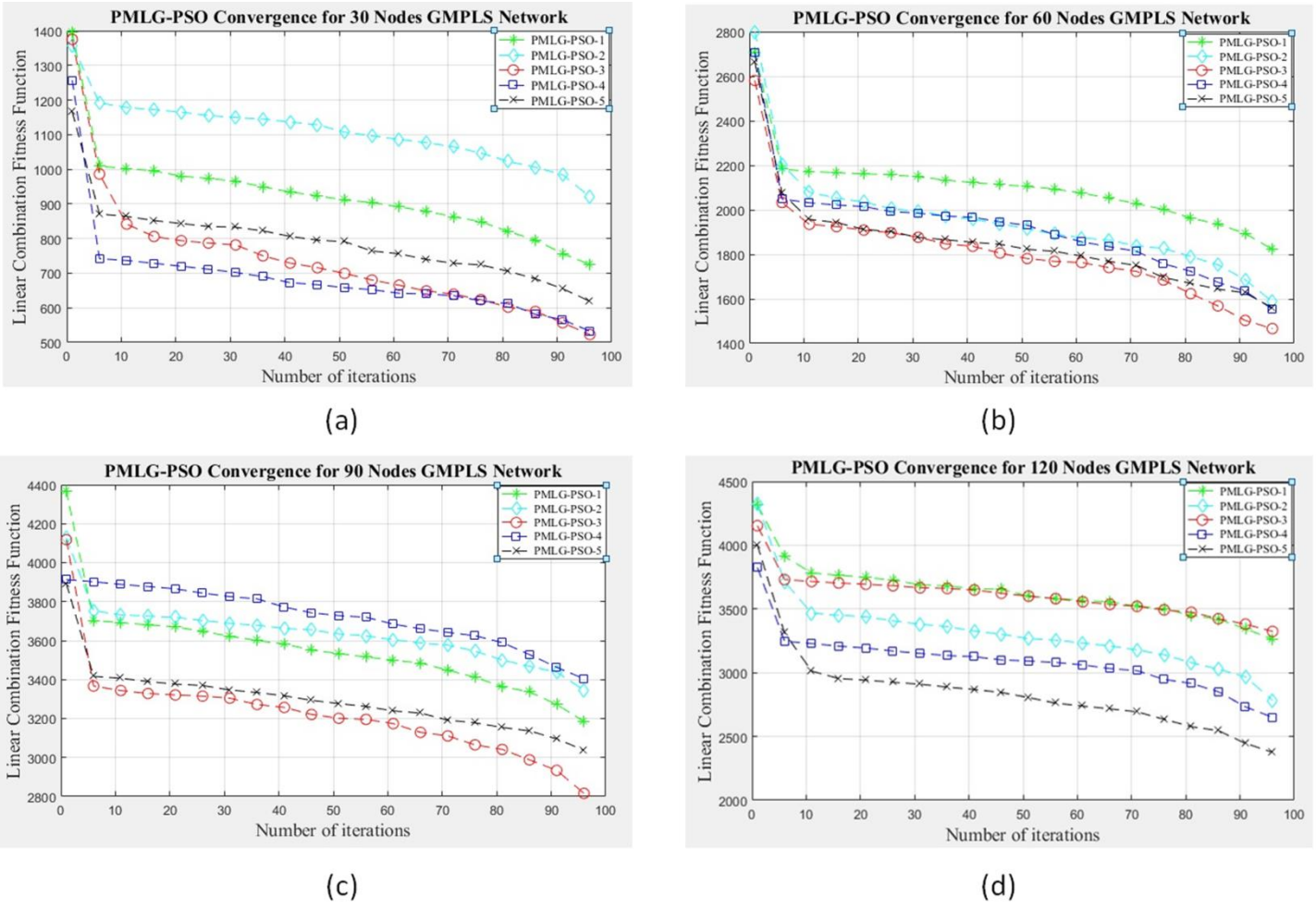


Fig. 4. 5: Analysis of Convergence of PMLG-PSO Algorithm

4.6.3 Krushkal-Wallis Test

To examine these statistical variations, the chapter uses a non-parametric test, the Kruskal-Wallis Test [108], [168]. The purpose of the Kruskal-Wallis test was to identify if at least one sample stochastically dominates over the others. Table 4. 3 show the simulation results for each version of $PMLG - PSO_n$ which were conducted for a different number of nodes; it also includes resulted H values. In the simulations, all $PMLG - PSO_n$ versions had the same number of iterations but a different number of particles. Run_n represents a number of simulations for each $PMLG - PSO_n$ algorithm version. Using the Kruskal-Wallis test, six groups of data $PMLG - PSO_n$ ($PMLG - PSO_1$ to $PMLG - PSO_6$) have tested. Based on test rules, two H values $i.e.$, H_0 and H_1 were considered. H_0 indicates that there is no significant difference between $PMLG - PSO_n$ ($PMLG - PSO_1$ to $PMLG - PSO_6$) data, while H_1 identifies a major difference. In the Kruskal-Wallis test a value of $\alpha = 0.05$ is used for the identification of H_0 or H_1. The *degree−of−freedom* df for α value is calculated where $df = k - 1$ and k represents a number of data sets [40]. In this case study ($PMLG - PSO_n$ versions), k value is 6 (as $n = 6$). For these data sets, by taking into account $\alpha = 0.05$ and $df = 5$and applying a state decision rule we get 11.0705. This means, H values less than 11.0705 satisfy H_0 hypothesis. On other hand, H values more than 11.0705 satisfy H_1 hypothesis. Having this in mind, when examining Table 4. 3, it can be noticed that for each case study shown, the computed H value is less than 11.0705. Therefore, according to Kruskal-Wallis test, there is no statistically significant difference among solutions produced by individual $PMLG - PSO_n$ versions. The experiment thus provides a proof that even a stochastic nature of the *PMLG- PSO* algorithm can produce optimal solutions that will not stochastically dominate over each other solutions. By other words, the algorithm does not produce abrupt solutions.

Table 4. 3: Kruskal-Wallis Test Results

Nodes	***Method***	$\boldsymbol{Run_1}$	$\boldsymbol{Run_2}$	$\boldsymbol{Run_3}$	$\boldsymbol{Run_4}$	$\boldsymbol{Run_5}$	$\boldsymbol{Run_6}$	***H***
80	$PMLG-PSO_1$	3.8881	5.8994	4.2613	0.0046	2.2354	0.9687	4.61
	$PMLG-PSO_2$	10.02	7.0455	2.126	3.4187	1.8147	4.1999	
	$PMLG-PSO_3$	5.9603	5.2494	6.908	1.1467	4.3732	1.7449	
	$PMLG-PSO_4$	4.0994	2.8054	0.5452	3.5105	1.0819	6.2051	
	$PMLG-PSO_5$	0.7129	0.489	0.802	3.992	7.6533	1.2524	
	$PMLG-PSO_6$	2.5965	4.064	2.8822	1.5261	2.3465	2.5549	
90	$PMLG-PSO_1$	3.2645	0.4765	4.5702	7.0599	0.0219	0.0532	2.25
	$PMLG-PSO_2$	2.0391	5.555	2.3688	1.6442	4.0108	4.56	
	$PMLG-PSO_3$	1.0528	0.7591	4.0091	2.1252	1.1673	3.1	
	$PMLG-PSO_4$	3.8526	2.8964	1.1428	4.9318	3.1934	0.6237	
	$PMLG-PSO_5$	1.1671	1.5639	3.4349	2.1304	3.4083	2.0638	
	$PMLG-PSO_6$	3.1653	3.089	6.4102	2.7346	0.7212	2.5105	
100	$PMLG-PSO_1$	0.0164	6.0832	4.7698	0.8023	5.8092	1.0933	1.75
	$PMLG-PSO_2$	2.0559	3.731	0.1134	3.7552	1.3694	7.1996	
	$PMLG-PSO_3$	0.1987	7.1935	0.6988	5.5008	4.5798	0.2009	
	$PMLG-PSO_4$	1.2743	1.8725	9.7466	2.2404	1.7511	3.5788	
	$PMLG-PSO_5$	2.6096	1.9812	2.7777	1.8384	3.5714	3.5314	
	$PMLG-PSO_6$	8.4271	4.3661	2.2835	2.6985	2.867	4.063	

110	$PMLG-PSO_1$	0.1169	5.847	0.6445	3.7258	4.5795	0.6803	4.66
	$PMLG-PSO_2$	6.1731	3.8477	0.5514	0.3387	0.4063	1.4105	
	$PMLG-PSO_3$	1.7349	5.2295	6.4817	1.3506	5.1806	2.0496	
	$PMLG-PSO_4$	2.4049	1.6217	4.8202	3.3051	6.1764	1.9375	
	$PMLG-PSO_5$	3.3695	0.7787	0.9641	2.7674	2.1304	0.886	
	$PMLG-PSO_6$	2.5208	2.0252	0.849	4.4458	3.9305	1.8917	

4.6.4 Comparative Study of *PMLG-PSO* with Metaheuristic Algorithms

The purpose of making comparison with the other popular metaheuristic approaches is to scrutinize the *PMLG-PSO* efficiency competing with other optimization techniques. For this assessment, this work uses PSO, an advance version as APSO, DA and bat algorithms. The experiments included a development of PSO, APSO, BAT and DA algorithms for the described MCOP optimization problem in MPLS networks. Each algorithm is simulated using the same *MATLAB 2016a* version, and results are collected to figure out the problems in the algorithms (if exists during the iterations). The results for both fitness functions (traffic load balancing and resource provisioning costs) were collected separately, as shown in Fig. 4. 7 and Fig. 4. 8, respectively. Fig. 4. 7 and Fig. 4. 8 show the results of the algorithms convergence, when any algorithm could not maintain balance between exploration and exploitation. In Fig. 4. 7, for all its cases (a, b, c, and d), algorithms (PSO, APSO, DA and BAT), apart from *PMLG-PSO*, had drastically failed to deal with the exploration and/ or exploitation problem and once their searching agents were trapped in their local/ global optima, the algorithms could not converge properly to find the optimal solutions for the resource provision fitness function. Consequently, these algorithms failed to find optimal solutions, and the result in this case is sub-optimal

solution. For example, in Fig. 4. 7 (d), the figure shows that PSO, APSO, DA and BAT algorithms, are poorly deteriorated to converge properly until the end of iteration. As a reference, it is appeared that PSO and APSO cannot converge after the 14^{th} iteration.

Likewise, for the traffic load balancing costs function, it can be seen in Fig. 4. 8, that PSO, APSO, DA and BAT algorithms had generated sub-optimal solutions and produced worse convergence activity with respect to the fitness function. As a case study, let us consider Fig. 4. 8 (a), where PSO algorithm cannot converge after 15^{th} iteration and repeats (produces) the same fitness function values. DA algorithm experiences the failure in various stages of convergence during the iterations, such as 25^{th} to 30^{th}, 36^{th} to 60^{th}, and 66^{th} to 78^{th} iterations. While on the other side, proposed *PMLG-PSO* algorithm has better convergence rate when compared to other algorithms (PSO, APSO, DA and BAT) in forms of generating a number of optimal solutions. It can be concluded from Fig. 4. 7 and Fig. 4. 8 that the proposed *PMLG-PSO* has not only solved local/ global optima problem but also has fastest convergence rate, for different MPLS network sizes, compare to other mentioned algorithms (PSO, APSO, BAT and DA).

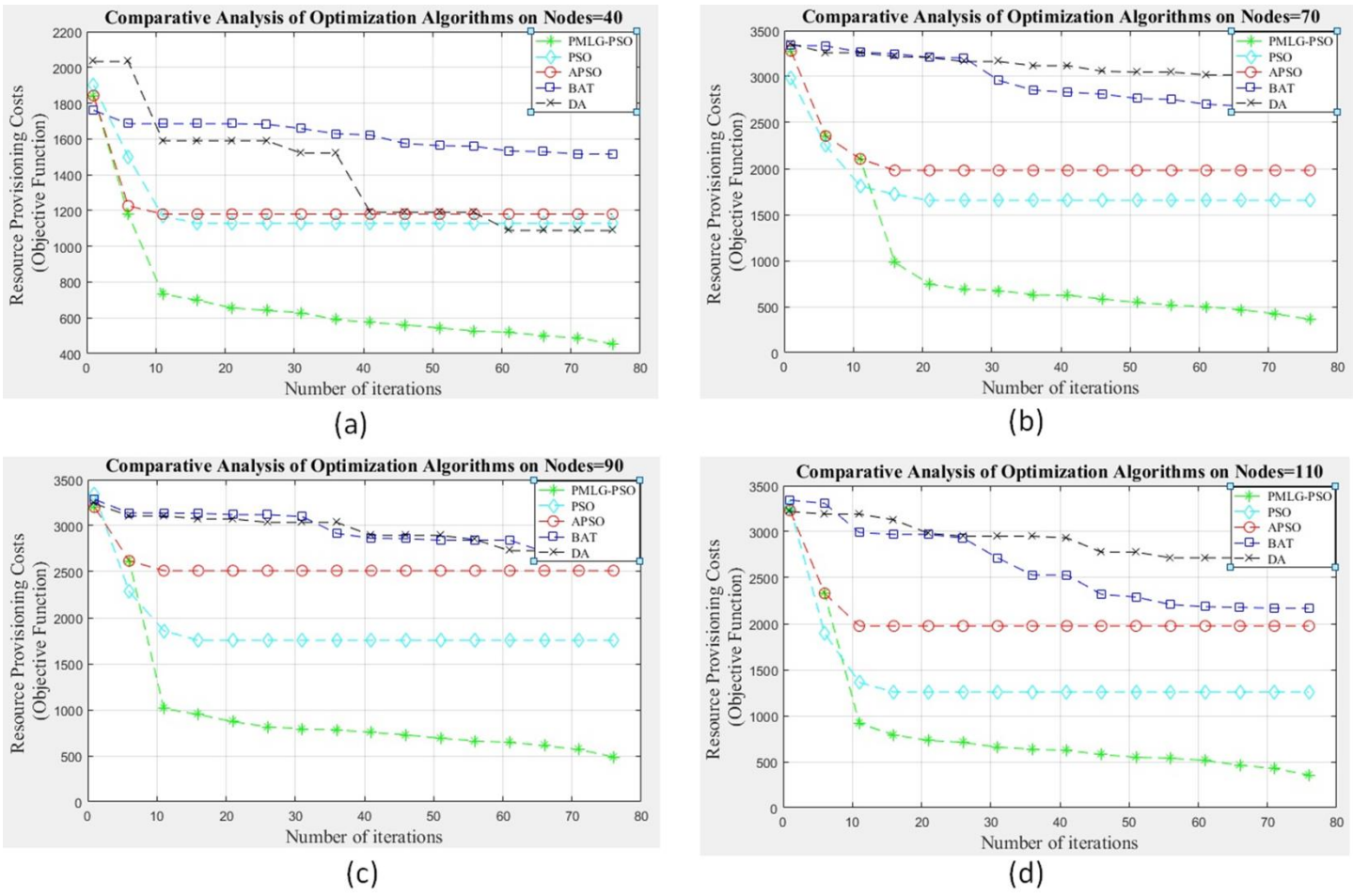


Fig. 4. 6: Comparison of Meta-heuristic Algorithms w.r.t Convergence for Resource Provisioning Function

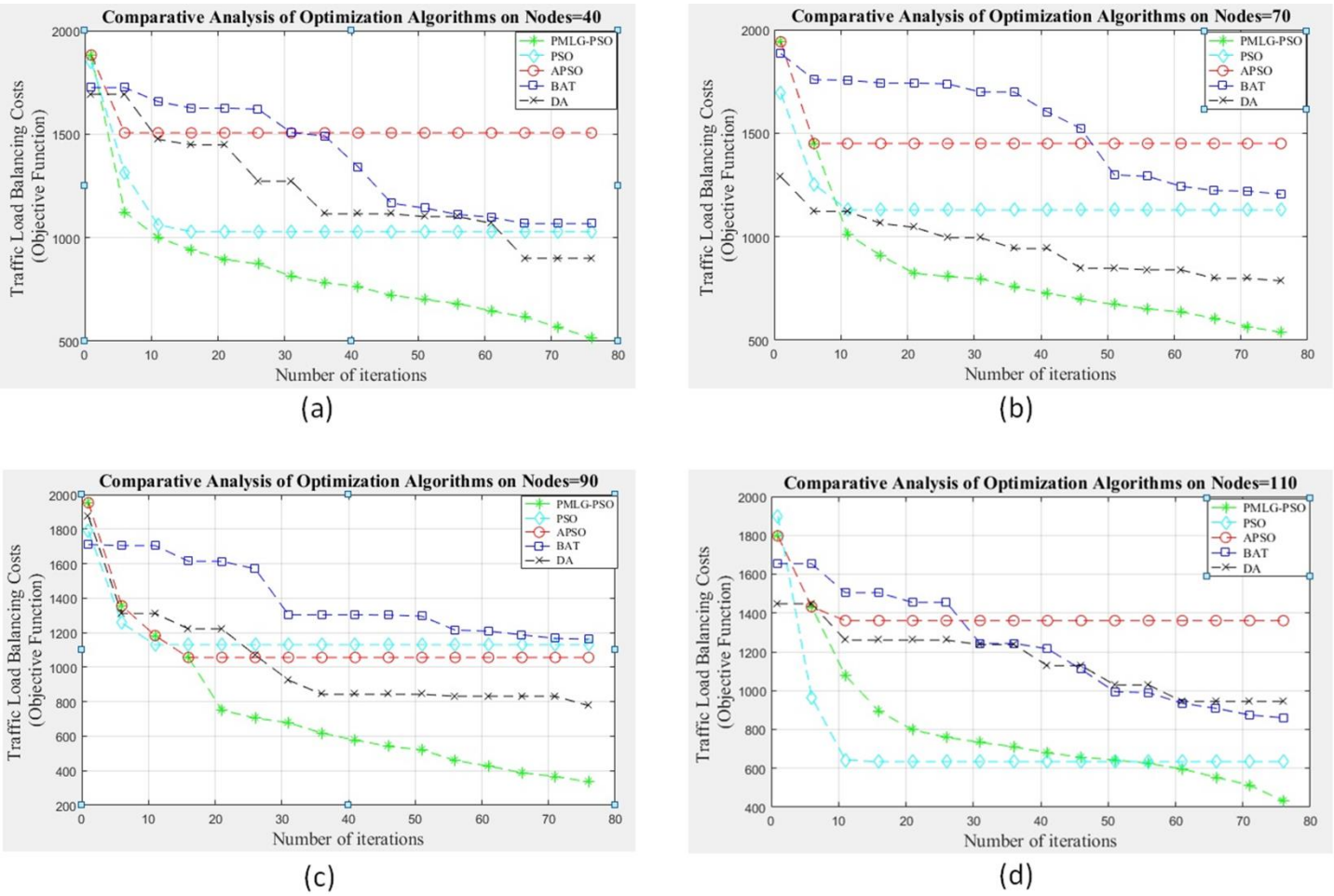


Fig. 4. 7: Comparison of Meta-heuristic Algorithms w.r.t Convergence Ratio for Load Balancing Function

4.6.5 Statistical Analysis

For further investigation, the same setup of the *PMLG-PSO* algorithm is run for 100 times to extract and then evaluate the overall performance of the algorithms using statistical parameters such as optimal fitness function costs, *MEAN* value and *Standard Deviation* values. The output results are given in the following tables. Table 4. 4 to Table 4. 7 represent the comprehensive comparative analysis of optimization algorithms over $30, 50, 80$ and 100 nodes networks. While reviewing each algorithm output in the form of an optimal solution in the given tables, it can be concluded that for each network scale, the optimum solutions are obtained from the proposed *PMLG-PSO* algorithm. As the minimum function costs are obtained from the

proposed algorithm and compare to other techniques. Similarly, for *MEAN* values, *PMLG-PSO* provides optimal *MEAN* values compare to other algorithms (PSO, APSO, DA and BAT). Here, minimum *MEAN* values are considered as the optimal values. As all mentioned algorithms have stochastic nature, therefore, the standard deviation parameters were used to monitor the deviation values of each run during all 100 iterations. For this case, DA algorithm has got the minimum standard deviation value but has sub-optimal values, whereas the other acceptable choice as for standard deviation values along with optimal solutions are from *PMLG-PSO* algorithm. The tables (Table 4. 4 to Table 4. 7) provide a conclusion that the proposed *PMLG-PSO* algorithm has maximum probability for generating optimal solutions even after implementing it into multiple numbers of simulations.

Table 4. 4: Statistical based Comparative Analysis of 30 Nodes Network

Resource Provisioning Costs Objective Function					
	PSO	**APSO**	**BAT**	**DA**	***PMLG-PSO***
Optimal Sol.	228	168.3926	475.9140	537.79	66.3332
Means	585.3900	758.9721	902.6723	615.2437	400.9275
Std Dev.	162.2130	318.7124	156.0202	49.6646	210.9559
Traffic Load Balancing Costs Objective function					
	PSO	**APSO**	**BAT**	**DA**	***PMLG-PSO***
Optimal Sol.	50.2371	284.1536	65.8446	141.07	43.0134
Means	771.5276	1168	1056.4	474.1919	467.4116
Std Dev.	233.5189	259.6976	348.5924	160.7355	224.6925

Table 4. 5: Statistical based Comparative Analysis of 50 Nodes Network

Resource Provisioning Costs Objective Function					
	PSO	**APSO**	**BAT**	**DA**	***PMLG-PSO***
Optimal Sol.	644	448.7235	877.32	774.38	80.8595
Means	1016.2	1381.8	1508.6	908.4148	465.0897
Std Dev.	221.678	507.7226	268.8695	68.1039	275.8044
Traffic Load Balancing Costs Objective function					
	PSO	**APSO**	**BAT**	**DA**	***PMLG-PSO***
Optimal Sol.	295.02	880.3437	354.54	505.34	103.0016
Means	1230.8	1968.1	1950.1	836.5914	490.1594
Std Dev.	423.7258	378.0275	511.4425	181.6391	254.2641

Table 4. 6: Statistical based Comparative Analysis of 80 Nodes Network

Resource Provisioning Costs Objective Function					
	PSO	**APSO**	**BAT**	**DA**	***PMLG-PSO***
Optimal Sol.	1070	844.74	1570.2	1035.8	89.738
Means	1761	2179.8	2555.7	1213.1	685.7291
Std Dev.	304.7261	827.6866	383.1460	97.4751	532.9176
Traffic Load Balancing Costs Objective function					
	PSO	**APSO**	**BAT**	**DA**	***PMLG-PSO***
Optimal Sol.	430.44	1428.7	735.68	585.2	121.78
Means	1890.8	3326.4	2679.2	907.0565	687.1046
Std Dev.	757.4162	627.0519	647.9020	184.7952	608.8075

Table 4. 7: Statistical based Comparative Analysis of 100 Nodes Network

Resource Provisioning Costs Objective Function					
	PSO	**APSO**	**BAT**	**DA**	***PMLG-PSO***
Optimal Sol.	1585	1315.2	1786.6	1333.8	80.878
Means	2230.5	2740.2	3066.1	1512.5	765.0592
Std Dev.	279.4764	1063.5	504.7764	108.3417	643.6658
Traffic Load Balancing Costs Objective function					
	PSO	**APSO**	**BAT**	**DA**	***PMLG-PSO***
Optimal Sol.	634.44	2227	983.44	736.77	140.03
Means	2226.9	2227	3634.7	1013.1	722.7989
Std Dev.	886.1651	641.8836	881.7565	192.8892	776.1407

4.7 Summary

The exponential growth in the data traffic across the network require significant throughput enhancement to prevent network congestion. To address this issue, various approaches in the form of algorithms were introduced. The objective of these algorithms is to spread the network traffic that guarantees network reliability and efficiency. PSO algorithm is famous for solving such type of complex optimization problems in different fields. However, PSO itself has an intense problem of balancing exploration and exploitation searching that undergoes its capability of generating optimal solutions in various conditions. Therefore, there was a need to offer such version of PSO that not only enhances its searching abilities but also helps to provide optimal solutions for complex optimization problems of networking. In this chapter, an optimization model of MPLS network has been presented in the form of two contrast objective functions loaded with multiple networking constraints, which is considered as MCOP problem that belongs to the class of NP-hard optimization problem. The two objective function

that are considered for MPLS optimization model were (i) resource provisioning cost, and (ii) traffic load balancing cost objective functions. For the given complex MPLS optimization model a novel *"Pareto based Modified Local Global Particle Swarm Optimization (PMLG-PSO)"* algorithm is offered as an approach for solving optimization problem. In additional to this, a Pareto approach is also adapted for contrasted objective functions in order to create non-dominant solutions and then plot then as Pareto front. In the chapter, number of experiments was conducted for solving MPLS network optimization problem by proposing optimal solutions and then also makes comparative analysis with other algorithms. In numbers, five experiments were conducted such as (i) producing Pareto front as non-dominant solutions, (ii) convergence analysis of proposed *PMLG-PSO* with various network scales, (*iii*) comparison of *PMLG-PSO* with PSO, APSO, BAT and DA in terms of convergence activity, (i*v*) Statistical parameters test using *MEAN*, standard deviation and optimal values in competence to other aforementioned algorithms, and (*v*) Khushkal-Wallis test for testing stochastic nature of *PMLG-PSO*.

While exploring the results obtained from the above mentioned experiments, it is shown that proposed *PMLG-PSO* outclass its competitive algorithms (PSO, APSO, BAT, and DA) in terms of convergence and also with statistical data obtained. *PMLG-PSO* has successfully generated Pareto front for objective functions that is shown in Fig. 4. 6 and also produces optimal solutions for various considered versions of the algorithm ($PMLG - PSO_n$), as shown in Fig. 4. 7. While comparing the *PMLG-PSO* with its rival algorithms (PSO, APSO, BAT, and DA), the proposed algorithm smartly handle local/ global optima problem and converge effectively towards optimal solutions compared to other algorithms (see Fig. 4. 8 and Fig. 4. 9). Moreover, for statistical results obtained in the form of data also showed the superiority of *PMLG-PSO* compared to its competing algorithms, as shown in the form of Table 4. 4 to Table 4. 7. As *PMLG-PSO* is stochastic nature algorithm, therefore, the Khushkal-Wallis test was conducted and the results were given in Table 4. 3. From the test, it is settled that the *PMLG-PSO* does not suffer with producing abrupt solutions. From all the experiments conducted and the results obtained in different form conclude that the proposed *PMLG-PSO* can smartly handle the local/ global optima problem which exists in *standard* PSO algorithm and it produce optimal solutions for the complex optimization problem in MPLS network having multiple objective functions. *PMLG-PSO* also shows its dominance by providing optimal solutions in various experiments with its competing algorithms (PSO, APSO, BAT, and DA).

Chapter 5

Adjustable Bat Algorithm for Multi-Objective Multiple Constrained Based Optimization Network Problem

Bat algorithm has got momentous popularity in recent years for its success of solving various optimization problems, specifically for multi-objective dependent problems. Nevertheless, the algorithm has various parameters that have not been explored before. This chapter will first describe the bat algorithm and its parameters that are used for optimal solutions searching in problem domain. The work in this chapter will highlight the significance of bat algorithm parameters and discuss them with experimental results. The limitations to the bat algorithm are also discussed. The literature review related to bat algorithm for network optimization is briefly discussed in the chapter. The multi-objectives based optimization model of MPLS network is presented and mathematically formulated. For the given optimization model, the novel version of bat algorithm is proposed in the later sections. The task of the proposed technique is to generate the optimal solutions for the MPLS optimization model. A comprehensive comparison study is included in the last section of the chapter, in which the proposed technique is compared with other metaheuristics in order to validate its performance.

5.1 Introduction to the Bat Algorithm

In 2010, X. Yang [81] contributed a novel bat algorithm in the family of swarm metaheuristics that collected remarkable recognition because of to its efficient convergence during its initial stage when fixing different optimization problems. Bat algorithm imitates the echolocation strategy of a bat searching for its prey. Bats emits calling signals inside a searched space as well as examine the collected echoes of these calls arrived from a prey. This plan of action assists the bat to reach in the direction of its prey. The bat algorithm is put into practice by

applying subsequent instructions that are given in the form of mathematical expressions. During the course of the initialization phase, the algorithm determines the initial frequency of the bat signals, given as follows [81], [169] – [170];

$$f_i^{t+1} = f_{max} + (f_{max} - f_{min})\,\beta \tag{5.1}$$

where β represents the random number in the range of [0 , 1] and f_{min}, f_{max} are used for minimum and maximum frequencies, respectively. At the processing stage, the β value changes with the random values defined within the range. In addition to this, the velocities v_i^{t+1} and positions x_i^{t+1} in each iteration update by the following equations [81], [169] – [170];

$$v_i^{t+1} = v_i^t + (x_i^t - x_{Global-best})\, f_i^{t+1} \tag{5.2}$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \tag{5.3}$$

In the Eq. (5.2), $x_{Global-best}$ is referred to the global best position. In the next stage of the algorithm, each i^{th} bat takes a random walk step to produce local best position, denoted as $x_{i,Local-best}^t$, by fulfilling the condition of $rand > P_r^t$ [81], [169] – [170];

$$x_{i,Local-best}^t = x_i^{t+1} + \varepsilon\, L_d \tag{5.4}$$

The process of echolocation in bat algorithm is highly reliant on its pulse rate (P_r^t) and loudness parameter (L_d^t). In cases where the bat gets near in the direction of its prey (towards optimal solution), the pulse rate (P_r^t) grows and loudness (L_d^t) reduces, by the following expression [81], [169] – [170];

$$L_d^{t+1} = \alpha\, L_d^t \tag{5.5}$$

$$P_r^{t+1} = P_r^t \left[1 - e^{-\gamma(t+1)}\right] \tag{5.6}$$

where α,γ and ε are the constants, which ranges between [0 , 1], whereas L_d represents the *Average Loudness*.

5.2 Literature Review

In recent years, researchers are intensively searching for metaheuristic optimization techniques that can offer fast convergence rate and generates optimal solutions for various complex optimization problems. It has also been discussed in the previous chapters that metaheuristic algorithms usually fails to maintain tradeoff balance between exploration and exploitation process and thus, come up with poor convergence rate. Even the popular metaheuristics suffer with this convergence problem during different iterations of the algorithm [171], [172].Despite the fact that the bat algorithm is well-liked due to its fast search approach to with optimal results in the beginning, there is always a space for improvements. Supported by the research carried out in [169], [173] - [174], it was discovered that the bat algorithm has a slower convergence rate during the later stages of the algorithm. It was additionally observed that shortly after the first couple of initial iterations, the algorithm can tends towards exploitation more than exploration in the searching space (unbalance exploration and exploitation). Bat algorithm offers better convergence in its local exploitation process by using its loudness and pulse rate parameters, while the exploration process is dependent on the random searching of bat population [175] - [176]. Although, Bat algorithm offers impressive convergence compared to other metaheuristics, but it contains the high risk of local optima problem after few iteration and consequently, the algorithm convergence slow down and reduce accuracy [177] - [179].

Since the optimization problems continued to become more complex, the researchers have come up with various versions of Bat algorithm. These attempts of improvements were done to enhance algorithm`s performance for solving various complex optimization problems [180]. At the early stages of improvements of Bat algorithm, Yang [181] (the founder of Bat algorithm), offered the improved version of bat for multi-objective optimization problems. Komarasamy and Wahi [182] offered a clustering based bat algorithms, called as KMBA (K-Means Bat Algorithm), A fuzzy logic based bat algorithm is presented by Khan *et al*. [183], named as FLBA (Fuzzy Logic Bat Algorithm), in which the fuzzy logic techniques were incorporated with bat algorithm`s parameters. To solve feature selection and classification problems, a BBA (Binary Bat Algorithm) was proposed by Nakamura *et al*. [184]. Fister *et al*. [185] presented QBA (Quaternion Bat Algorithm) for solving large-scale optimization problems

and computational geometry. IBA (Improved Bat Algorithm) is presented by Yilmaz *et al*. [186] for solving continues constrained and unconstrained optimization problems.

Some researchers adopted levy flight technique for improving bat algorithm. Using levy flights and chaotic maps inside the Bat algorithm, the CBA (Chaotic Bat Algorithm) is offered by Lin *et al*. [187]. On the other hand, with the combination of levy flights and differential operations into the bat algorithm, Xie *et al*. [188] proposed DLBA (Differential Operator and Levy Flight Bat Algorithm).Another improved version of bat algorithm which was implanted with levy flights and subtle loudness level variations and pulse rates were presented by Jamil *et al*. [189] and named as IBA (Improved Bat Algorithm). Zhou *et al*. [190] improved the exploration process of bat algorithm. For optimum design of a power system stabilizer, Ali *et al*. [191] presented a new version of bat algorithm.

In addition to this, some researchers offered the hybrid versions of bat algorithm with other metaheuristics for solving various complex optimization problems [192] - [196]. To improve the global search, a chaos mechanism based bat algorithm was presented by Gandomi and Yang [197]. For continuous and combinatorial optimization problems, Iztok Fister Ji. *et al*. [198] developed the self-adaptive bat algorithm. Fister et al. [199] also presented hybridized model of bat and DE strategies. A hybrid algorithm of bat and HS (Harmony Search) was presented by Gaige Wang and Lihong Guo [200] in order to solve the global numerical optimization problems. While some authors make a comprehensive analysis of bat algorithm with other metaheuristics to understand the behavior of the algorithm while applying to various benchmark functions and optimization problems of different applications [201] - [202].

This aim of this chapter is to investigate various parameters of bat algorithm and improved its convergence by proposing a novel version and applies for MPLS optimization. The matter of effective convergence in the bat algorithm in relation to the MCOP optimization problem is still an open and unsettled research problem. The significance of the bat algorithm's loudness parameter has not been thoroughly studied yet.

5.3 MPLS Optimization Model

This chapter highlights the significance of loudness parameter with the support of experimental results and will demonstrate the usefulness of the loudness parameter to enhance the bat algorithm functionality regarding improved solutions for the MPLS network optimization problem. The chapter is going to propose a modified model of the bat algorithm which is integrated with a loudness optimizer and named as the *Adjustable Bat (ABAT)* algorithm. The *ABAT* will be applied for producing both, optimal and non-dominated, solutions in the form of the Pareto front by taking into consideration of two objective functions: The Load Balancing Costs and Routing Costs. In the case of applying the *ABAT* algorithm, various loudness levels will be taken into consideration and then apply them to a various number of nodes in the MPLS network to examine the algorithm overall performance. In verification experiments that will be discussed in the future, a variety of traffic flows is supposed to be directed towards ingress LSRs in MPLS domain. The goal of ingress LSRs is going to determine multiple paths dependent on the quantity of traffics, which will be accomplished through the recommended *ABAT* algorithm by ingress LSRs.

The simulation of MPLS networks is conducted by employing graph theory notations; $G = (R, E)$. The network structure was symbolized as a graph G in which its vertices R signify routers, together with its edges E is indicated as links in between vertices. The total number of r_i^{th} routers used to be assembled as R_{set} or $R = \{r_i, i = 1,2,3, \dots, n\}$, and the number of links e_i collection can be signified as E_{set} or $E = \{e_{1_{(i,j)}},\ e_{2_{(i,j)}}, e_{3_{(i,j)}}, \dots, e_{n_{(i,j)}},\}$. Each individual $e_{i_{(i,j)}}$ symbolizes a link between a source node i to an adjacent or interconnected node j. The offered bandwidth is going to be signified by $B_{cost_{(i,j)}} = \{b_{icost_{(i,j)}}, \dots b_{ncost_{(i,j)}}, i = 1,2,3, \dots, n\}$. As soon as the traffic demand $i_{traffic}^{th}$ is collected at ingress LSR, a quadruple approach $(t_{ID};\ s_{ID};\ d_{ID};\ b_{ID})$ is taken into account, in which t_{ID} is taken as an identification (ID) towards a particular traffic request, s_{ID} is the ID for the source (ingress LSR) router, d_{ID} is the destination (egress LSR) router ID and b_{ID} is a bandwidth ID for a specified LSP. Here it is assumed that all the QoS specifications are integrated in to available bandwidth ranges. The experiments made use of two objectives functions, and are described in the next sections.

5.3.1 Computing Routing Delay Cost Objective Function

For MPLS networks, the network vendors and service suppliers receive the cost per packet flow through a link and described as the routing cost. The delay of traffic movement within network is often experienced simply because of the excessive traffic chunks, which is acknowledged as the routing delay cost. Due to this fact, the objective function's task is to minimize this routing delay cost to optimize network traffic flow. It is believed that an amount of traffic demands is delivered at the ingress LSR, in which every traffic $i^{th}_{traffic}$ is associated to the total traffic set $Traffic_{set}$; $.e\ \ i^{th}_{traffic} \in Traffic_{set}$. The task for the proposed algorithm is to do computations for a number of optimal paths for all traffic requests $i^{th}_{traffic}$ by means of minimizing the described routing cost objective function. The optimal paths is supposed to be computed for the traffic demands obtained and link costs that are correlated together with some other constraints. This approach can be expressed mathematically as follow;

$$mini\{Routing_{delay\ cost}\} = \sum_{i^{th}_{traffic} \in Traffic_{set}} R \sum_{e_{i_{(i,j)}} \in E} I_{delay} \quad \forall \quad p^{i^{th}}_{traffic} \in P_{set} \tag{5.7}$$

Subject to constraints;

$$p^{i^{th}}_{traffic} = \sum_{i^{th}_{traffic} \in X^{e}_{traffic}} x_{I_{delay}} \begin{cases} x_{I_{delay}} = 1, & p^{i^{th}}_{traffic} \in P_{set} \\ x_{I_{delay}} = 0, & otherwise \end{cases} \tag{5.7 a}$$

$$X^{e}_{traffic} > 0 \quad \forall \ e_{i_{(i,j)}} \in E \tag{5.7 b}$$

$$x_{I_{delay}} > 0 \quad \forall \ e_{i_{(i,j)}} \in E \tag{5.7 c}$$

$$\sum_{b_{i_{cost}} \in B_{cost(i,j)}} \forall \ i^{th}_{traffic} \in Traffic_{set} \tag{5.7 d}$$

In the above equations, P_{set} signifies a set of paths and this set contain the number of computed paths $p_{traffic}^{i^{th}}$. The constraint associated with these paths; $i.e$ there must be a routing/ network delay I_{delay} with each computed path $p_{traffic}^{i^{th}}$ which has traffic $x_{traffic}^{i^{th}}$. Eq. (5.7) mathematically describes the routing cost objective function, which is to minimize the routing delay cost in order to compute the optimal paths along with abovementioned constraints. According to constraint illustrated in Eq. (5.7 a), every link $e_{i_{(i,j)}}$ that has selected for path computation must have a traffic flow $x_{traffic}^{i^{th}}$ over it and also contains network/ link delay (denoted with link delay variable $x_{I_{delay}}$).When the delay is added in the path computation that is, the delay variable $x_{I_{delay}}$ value is 1, in that case the path $p_{traffic}^{i^{th}}$ is added in the optimal paths set P_{set} alternatively; $p_{traffic}^{i^{th}}$ will not be regarded as the optimal path if $x_{I_{delay}} = 0$. Based upon constraint Eq. (5.7 b), whenever $X_{traffic}^{e} > 0$, it means that there will be traffic flow over nominated links $e_{i_{(i,j)}}$ for path computation. In plain terms, this constraint guarantees the existence of traffic flow over the particular links chosen for the path calculation. According to constraint Eq. (5.7 c), there has be a routing delay (symbolized by $x_{I_{delay}}$ variable) across the links; for this reason, the condition of $x_{I_{delay}} > 0$ must be satisfied for each link $e_{i_{(i,j)}}$. In constraint Eq. (5.7 d), the link $e_{i_{(i,j)}}$ will only be chosen for LSP computation that has a traffic flow, if the link $e_{i_{(i,j)}}$ comprises the bandwidth which should be greater than 0 and is consequently qualified for path computation having minimum routing cost function.

5.3.2 Finding Load Balancing Routing Cost Objective Function

Regarding MPLS dependent networks, the traffic-engineering (TE) assists as the procedural framework which governs the load balancing strategy. Broadly speaking, the source routing is commonly executed in MPLS networks, in which only edge LSRs are put into use for the path computation. This type of strategy of path computation can easily have immense damaging impacts on the system of accomplishing the optimal load balancing as well as lead to network overload. In the chapter, the hierarchical methodology of the load balancing routing in

the MPLS network is recommended. Let us simply take the subset of border routers as R^*_{set} or $R^* = \{r_1^*, r_2^*, r_3^*, \ldots r_n^*\}$, in which each r_i^* acts as LER and $R^* \in R$. The traffic flow $i^{th}_{traffic}$ routed by R^* is written as t_r. The routed traffic flow t_r coming from the source LER towards the destination router is s^{t_r} and d^{t_r}, correspondingly. When it comes to the operation of the source routing for each LER r_i^*, let us allocate a routing variable as $x^{t_r}_{(i,j)}$. Each of these $x^{t_r}_{(i,j)}$ variables recognizes the chunk of the traffic movement rate coming from a r_i^* router which is transmitted through the link $e_{i_{(i,j)}}$. With regard of accomplishing a multipath routing within MPLS network; it is necessary to apply the appropriate following constraints;

$$0 \leq x^{t_r}_{(i,j)} \leq 1 \tag{5.8}$$

For avoiding the packet loss across the network, the following constraints are implemented as;

$$\sum_{e_{(i,j)} \in E} x^{t_r}_{(i,j)} - \sum_{e_{(i,j)} \in E} x^{t_r}_{(i,j)} = 1, \qquad r_i^* = s^{t_r} \tag{5.9}$$

$$\sum_{e_{(i,j)} \in E} x^{t_r}_{(i,j)} - \sum_{e_{(i,j)} \in E} x^{t_r}_{(i,j)} = 0, \qquad r_i^* \neq s^{t_r}, d^{t_r} \tag{5.10}$$

$$\sum_{e_{(i,j)} \in E} x^{t_r}_{(i,j)} - \sum_{e_{(i,j)} \in E} x^{t_r}_{(i,j)} = -1, \qquad r_i^* = d^{t_r} \tag{5.11}$$

The constraint defined in Eq. (5.9) gives the demonstration of the traffic movement s^{t_r} coming from the source router LER r_i^* which must have the value of 1. In addition to this, the constraint of Eq. (5.10) verifies the data flow direction between the LSRs as $r_i^* \neq s^{t_r}, d^{t_r}$, and the opposite data flow $i.e$ from d^{t_r} while having value of -1, as given in Eq. (5.11). The most critical stage in the hierarchical routing is considered as the avoidance of a link overload by bringing in the concept load balancing using Eq. (5.12);

$$\sum_{R^* \in R} \sum_{i^{th}_{traffic} \in Traffic_{set}} x^{t_r}_{(i,j)} \leq b_{i_{cost}} \alpha \tag{5.12}$$

wherein α can be described as control variable for balancing, and it decides the upper border threshold degree of the link utilization regarding the link capacity, given that $\alpha = \frac{link\ utilization}{link\ capacity}$. In keeping with the specifications for traffic engineering, the objective of load balancing is actually to minimize the described threshold value of α at each r_i^* router. In this case, the load balancing function which is formulated on the hierarchical routing model, having $r_i^* \in R_{set}^*$ will be reported as;

$$mini\{Load_{balancing\ routing}\} = \sum_{r_i^* \in R_{set}^*} \alpha \tag{5.13}$$

In the subsequent sections, the chapter offers a novel ABAT algorithm for determining both the optimal (minimum) routing costs and the load balancing costs as objective functions for MPLS networks.

5.4 Proposed *Adjustable Bat (ABAT)* for the Optimization Problem

This particular section will propose *ABAT* algorithm along with Pareto plan. The mission is to improve the searching functionality $i.e.$, impressively enhance the convergence rate of the bat algorithm by integrating with some new parameters. In the standard bat algorithm, the average Loudness L_d factor is determined on the computation of average loudness levels of bats having difference of the maximum loudness $L_{d_{max}}$ and minimum loudness $L_{d_{min}}$, and in which the L_d value fluctuates throughout the iterations. In simple terms L_d is calculated to be the average between $L_{d_{max}}$ and $L_{d_{min}}$ during algorithm's iterations. As mentioned in to [169], whenever the difference between $L_{d_{max}}$ and $L_{d_{min}}$ appears to be high, the convergence factor of the algorithm is going to be improved in relation to *"offer"* the optimal solutions. This implies that the significance of the average Loudness L_d parameter has been *"ignored"* so far. In other words, the standard bat algorithm could not take advantage of this important and vital parameter for improving its convergence rate. Therefore, a "*LO* (*Loudness Optimizer*)*"* is introduced into our proposed *ABAT* algorithm for the computation of the optimal average loudness L_d value. This helps the algorithm performance in terms of improved convergence rate. In the suggested

approach of *ABAT*, when the algorithm trapped at its local optima and as a result does not search the global searching space, then a new parameter *LO* is launched. When it comes to the computation of *LO*, a random average value of Loudness L_d will be chosen from the historical average loudness L_d values mentioned as l_d. l_d is then integrated in the mathematical statement as follows;

$$LO = l_d + \left(L_{d_{max}} - L_{d_{min}}\right) \times Gaussian\,(\mu\,, \sigma^2) \tag{5.14}$$

In Eq. (5.14), $Gaussian(\mu, \sigma^2)$ indicates a Gaussian distribution involving the mean of $\mu = 0$ and time-varying standard deviation. This equation supports the escape of searching agent from local optima using *LO* during number of iterations.

$$\sigma = \sigma_{max} - (\sigma_{max} - \sigma_{min}) \tag{5.15}$$

While $\sigma_{max} = 1$ and $\sigma_{min} = 0.1$, as as revealed based on the scientific study made in [108]. The predicted *LO* value is then utilized as an average Loudness L_d, and will be employed for the calculation of $x^t_{i,Local-best}$, as displayed in Eq. (5.4). The pseudo-code of the offered *ABAT* is provided in the format of *Algorithm* 5. 1 along with the method of *LO* computation which is being discussed by the following steps;

1. According to *Algorithm* 5. 1, when $x^{t+1}_{Glbal-best}$ is achieved with the support of objective function f_1 and f_2 values, at this stage the $if\ condition$ will be applied.

2. if the fitness value of $x^{t+1}_{Glbal-best}$ is found to be an optimal solution, compare to its previous $x^t_{Glbal-best}$, then the algorithm will store its optimal solutions.

3. On the other hand, if $x^{t+1}_{Glbal-best}$ is not optimal, then the algorithm will jump towards *LO* process.

4. The *LO* will be computed based on the given Eq. (5.14) and then will be considered as L_d.

5. The L_d value will be used to compute $x^{t+1}_{Glbal-best}$ by using Eq. (5.4)

6. Then again comparison between fitness values of $x^{t+1}_{Glbal-best}$ and $x^t_{Glbal-best}$ will be taken and this if continues until the condition met

Algorithm 5. 1: Pareto based *Adjustable Bat (ABAT) Algorithm* for MPLS Network Optimization

INPUT: fitness functions and searching agents for ABAT algorithm

Result: Optimal Paths for MPLS networks for $Routing_{delay\ cost}\ (f_1)$ and $Load_{balancing\ cost}(f_2)$

for f_1 and f_2 ***do***

  for bats ***B*** in each dimension d ***do***

    initialize random positions $x_i = x_1, x_2, x_3, \ldots x_n$

    initialize random velocities $v_i = v_1, v_2, v_3, \ldots v_n$

    initialize frequencies (maximum and minimum frequency with ranges) $f_i = [f_{max}, f_{min}]$

    initialize loudness levels as (maximum and minimum loudness with ranges) $L_d = [L_{d_{max}}, L_{d_{min}}]$

    compute initial frequency by using Eq.(5.1)

    compute fitness function solutions for f_1 and f_2

    compute $x^t_{Global-best}$

  end

  while number of iterations < maximum number of iterations ***do***

    for f_1 and f_2, $i = 1{:}N$ in each d ***do***

      update frequency f_{req} using Eq. (5.1) with random β value

      update velocities v_i using Eq. (5.2)

      update positions x_i using Eq. (5.3)

      compute f_1 and f_2 based on updated x_i

      store non-dominant solutions as Pareto solutions

      if $rand > P^t_r$ ***do***

        compute $x^t_{i,Local-best}$ based on Eq. (5.4)

      end

      compute $x^{t+1}_{Global-best}$

      if updated $x^{t+1}_{Global-best} < previous\ x^t_{Global-best}$ ***do***

        update f^t_1 with f^{t+1}_1 and f^t_2 with f^{t+1}_2

        update $x^t_{Global-best}$ with $x^{t+1}_{Global-best}$

minimize loudness L_d^{t+1} using Eq.(5.5)

increase pulse rate P_r^{t+1} using Eq.(5.6)

compute average loudness L_d^t

else

take random value from previous average loudness values l_d

compute loudness optimizer LO using Eq. (5.14) and Eq. (5.15)

take LO as average loudness value L_d

compute $x_{Global-best}^{t+1}$

update f_1^t with f_1^{t+1} and f_2^t with f_2^{t+1}

update $x_{Global-best}^t$ with $x_{Global-best}^{t+1}$

minimize loudness L_d^{t+1} using Eq.(5.5)

increase pulse rate P_r^{t+1} using Eq.(5.6)

end

end

end

end

5.5 Experimental Arrangement and Analysis of Obtained Results

With the implementation the suggested ABAT algorithm, two objectives functions (Routing Delay Cost and Load Balance Routing Cost) happen to be examined in order to determine the optimum together with Pareto front solutions as for MPLS networks optimization problem. In the later sections, the short notation for the routing delay costs will be routing costs, and load balancing routing costs will likely to be load balancing costs. The application of the *ABAT* algorithm is considered to measure its effectiveness of finding optimal and non-dominated solutions for MPLS networks in opposition to Yang's bat algorithm notated as PSO, *standard* bat, DEA, and CBA [79], [81], [83], [203], in order to test its convergence rate for a variety of

network scales. Variety of experiments has been carried out on different experimental setups. Such setups differ with regards to network conditions and sizes, testing parameters, proposed ABAT versions along with loudness configurations and so on, for delivering the results in the form of graphic representation together with tables (statistical evaluation). Five well-adjusted versions of the *ABAT* algorithm had been examined. For each single version, the variants of an established loudness level are applied, minimum $L_{d_{min}}$ and maximum loudness level $L_{d_{max}}$. Table 5. 1 displays the different experimental setups whilst Table 5. 2 provide the parameter controls of algorithms utilized in experimental research work.

Table 5. 1: Experimental Setups

5.5.1 Pareto Front Graph

5.5.2 Investigation of Various Versions of ABAT Algorithm

***ABAT* Versions**	**$L_{d_{max}}$, $L_{d_{min}}$ Range**	**No. of Nodes in MPLS Network**	**No. of Iterations**
ABAT−1	$L_{d_{max}} = 3$, $L_{d_{min}} = 0$		
ABAT−2	$L_{d_{max}} = 10$, $L_{d_{min}} = 0$		
ABAT−3	$L_{d_{max}} = 20$, $L_{d_{min}} = 0$	22, 44, 70 , and 100 Nodes	100
ABAT−4	$L_{d_{max}} = 30$, $L_{d_{min}} = 0$		
ABAT−5	$L_{d_{max}} = 40$, $L_{d_{min}} = 0$		

5.5.3 Comprehensive Analysis of ABAT versions using Statistical Parameters

***ABAT* Versions**	**Measuring Parameters**	**No. of Nodes in MPLS Network**	**No. of *Runs***
ABAT−1			

ABAT−2	*Mean*		
ABAT−3	Standard Deviation	22, 44, 70 , and 100 Nodes	100
ABAT−4	Optimal Value		
ABAT−5			

5.5.4 Comparative Study of ABAT Algorithms with Standard Bat, CBA, PSO, and DEA Algorithms

5.5.4.1 Investigation of ABAT Convergence Curves

Algorithm	No. of iterations	No. of ***Runs***	Objective Functions
ABAT			
BAT		$200, 400, 600, 800,$	Routing Costs
CBA	100	$1000, 1200, 1400,$	Load Balancing Cost
PSO		$1600, 1800, and\ 2000$	
DEA			

5.4.4.2 Mean of Optimal Solutions Obtained Through Different Number of

Algorithm	No. of iterations	No. of ***Runs***	***Means*** Optimal Values
ABAT			
BAT		$200, 400, 600, 800,$	$\frac{\sum_{n=1}^{N} optimal\ values\ in\ each\ Run}{Total\ No. of\ Runs}$
CBA	100	$1000, 1200, 1400,$	
PSO		$1600, 1800, and\ 2000$	
DEA			

5.4.4.3 Data Analysis based on Statistical Techniques

Algorithm	Measuring Parameters	No. of Nodes MPLS Networks	No. of ***Runs***

ABAT			
BAT	*Mean*		
CBA	Standard Deviation	$22, 44, 70$, and 100 Nodes	100
PSO	Optimal Value		
DEA			

Table 5. 2: Parameter Settings for Algorithms

Algorithm	**Parameter**	**Value**
ABAT	Topology	Fully connected
	Minimum and Maximum frequencies(f_{min}, f_{max})	$f_{min} = 1$, $f_{max} = 4$
	β range	$\beta = [0 , 1]$
	Initial velocity limit	10% of dimensional range
	Minimum and Maximum Loudness ranges$(L_{d_{min}} , L_{d_{max}})$	$L_{d_{min}} , L_{d_{max}}$ Changes as Table 5. 2
	α , γ and ε constants	Random values ranges $= [0 , 1]$
BAT	β limits	$\beta = [0 , 1]$
	Minimum and Maximum frequencies (f_{min}, f_{max})	$f_{min} = 0$, $f_{max} = 3$
	Minimum and Maximum Loudness ranges $(L_{d_{min}} , L_{d_{max}})$	$L_{d_{min}} = 1$, $L_{d_{max}} = 5$
	Initial velocity limit	10% of dimensional range
CBA	Minimum and Maximum frequencies (f_{min}, f_{max})	$f_{min} = 0$, $f_{max} = 2$

	ε constants	Value ranges $= [-1, 1]$
	Minimum and Maximum Loudness ranges $(L_{d_{min}}, L_{d_{max}})$	$L_{d_{min}} = 0$, $L_{d_{max}} = 1$
	α and γ constants	Both Value ranges $=$ [0.9 , 0.975]
PSO	Cognitive and social behavior constants (c_1, c_2)	2 , 2
	Weight inertia (ω)	Linear reduction from 0.9 to 0.1
	Velocity limit	10% of dimensional range
DEA	Predefined Probability (PP)	0.1 value

5.5.1 Pareto Front Graph

As previously explained that the bat algorithm has often referred for its inferior exploration ability and because of this, the algorithm may possibly get stuck into local optima with regard to multimodal functions [204]. Concerning these types of circumstances, in which the optimization problem corresponding with multi-objective functions, the Pareto-based technique can potentially be put into practice that balances the two contradictory objective functions and therefore providing solutions in the format of Pareto front [205] - [209] . It should be taken into consideration that updating the loudness level is mandatory during iterations because the bat's echolocation signals (loudness level) decrease as the bat comes closer towards its prey (solution). For that particular reason, $L_{d_{max}}$ and $L_{d_{min}}$ levels must be identified at the initiating stage, however this critical stage can be utilized by offering with those $L_{d_{max}}$ and $L_{d_{min}}$ values that aids algorithm`s efficiency . This idea has not been researched or practiced before. In addition to this, to examine the effectiveness of the proposed *ABAT* for seeking out the Pareto front based solutions; the experiments were therefore carried out for various loudness levels as listed in Table 5. 1. Fig. 5. 1 (a – d) reflects the Pareto front solutions using the various sizes of the MPLS networks. During assessing the Pareto fronts graphs, it is discovered that irrespective

of the variations in *ABAT* versions and network sizes, all *ABAT* versions have successfully establish Pareto based solutions when considering both mentioned objective functions. On top of this, it was also experienced from the results that both objective functions are opposite to each other, as routing cost function decrease with the increase of load balancing cost function regarding *ABAT* versions. As for instance, in Fig. 5. 1 (c), it is often noticed that the load balancing costs increases with the decrease of the routing cost for *ABAT-2* version. Similar performance can be observed as a result of other versions (*ABAT- 1*, *ABAT-3*, *ABAT-4*, and *ABAT-5*). Whilst the other *ABAT* versions graphs are displayed in Fig. 5. 1 (a – d) that indicates similar outcomes. As a consequence, it can be deducted that at various loudness levels, the offered *ABAT* algorithm is a favorable choice for networking administration and this algorithm can auspiciously deliver results as Pareto based solutions for MCOP optimization problems in different scales of MPLS networks.

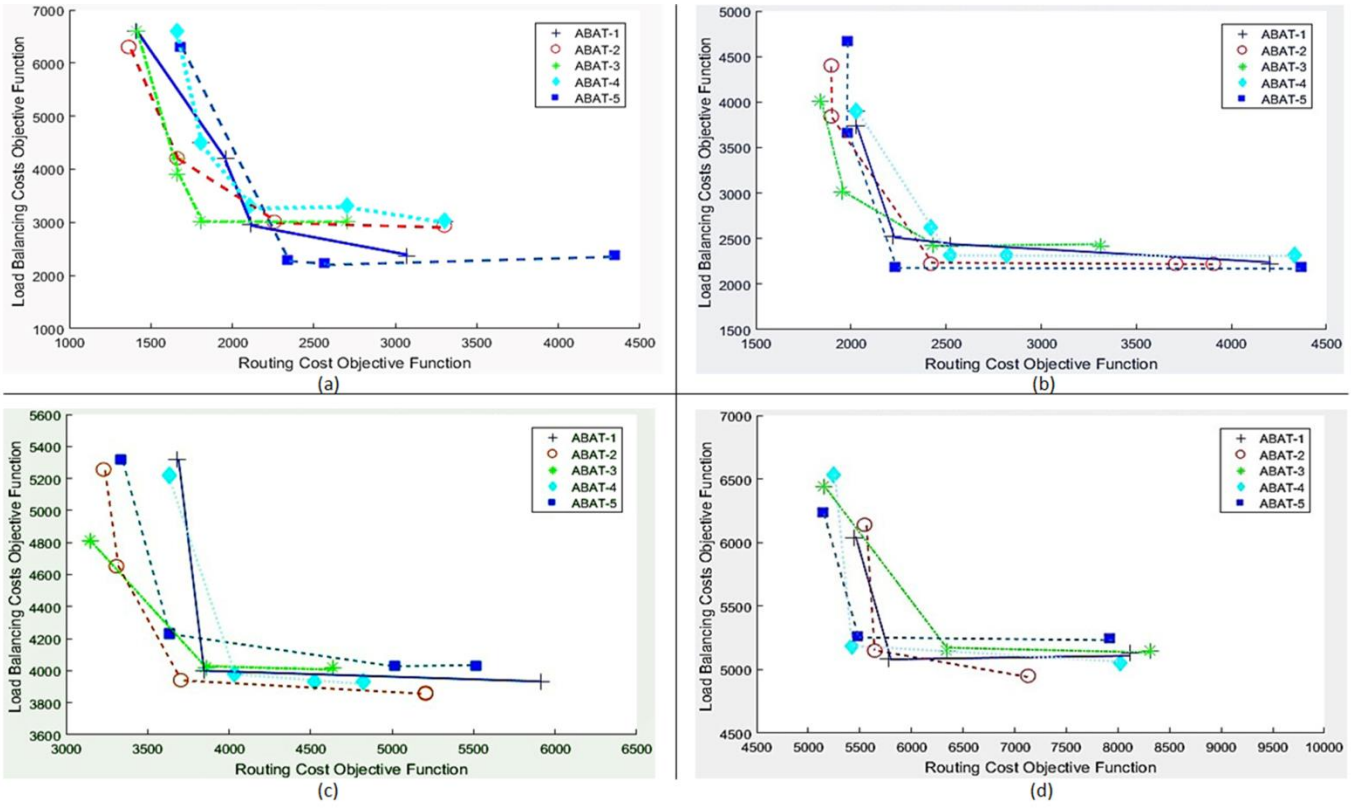


Fig. 5. 1: Pareto based ABAT Algorithm for 22, 44, 70, and 100 nodes MPLS Networks

5.5.2 Investigation of Various Versions of *ABAT* Algorithm

In order to explore the influential role of loudness parameter in the form of proposed *loudness optimizer* in the *ABAT* for the considered optimization problem, the convergence rate per each loudness level is investigated, as presented in Fig. 5. 2 (a − d). The experimental setup is specified in Table 5. 1. The findings collected from Fig. 5. 2 (a − d) recognizes the importance of the loudness parameter, in which the variation of loudness helps to improve the exploration process during a number of iterations. Regarding the considered different network sizes in the experiments, it is observed that the convergence rate of *ABAT-5* is more effective in comparison to the last *ABAT-1*. Technically speaking, this particular development is discussed as follows: whenever the loudness parameter $L_{d_{max}}$ increases contrast to $L_{d_{min}}$, the algorithm converges rapidly in direction of its optimal solution, whereas, the $L_{d_{max}}$ having lower value or close to $L_{d_{min}}$ value, will negatively affect the algorithm's convergence and for that reason, the algorithm will come up with sub-optimal solutions for small to large scales of networks. As an example, Fig. 5. 2 (d), it can comfortably be noticed that *ABAT-1* ($L_{d_{max}} = 3$, $L_{d_{min}} = 0$) has poorer convergence rate or low-quality exploration function around optimal solutions. On the other hand, *ABAT-2* ($L_{d_{max}} = 10$, $L_{d_{min}} = 0$) has more effective convergence rate compare to *ABAT-1*. In identical fashion, ABAT-3 has improve convergence rate versus *ABAT-2*, *ABAT-4* comes with improved results as compared to *ABAT-3* and at the end, *ABAT- 5* ($L_{d_{max}} = 40$, $L_{d_{min}} = 0$) has the most effective convergence rate and as a consequence yields optimal solutions. Very similar results are achieved for all sizes of the available MPLS networks, as mirrored in Fig. 5. 2 (a, b, and c). By exploring the results of Fig. 5. 2 (a − d), it is also revealed that *ABAT-1* (has very low $L_{d_{max}}$ value) not solely deals with low convergence rate but may possibly be trapped in the local minima problem. It is noticed via investigation of results that the algorithm shouldn't be implemented with a very low value of $L_{d_{max}}$ for any network size, because in this scenario, the algorithm will possibly drop into its local optima and will be continuously trapped there till the ending of iterations. These particular results therefore describe an improvement to the claim made in [204].

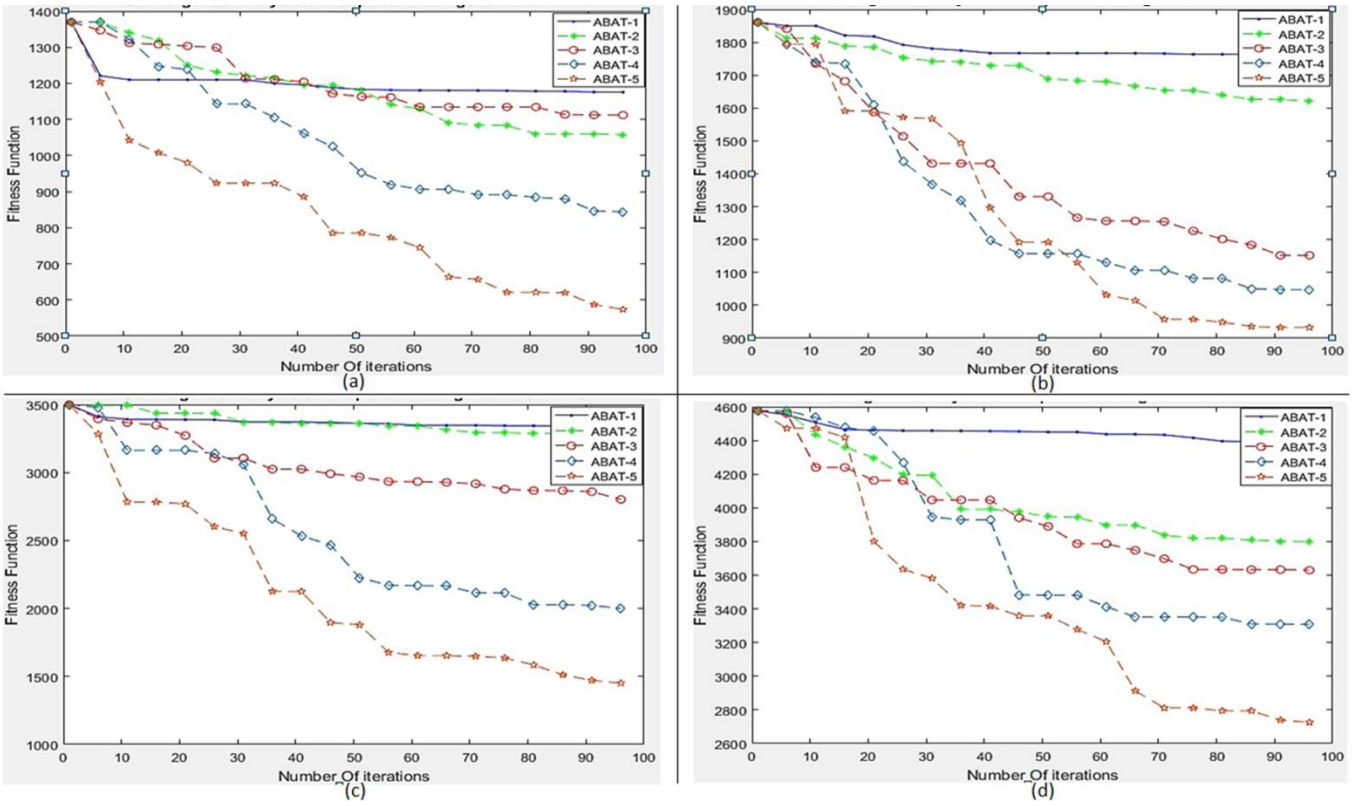


Fig. 5. 2: Convergence Analysis of Proposed ABAT in 22, 44, 70, and 100 nodes networks

5.5.3 Comprehensive Analysis of *ABAT* versions using Statistical Parameters

By using MATLAB R2016a, the presented *ABAT* algorithm was going to be implemented for MCOP based optimization problem in MPLS networks along with different range of nodes. To analyze the algorithm efficiency, the algorithm had been implemented in MATLAB for 100 times (annotated as $100\,Runs$). Statistical variables (parameters) which include Mean, SD (Standard Deviation), MR (Minimum Routing Cost), and MB (Minimum Load Balancing Cost) as the optimal solutions, are then recorded to examine the algorithm overall performance, provided in Table 5. 3 – Table 5. 6 comprise the measurements collected.

Table 5. 3: Analysis of ABAT Algorithm for 22 Nodes network

Routing Costs Objective Functions					
	$\boldsymbol{ABAT-1}$	$\boldsymbol{ABAT-2}$	$\boldsymbol{ABAT-3}$	$\boldsymbol{ABAT-4}$	$\boldsymbol{ABAT-5}$
Mean	2222.8	2138	1857.7	1640.4	1381.7
SD	215.6606	224.1177	279.7278	310.1194	365.5272
MR	1754.9	1634.9	1005.4	909.84	150.08
Load Balancing Costs Objective Functions					
	$\boldsymbol{ABAT-1}$	$\boldsymbol{ABAT-2}$	$\boldsymbol{ABAT-3}$	$\boldsymbol{ABAT-4}$	$\boldsymbol{ABAT-5}$
Mean	2727.2	2492.4	2214.1	2012.6	1787.3
SD	332.0824	374.3274	420.8805	453.7138	507.9096
MR	1691.7	1518.6	1185.7	1052.2	403

Table 5. 4: Analysis of ABAT Algorithm for 44 Nodes network

Routing Costs Objective Functions					
	$\boldsymbol{ABAT-1}$	$\boldsymbol{ABAT-2}$	$\boldsymbol{ABAT-3}$	$\boldsymbol{ABAT-4}$	$\boldsymbol{ABAT-5}$
Mean	3887.4	3587	3282.1	3035.7	2645.3
SD	237.2402	307.8458	335.9810	403.6638	543.3792
MR	3149.3	2702.9	2345.3	1781.7	1257.9
Load Balancing Costs Objective Functions					
	$\boldsymbol{ABAT-1}$	$\boldsymbol{ABAT-2}$	$\boldsymbol{ABAT-3}$	$\boldsymbol{ABAT-4}$	$\boldsymbol{ABAT-5}$
Mean	4258.5	4062.5	3736.7	3434.3	2833
SD	510.0274	409.4460	568. 3467	590.7590	632.9439
MR	2927.1	2845	2088.3	1767.6	1355.6

Table 5. 5: Analysis of ABAT Algorithm for 70 Nodes network

Routing Costs Objective Functions					
	$\boldsymbol{ABAT-1}$	$\boldsymbol{ABAT-2}$	$\boldsymbol{ABAT-3}$	$\boldsymbol{ABAT-4}$	$\boldsymbol{ABAT-5}$
Mean	3124.3	2792.5	2415.8	1954.1	1607.2
SD	188.9614	222.1953	279.4749	417.1811	563.9224
MR	2450.1	2194.4	1715.8	1005.1	301.41
Load Balancing Costs Objective Functions					
	$\boldsymbol{ABAT-1}$	$\boldsymbol{ABAT-2}$	$\boldsymbol{ABAT-3}$	$\boldsymbol{ABAT-4}$	$\boldsymbol{ABAT-5}$
Mean	3383.7	3013.4	2618.2	2161.3	1678
SD	283.0054	241.0311	341.1794	490.7355	634.5006
MR	2701.4	1952.3	1883.4	900.2	102.37

Table 5. 6: Analysis of ABAT Algorithm for 100 Nodes network

Routing Costs Objective Functions					
	$\boldsymbol{ABAT-1}$	$\boldsymbol{ABAT-2}$	$\boldsymbol{ABAT-3}$	$\boldsymbol{ABAT-4}$	$\boldsymbol{ABAT-5}$
Mean	4572.3	4129.8	3664.5	3223.8	2869.5
SD	197.7087	244.5379	372.9312	505.0768	539.2802
MR	4061.1	3422.8	2537.4	1763.1	1585.8
Load Balancing Costs Objective Functions					
	$\boldsymbol{ABAT-1}$	$\boldsymbol{ABAT-2}$	$\boldsymbol{ABAT-3}$	$\boldsymbol{ABAT-4}$	$\boldsymbol{ABAT-5}$
Mean	4802.9	4462.1	3938.1	3564.3	2974.4
SD	351.6230	365.0745	453.3740	512.0764	680.9660
MR	3971.8	3205.4	2717.9	2453.3	1280.8

Evaluating these results, the *ABAT-5* values can be proclaimed as the optimal solutions for $22, 44, 70$ and 100 nodes networks. In essence, *ABAT-5* (using the maximum value of $L_{d_{max}}$ contrast to other remaining *ABAT* versions), provided minimum (Mean, routing and load balancing costs) values. Please note, the optimal solutions are assumed as minimum values. *ABAT-1* delivered the worst solutions ($i.e.$, maximum values of Mean, routing and load balancing costs) regarding both objective functions for all scales of network optimization. Having said that, it is worth noting that the standard deviation (indicating the deviation from the Mean values) of *ABAT-1* is minimum, while *ABAT-5* has maximum. In the simulations it is also discovered that standard deviation values increase from the *ABAT-1* version to *ABAT-5* version. Very much the same results have been noticed for all regarded scales of networks. In depth analysis of Table 5. 3 – Table 5. 6, it came to the conclusion that the proposed *ABAT* algorithm delivers optimal solutions for larger $L_{d_{max}}$ values ($i.e$ *ABAT-5*), and comes with minimum (optimal) routing and load balancing costs for any scale of MPLS network after 100 iterations. Comparable outcomes are observed for minimum Mean values (as optimal solutions) for *ABAT-5* versions. Whereas $L_{d_{max}}$ along with its growing values (from *ABAT-1* to *ABAT-5*) will produce maximum standard deviation values for *ABAT-5*. In addition, for the large networks having maximum $L_{d_{max}}$ value, there is a maximum possibility of distinct solutions, which indicates the stochastic feature of the algorithm. Last but not least, depending on Fig. 5. 2 (a – d) and the results of Table 5. 3 – Table 5. 6 results, it can be concluded that in *ABAT-5* (having maximum $L_{d_{max}}$) value results helps to improve exploration plan of action of the proposed *ABAT* algorithm. On the other hand the *ABAT* version with lower $L_{d_{max}}$ value provides the worst performance regarding poor convergence rate from a smaller to a bigger scale of MPLS networks.

5.5.4 Comparative Study of *ABAT* Algorithms with Standard Bat, CBA, PSO, and DEA Algorithms

To conclusively illustrate the crucial role of the *LO* in the projected *ABAT*, a wide variety of experiments were performed to validate the effectiveness of *LO* towards the generation of the optimal solutions (for both objective functions). The foundation of the following experiments

were to put together a comprehensive comparative research between the offered *Adjustable Bat algorithm* and some other well-known metaheuristic algorithms which include standard Bat, CBA, PSO, and DEA algorithms. Performance experiments were carried out by examining routing and load balancing costs objective functions. This comparative investigation was conducted in three sets of experiments which are briefly explained in the following sections of the chapter. Fig. 5. 3 and Fig. 5. 4 depict the comparative study convergence curves, while Fig. 5. 5 and Fig. 5. 6 present the Mean Optimal values for both objective functions. Aside from this, the collected data of *ABAT* and other reported algorithms (Bat, CBA, PSO, and DEA) are empirically reviewed by using statistical parameters, presented in Table 5. 7 – Table 5. 10.

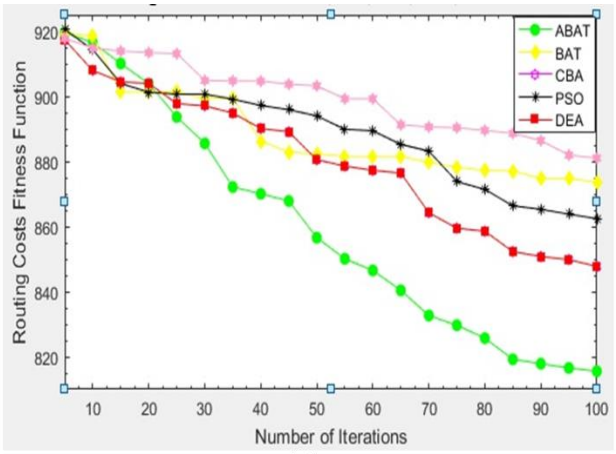
5.5.4.1 Investigation of *ABAT* Convergence Curves

Fig. 5. 3 and Fig. 5. 4 contrasts the convergence curves gained by implementing the *ABAT* algorithm with *standard* Bat, CBA, PSO, and DEA algorithms for the optimization of $22, 44, 70$ and 100 MPLS nodes networks. The experimental setup variables (parameters) are shown in Table 5. 1. As of collected results for both objective functions, as displayed in Fig. 5. 3 and Fig. 5. 4, it looks obvious that the convergence curves of *ABAT* outclass its competitive algorithms (*standard* Bat, CBA, PSO, and DEA).

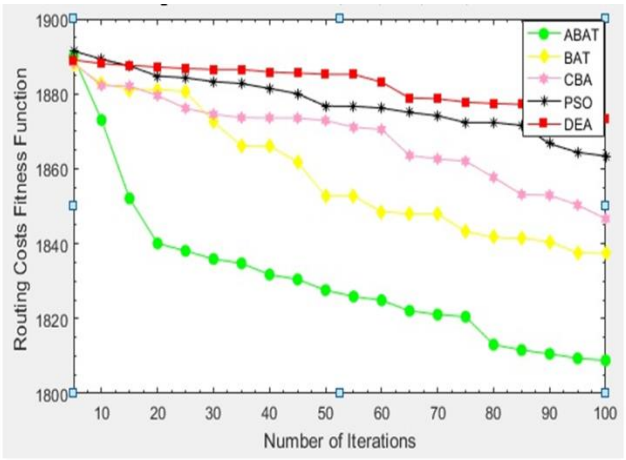
However, *standard* Bat, CBA, PSO, and DEA algorithms have actually come with poor convergence and are unsuccessful to generate optimal solutions contrast to *ABAT*. It was also witnessed in the results that the weak convergence of these algorithms (*standard* Bat, CBA, PSO, and DEA) resulted in repetitive and suboptimal solutions for a number of iterations. As an instance, in Fig. 5. 3 (c), the DEA algorithm fails to converge in between 15th and 50th iterations and as a result, brings repetitious/ suboptimal solutions. The identical routines can be detected for the standard Bat, CBA, and PSO algorithms which do not succeed to converge anywhere between 60^{th} and 70^{th}, 30^{th} and 60^{th} or 50^{th} and 60^{th} iterations, respectively. In a similar manner, the outcomes of Fig. 5. 3 (a, b and d) were received, in which standard Bat, CBA, PSO, and DEA algorithms drastically unsuccessful to have better convergence rate compared to *ABAT* algorithm. Fig. 5. 4 (a, b, c, and d) portraying the results for load balancing costs functions, which demonstrates the similar behavioral routine received by the offered *ABAT* in contrast to the competitive algorithms. According to the results in Fig. 5. 4, an excellent convergence rate is

accomplished by the *ABAT* in comparison to the *standard* Bat, CBA, PSO, and DEA algorithms. As an illustration, in Fig. 5. 4 (d), the *ABAT* algorithms outperforms its competing algorithms with regards to far better convergence curves in direction of optimal solutions for a provided number of iterations. The *standard* Bat, CBA, PSO, and DEA algorithms are seriously affected from problems of repetitive values (as sub-optimal solutions) which influence their convergence rate. In a similar way, Fig. 5. 4 (a, b, and c) show that the *ABAT* convergence rate is more effective when compared with the other stated algorithms. Exploration and exploitation problems *"disturb"* the convergence rate of the algorithms with the exception of *ABAT*, as observed in Fig. 5. 3 and Fig. 5. 4.

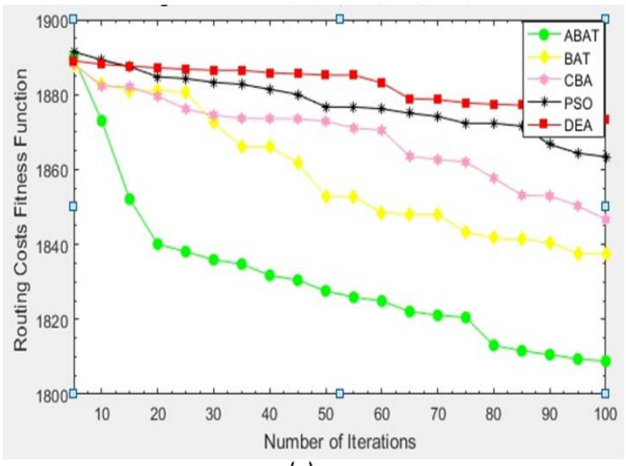
All these results confirm that the offered *ABAT* algorithm manages the exploration and exploitation problem considerably better by using *LO* and therefore, come with an excellent convergence rate when put side by side with various other mentioned algorithms in the experiments.



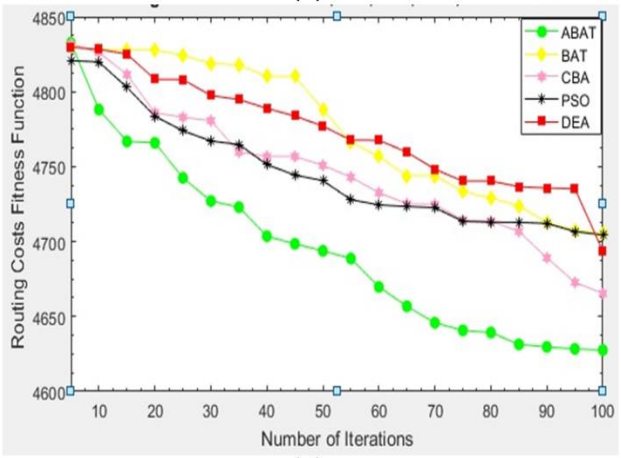
(a)



(b)



(c)



(d)

Fig. 5. 3: Comparison of *ABAT* & BAT, CBA, PSO, DEA for Routing Costs Function in 22, 44, 70, 100 Nodes

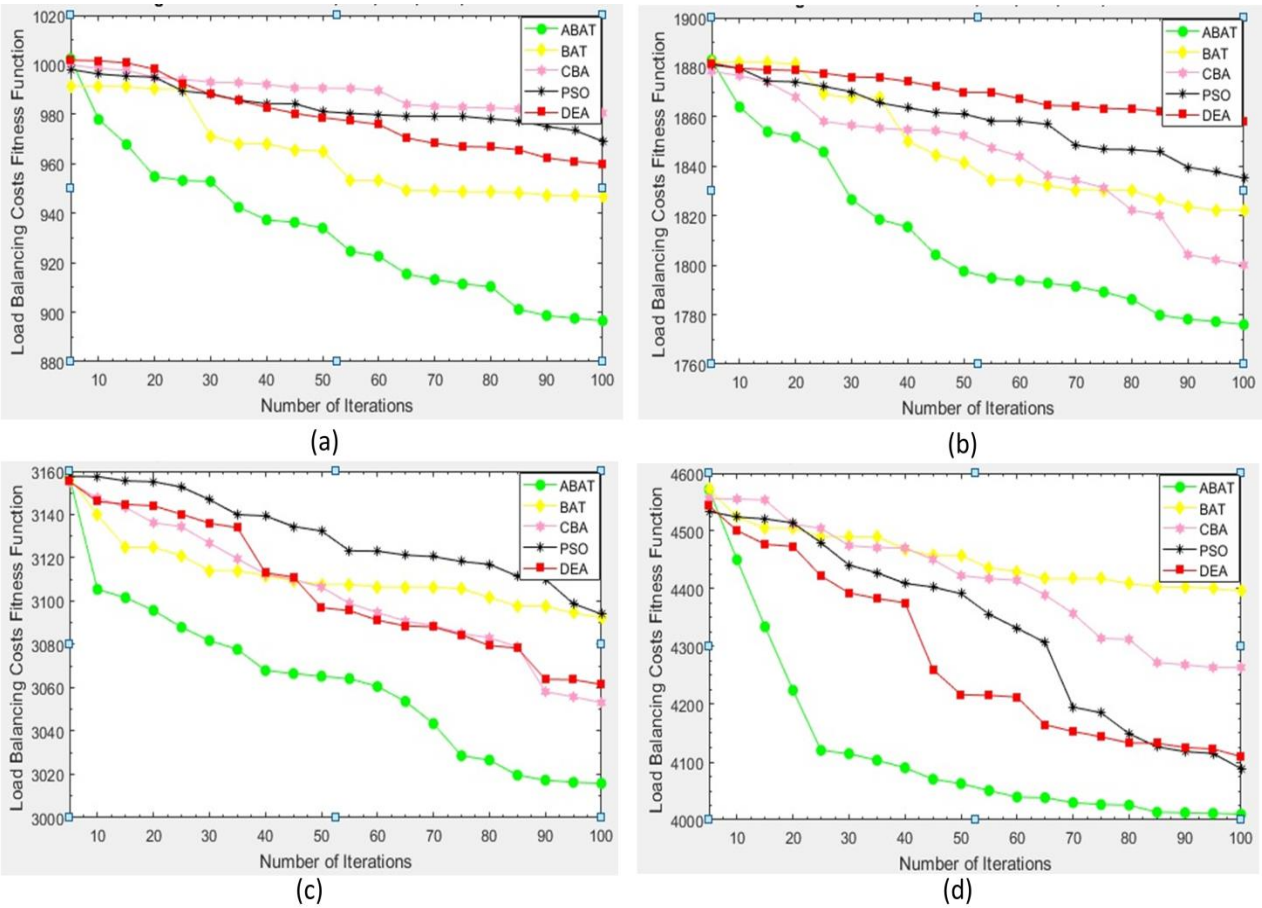


Fig. 5. 4: Comparison of *ABAT*, BAT, CBA, PSO, DEA of Load Balancing Function in 22, 44, 70, 100 Nodes

5.4.4.2 Mean of Optimal Solutions Obtained Through Different Number of *Runs*

To broaden the research about *ABAT* algorithm, another experimental setup is designed (see Table 5. 1), in which all the algorithms taken into account were operated for a various number of times (termed as *Run*) to collect statistics related data. The *Mean* values stand for the optimal *Mean* values following allocated number of *Runs* and are then showcased in Fig. 5. 5 and Fig. 5. 6, in order to do a comparison of *Mean* optimal values of *ABAT* and *standard* Bat,

CBA, PSO, and DEA algorithms, correspondingly. The aim of this analysis was to create a comparative investigation between the mentioned algorithms and *ABAT*. Based on the results provided in Fig. 5. 5 and Fig. 5. 6, it is found that the *ABAT* come with optimal solutions for *Mean* values for $22, 44, 70,$ and 100 MPLS nodes network in accordance with both objective functions. It needs to be noted here, that minimum *Mean* values are taken into account of optimal solutions in the experiment. Both in Fig. 5. 5 and Fig. 5. 6, *ABAT* delivered minimum Mean values for all number of iterations. This particular experiment verified that the recommended *ABAT* algorithm provides optimal solutions contrast to its *"opponents"* algorithms, the *standard* Bat, CBA, PSO, and DEA, no matter what the number of simulations and continuously generated the same outcomes from smaller to wide scale MPLS networks.

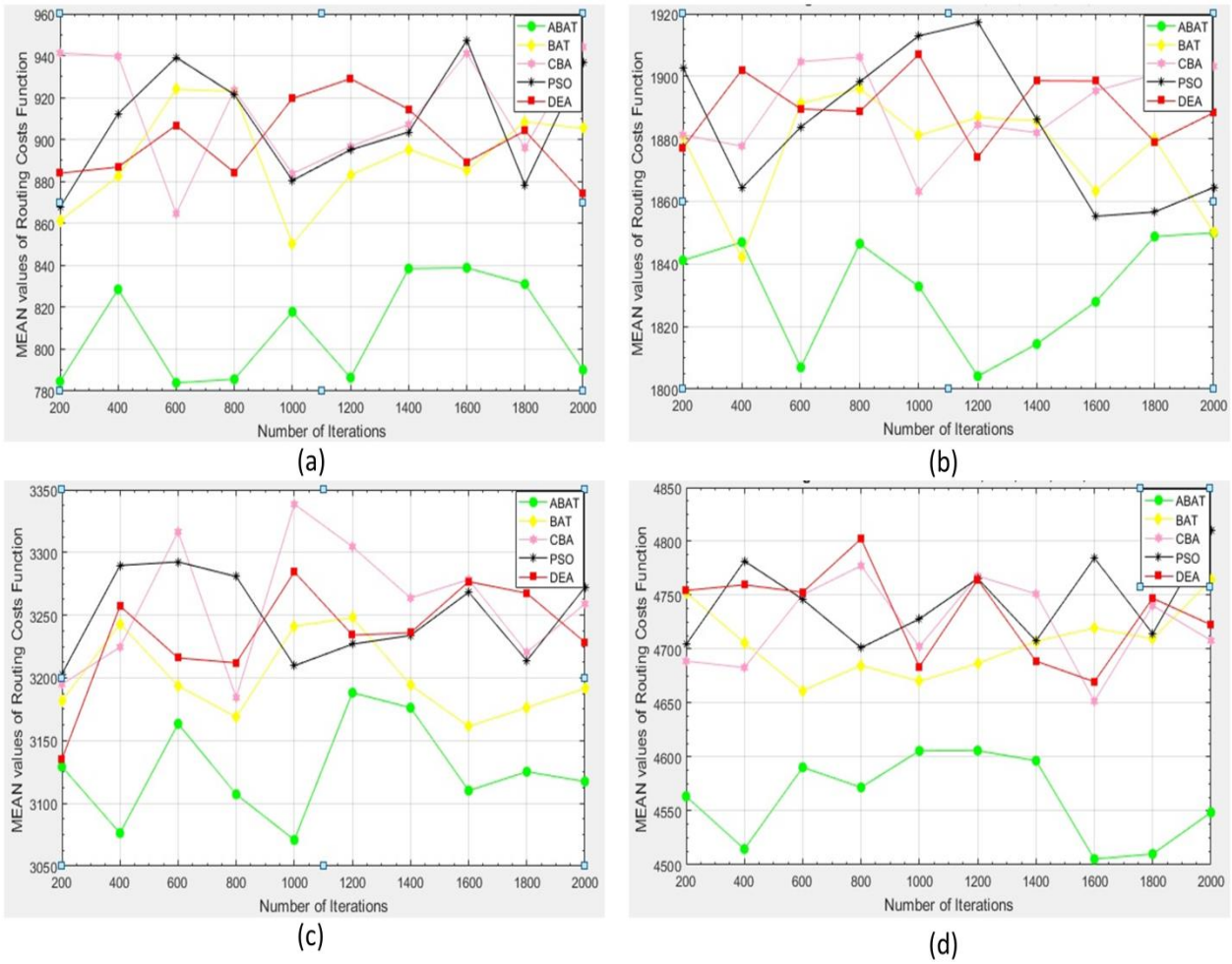


Fig. 5. 5: Mean Routing Function for analysis of ABAT, BAT, CBA, PSO, DEA on 22, 44, 70, 100 Nodes

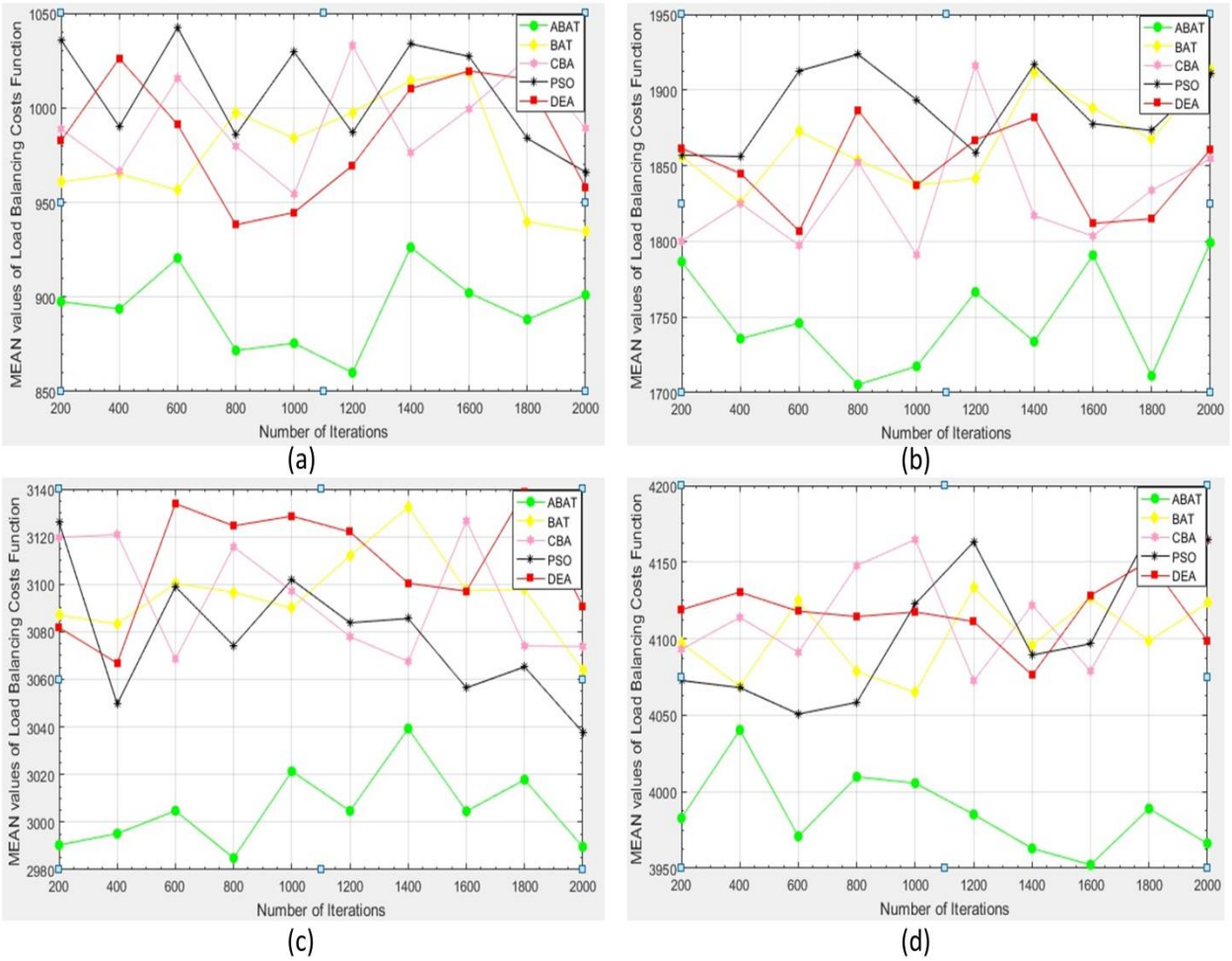


Fig. 5. 6: Mean Load Balancing Function for ABAT, BAT, CBA, PSO, DEA on 22, 44, 70, 100 Nodes

5.4.4.3 Data Analysis based on Statistical Techniques

To facilitate further comparison with critical analysis, the experiments continued by compiling the data as a result of 100 times iterations for both objective function. The compiled data was going to be analyzed by using statistical parameters which include *Mean*, Standard Deviation (SD), and Optimal solutions (minimum routing cost (MRC) and load balancing cost (MLBC)) for $22, 44, 70,$ and 100 nodes MPLS networks. The compiled data is received in the form of results which are then structured in Table 5. 7 – Table 5. 10 for both objective functions.

Table 5. 7: Comparative Study of *ABAT* & BAT, CBA, PSO, DEA Algorithms for 22 Nodes Network

Routing Costs (RC) and Load Balancing Costs (LBC) Objective Functions						
Algorithms	**MRC**	***Means* (RC)**	**SD (RC)**	**MLBC**	***Means* (LBC)**	**SD(LBC)**
ABAT	906.04	1219.99	206.04	582.45	851.78	164.85
BAT	1502.32	1895.48	240.32	1053.96	1380.71	182.74
CBA	1673.20	2068.43	246.77	922.61	1334.73	251.98
PSO	1125.68	1400.47	179.80	894.94	1449.03	320.34
DEA	1811.27	2248.18	306.74	1350.10	2086.65	423.45

Table 5. 8: Comparative Study of *ABAT* & BAT, CBA, PSO, DEA Algorithms for 44 Nodes Network

Routing Costs (RC) and Load Balancing Costs (LBC) Objective Functions						
Algorithms	**MRC**	***Means* (RC)**	**SD (RC)**	**MLBC**	***Means* (LBC)**	**SD(LBC)**
ABAT	2521.94	2936.03	215.35	2651.27	3235.28	338.11
BAT	3932.76	4366.60	259.25	3855.41	4291.89	266.17
CBA	2965.05	3422.29	288.24	2929.63	3475.10	326.72
PSO	2815.68	3243.38	224.63	3129.37	3559.60	252.09
DEA	4015.30	4295.31	171.54	3123.98	3866.59	403.33

Table 5. 9: Comparative Study of *ABAT* & BAT, CBA, PSO, DEA Algorithms for *70* Nodes Network

Routing Costs (RC) and Load Balancing Costs (LBC) Objective Functions						
Algorithms	**MRC**	***Means* (RC)**	**SD (RC)**	**MLBC**	***Means* (LBC)**	**SD(LBC)**
ABAT	5173.08	5656.22	278.69	5455.48	6035.24	346.27
BAT	6836.80	7538.14	421.51	5662.95	6335.85	394.29
CBA	5525.37	6292.02	455.77	6058.53	6689.46	389.82
PSO	5791.05	6351.15	353.56	5892.58	6519.28	375.48
DEA	6925.95	7610.13	351.21	6116.69	6845.79	400.41

Table 5. 10: Comparative Study of *ABAT* & BAT, CBA, PSO, DEA Algorithms for 100 Nodes Network

Routing Costs (RC) and Load Balancing Costs (LBC) Objective Functions						
Algorithms	**MRC**	***Means* (RC)**	**SD (RC)**	**MLBC**	***Means* (LBC)**	**SD(LBC)**
ABAT	7734.29	8359.70	387.95	7671.31	8291.48	392.61
BAT	7933.28	8812.44	546.30	7905.28	8694.99	446.22
CBA	8003.59	8997.32	609.33	8067.90	8918.60	532.62
PSO	7923.68	8969.13	590.75	7916.65	8894.25	559.64
DEA	8004.35	8880.00	471.66	8123.17	9068.55	563.36

It was soon revealed that the projected ABAT algorithm provided optimal solutions on the subject of minimum routing and load-balancing costs, and minimum Mean and standard deviation values in contrast to *standard* Bat, CBA, PSO, and DEA algorithms. When looking at the data received from a *"network optimization perspective"*, the final results reveals the information that the scale of MPLS network can play a major factor in network management. It is found, that as the network size grows the algorithm convergence is going to be extremely

affected. By checking out the statistics in Table 5. 7 – Table 5. 10 for both functions, the *Mean* and Standard Deviation values increase for big network size. It should be taken into note here that in the experiments the lower limit values are regarded as optimal values. For instance, taking a look at *ABAT* algorithm in Table 5. 7 (22 nodes network), the minimum routing costs: 906.04 is the optimal (minimum) value contrast to the 2521.94 routing costs in Table 5. 8 (44 nodes network). The same outcome was obtained for on increased network size (Table 5. 9 and Table 5. 10), wherein for the *ABAT*, the routing costs 5173.08 for a 70 nodes network (Table 5. 9) is the optimal value contrast to the 7734.29 routing costs value of a 100 nodes network (Table 5. 10). A very similar routine is noted for the minimum (optimal) and *Mean* values of both load balancing costs function. For a comparison, the exact same pattern is also put into practice by *standard* Bat, CBA, PSO, and DEA algorithms, which provided sub-optimal solutions comparing to the *ABAT* algorithm (having optimal solutions from Table 5. 7 – Table 5. 10. Having said that, Standard deviation values are random and, the network size doesn't necessarily influence this parameter.

5.6 Summary

The chapter presented an optimization model for MPLS network by considering routing delay and load balancing costs objective function. The MPLS optimization problem is formulated as MCOP optimization problem. For the given optimization version of MPLS networks, the chapter come up with a novel model of an *Adjustable Bat algorithm*, integrated with *loudness optimizer* parameter for MCOP optimization in MPLS network. *ABAT* algorithm was tailored to discover both, the optimal and non-dimensional solutions as available paths in the form of Pareto frontier curves for the given objective functions. *ABAT* was simulated operating upon a variety of *ABAT* versions $i.e.$, *ABAT-1*, *ABAT-2*, *ABAT-3*, *ABAT-4*, and *ABAT-5*. Information coming from the results reported that the offered Pareto based *ABAT* converged rapidly and provide an optimal solution when $L_{d_{max}}$ value is maximum. In addition to this, an overall performance comparability between the Pareto based *ABAT* and *standard* Bat, CBA, PSO, and DEA algorithms is demonstrated in various experiments while applying to different MPLS network sizes. The findings obtained in the form of results proves the superiority of the *ABAT* relating to the convergence rate enhancement and it is also discovered that the *LO* parameter can play a

crucial part for enhancing the convergence rate of the *ABAT* algorithm. It has also verified from the results that *ABAT-5* (with its higher $L_{d_{max}}$) delivers improve results as optimal solutions but that the standard deviation will increased while contrasted to *ABAT-1* (with its lower $L_{d_{max}}$).

Chapter 6

Hybrid Model of Bat and Dolphin Echolocation-based Algorithm for Solving Optimization Problem

Dolphin Echolocation algorithm is proposed in 2017, which relies on its echolocation parameter for prey searching in the problem domain analogous to Bat algorithm. In other words, both algorithms principally depend on their echolocation feature for prey searching. Nevertheless, the significance of this feature has never been debated and highlighted before. Furthermore, both algorithms have some limitations that cause them to converge slowly and thus limit them to produce sub-optimal solutions for various optimization problems. The work in this chapter will discuss the echolocation feature of both algorithms and also highlight the slow convergence problem. The focus of the chapter is to propose a hybrid model of bat and dolphin echolocation algorithm for the MPLS optimization which is a complex optimization problem. The state-of-the-art literature review for network optimization using metaheuristic algorithms is discussed in the chapter. Then, the optimization model of MPLS network is mathematically formulated. For the given optimization model, a proposed hybrid model as an algorithm will be applied and compare it`s performance with other metaheuristics. For further verification of algorithm`s performance, the proposed algorithm will also be investigated over benchmark mathematical optimization functions in the last section of the chapter along with conclusion.

6.1 Introduction

A large number of researchers, working in the field of AI-based optimization algorithms have been fascinated by nature, thus they express their inspiration by developing or working on nature-inspired algorithms. Nature-inspired algorithms belong to a family of metaheuristic algorithms formulated by learning from nature. These algorithms can be further classified on the

basis of biology, chemistry and physics-inspired algorithms [210] – [213]. Within these sub-classes, bio-inspired algorithms have got the attention of researchers working in the metaheuristic field. Some of the popular algorithms here may include the PSO [79], bat [81], ACO [82], DEA [83], GWO [84] and glow-warm optimization algorithm [214]. Bio-inspired algorithms can further be categorized into swarm-intelligence-based and non-swarm-intelligence-based algorithms [215] – [216]. Irrespective of these algorithm groups and nature, metaheuristic algorithms normally suffer from the exploration and exploitation problem. This is described as the complex task of balancing between the exploitation and exploration capabilities of the algorithm [217] – [219]. This means that the endeavor continues in this particular field of study of proposing new or modified hybrid-model metaheuristic algorithms.

In recent years, two bio-inspired algorithms have been proposed in [81] and [83], known as the BA (Bat Algorithm) and DEA (Dolphin Echolocation Algorithm) respectively. Although BA is SI-based and DEA is not SI-based algorithm, both algorithms rely on the common feature of echolocation for the purpose of prey searching (finding optimal solutions) [81], [83], and [220]. The bat echolocation process uses sound waves and echoes in space while the DEA algorithm uses a form of echolocation known as sonar, usually underwater. Comprehensive research has been presented regarding the similarities and differences between BA and DEA [221] – [224]. Both algorithms have been substantially accepted due to producing effective optimal solutions for various optimization problems. Nevertheless, both (BA and DEA) suffer from the exploration and exploitation problem and various attempts have been made to improve their performance [204], [225]. The research in this chapter focuses on the hybridizing modeling of the algorithm concept in order to solve premature convergence, producing sub-optimal solutions usually caused by the effect of the exploration/ exploitation problem. This chapter proposes a hybrid model of the BA and DEA algorithms. The aim was to emphasize the powerful features of both algorithms and thus develop a new hybrid model. This proposed hybrid algorithm takes advantage of the echolocation of both BA and DEA.

6.2 Echolocation-based Metaheuristic Algorithms

Griffin [226] was the first researcher who used the term *"echolocation"*. This depicts the searching ability of flying bats to detect obstacles and find their prey by simply listening to the echoes coming from high frequency clicks that they release. Some mammals and birds rely on echolocation when searching for prey and navigating locations. One of the best echolocation-dependent marine mammals is the dolphin[227] . Both the BA and DEA algorithms rely on the echolocation technique to search for their prey's position in the searching space. The section below briefly describes each algorithm.

6.2.1 Bat Algorithm

In 2010, Yang came with new metaheuristic algorithm named bat algorithm. Bat algorithm (BA) belongs to the family of swarm-intelligence algorithms, which uses echolocation system of the micro bats.. As a metaheuristic algorithm, BA imitates the searching behavior of bats using echolocation system for finding prey (optimal solution) [81],[169]. BA algorithm is already briefly discussed in chapter 5, therefore, in this section the algorithm will be shortly reviewed with the support of its pseudo code as given in *Algorithm* 6. 1.

Algorithm 6. 1: Pseudo code of Bat Algorithm

INPUT:

1: initialize the objective function as $f(x)$

2: initialize random bat population of $x_i = \{x_1, x_2, x_3, \ldots . x_n\}$

3: ***for*** each bat x_i in the swarm ***do***

4: initialize random velocities v_i, pulse-rate P_r^t and loudness L_d parameter

5. initialize frequency f_i using Eq. (5.1)

6. ***end***

7. ***Repeat*** till number maximum number of iterations

8. ***for*** each bat x_i in the swarm ***do***

9. update new solutions as $f_i^{t+1}, v_i^{t+1}, x_i^{t+1}$ through Eq. (5.1), Eq. (5.2), and Eq. (5.3)

10. ***if*** $rand > P_r^t$

11. update Local best as $x_{i,Local-best}^t$ using Eq. (5.4)

12. ***end***

13. ***if*** $f_{x_{old}} > f_{x_{new}}$

14. updated solution of $x_i^t = x_i^{t+1}$

15. update $f_{x_i} = f_{x_{new}}$

16. decrease L_d^t and increase P_r^t through Eq. (5.5) and Eq. (5.6)

17. ***end***

18. ***end***

19. ***until*** *the termination criteria is met*

OUTPUT:

20. ***Rank*** *the bats based on solution and return the current best bat to the swarm*

6.2.2 Dolphin Echolocation Algorithm

In 2013, a nature-inspired metaheuristic algorithm influenced by the echolocation system was proposed by Kahev and Farhoudi, given the name the dolphin echolocation algorithm [83]. With the guidance of a bio-sonar system, dolphins detect, distinguish between and move towards their prey. Dolphins generate a special sound in the form of clicks. When a click strikes its prey, the sound-wave energy is reflected back to the dolphin. The time interval between the click and

echo assists the dolphin in calculating the distance to its prey. By making use of the echolocation feature, the dolphin first explores the searching space and then restricts the trace to discover the exact position of the prey. This particular strategy facilitates the dolphin by limiting the random search space proportional to the distance to its prey. The dolphin produces high frequency clicks, where the first click is used for targeting searching while the other click helps in terms of distance estimation between the dolphin and its prey. In the DEA, there are two stages used for prey or optimal solution searching. The first stage is to conduct a global search in the searching space (to look for unexplored regions) while the second stage concentrates on searching for better solutions compared to the previously searched solutions. In other words, the global search gradually changes to local search over the course of the algorithm iterations. In DEA, the algorithm operator defines a curve according to which the optimization convergence is managed, and the algorithm parameters will be set to follow along the curve. This means that for every single variable, there are a variety of alternatives in the feasible region. During the iterations, the algorithm follows the best alternative according to the predefined convergence curve. DEA defines the CF (Convergence Factor) index in order to control the exploration and exploitation process. With the assistance of this curve, the convergence criterion influences the algorithm; therefore the convergence of the algorithm has become less dependent on the parameters involved [83].

Before starting optimization process with DEA, the first step is the sorting of searching space by using *"ordering searching space rule"*. During initial searching processing of the DEA, the algorithm employs this rule, in which random variables are initialized in searching space. For optimizing each variable, sort the alternatives in either ascending or descending order matrix. Using this technique, for j^{th} variable (as columns in searching matrix), vector A_j having length LA_j is created. LA_j contains all possible alternatives of j^{th} variable and organize these vectors next to each other in the form of columns in matrix. $Matrix\ Alternatives_{MA*NV}$ matrix is created, where $MA(Maximum\ number\ of\ Alternatives)$ is $\max(LA_j)_{j=1:NV}$ having $NV(number\ of\ variables)$. A curve should be assigned, according to which convergence factor (CF) will change during algorithm`s loops. The changes of CF will be change according to the following curve Eq.(6. 1) [83]:

$$PP(loop_i) = PP_1 + (1 - PP_1)\frac{loop_i^{power} - 1}{(number\ of\ loops)^{power} - 1} \quad (6.\ 1)$$

Where;

$PP \rightarrow$ Predefined Probability

$PP_1 \rightarrow$ CF during the first loop, in which the solutions are selected randomly

$loop_i \rightarrow\ i^{th}$(current) loop

$power \rightarrow$ Degree of the curve

$number\ of\ loops \rightarrow$ Total number of loops in which the algorithm will reach to its convergence point. The steps followed by DEA can be explained as follow;

Step 6. 1:

Initialize the random number of locations NL in searching space for DEA. During this stage, a matrix will be generated as $L_{NL\times NV}$, where NV is for number of variables or dimension for each location.

Step 6. 2:

Compute predefined probability PP according to (6. 1).

Step 6. 3:

Compute the fitness function or objective function f_x for the given locations.

Step 6. 4:

From $Matrix\ Alternatives_{MA*NV}$, find the position of the locations $L_{(i,j)}$ in each j^{th} column and name it as A. Therefore, A specifies the index of $L_{(i,j)}$ value in $Matrix\ Alternatives_{MA*NV}$.

Step 6. 5:

Initialize the Effective radius R_e for each $L_{(i,j)}$ in $Matrix\ Alternatives_{MA*NV}$. It is recommended to choose 1 / 4 of the searching space.

Step 6. 6:

Compute the accumulative fitness function according to (6. 2).

$$AF_{(A+k)j} = \frac{1}{R_e} * \ (R_e - \ |k|)\ f_{x_i} + \ AF_{(A+k)j} \qquad , \qquad [-R_e\ ,\ R_e] \tag{6. 2}$$

Step 6. 7:

Find the best location in the matrix of $Matrix\ Alternatives_{MA*NV}$ for each j^{th} column. In other words, find the alternatives assigned to the variables and let their $AF = 0$.

Step 6. 8:

For the variables of $j_{i=1:NV}$, compute the probability P_{ij} of the alternatives $(Matrix\ Alternatives_{MA*NV})$ as $i_{i=1:AL_j}$

$$P_{ij} = \frac{AF_{ij}}{\sum_{i=1}^{AL_j} AF_{ij}} \tag{6. 3}$$

Step 6. 9:

Assign probability equal to PP according to the procedure mentioned in *Algorithm* 6. 2 (from line $32:40$). For optimization, DEA follow the main steps are described in the form of pseudo code in *Algorithm* 6. 2: Pseudo code of Dolphin Echolocation Algorithm.

Algorithm 6. 2: Pseudo code of Dolphin Echolocation Algorithm

---

INPUT:

initialize random locations as matrix of $NL \times NV$

Initialize the $Matrix\ Alternatives_{MA*NV}$

initialize power degree as $Power = 1$

initialize maximum number of loops as $Loops\ Number$

initialize predefined probability as $PP_1 = 0.11$

initialize $-R_e$ and R_e within the range of any defined integers

compute the initial value of PP using (6. 1)

REPEAT:

While $No.of\ loop < Max.no.of\ loops$ ***do***

    for $i = 1: NL$

        compute the fitness function as f_x

    end

    for $i = 1: NL$

        for $j = 1: NV$

            find location position $L_{(i,j)}$ from j^{th} column of $Matrix\ Alternatives_{MA*NV}$, name as A

            for $k = -R_e : R_e$

                compute $AF_{(A+k)j}$ using (6. 2)

            end

```
        end
    end
    for j = 1: col from Matrix Alternatives_{MA*NV}
        for i = 1: row from Matrix Alternatives_{MA*NV}
            if i = Best Location_j
                AF_ij = 0
            end
        end
    end
    for j = 1: col from Matrix Alternatives_{MA*NV}
        for i = 1: row from Matrix Alternatives_{MA*NV}
            compute probability of choosing alternatives as P_ij using (6. 3)
        end
    end
    for j = 1: col from Matrix Alternatives_{MA*NV}
        for i = 1: row from Matrix Alternatives_{MA*NV}
            if i = Best Location_j
                P_ij = PP
            else
                P_ij = (1 − PP)P_ij
```

38: | | | | ***end***

39: | | | ***end***

40: | | ***end***

41: | ***end***

OUTPUT:

42: Optimal Solutions for fitness function

6.3 Literature Review

Dolphins make use of an echolocation system in order to detect and distinguish between objects under water. Bats have an integrated echolocation system that uses sound waves that usually vary and diversify depending on both the prey's location and the environment. Scientific studies have indicated that a dolphin's echo signals often fail to sense prey size or structure recognition, whereas the bat echo signals do so effectively. The signal duration, repetition and bandwidth varies as the bat gets nearer to the prey location. This particular bat feature cannot be found in the dolphin echolocation system. The broadband and short duration of dolphin echo signals improves the dolphin's sensing capabilities in accordance with the environment and prey's location. This function facilitates the dolphin in narrowing down its search from a global search to a local search in an effective way in the prey's direction [227] – [230]. On the other hand, bats have an impressive searching capability in the starting stage of the algorithm but later on, the algorithm severely suffers from poor searching in its local searching space, meaning that it can easily be trapped by the local optima [169], [204]. This means that both algorithms are embedded with an echolocation system that has some limitations in terms of its searching capabilities. Several attempts have been made to improve the BA and DEA algorithms based on their echolocation feature. For instance, Kim *et al*. [231] presented BIH (Bat Intelligent Hunting) algorithm for solving various complex optimization problems. In BIH, bat locates and captures its prey utilizing its echolocation feature. The proposed algorithm was applied for multiprocessor scheduling problem. Bora *et al*. [232] proposed a novel version for bat that is used to optimize

the mono and multi-objective based optimization problems related to direct current wheel motor. Further modification in the echolocation feature of bat was proposed by Biswal *et al*. [233] to provide the optimal solutions in the economic load dispatch system. Niknam *et al*. [234] presented a multi-objective based modified bat algorithm for constrained based dynamic economic dispatch problem. A self-tuning integral controller in bat echolocation feature based algorithm was presented by kumaravel and Kumar [235] for static synchronous compensator problem. On the other hand, many research applied DEA for various applications optimization problems. For instance, Zulkifli *et al*. [236] optimize the size of photovoltaic system by utilizing the version of DEA algorithm. DEA with tuning of its parameters is applied for multi-objective based reactive power dispatch optimization problem by Kanagasabai *et al*. [237]. The hybrid model of DEA and ACO was presented by Mahdi *et al*. [238] for optimal discrete sizing of truss structures. Similarly, Mahesh *et al*. [239] presented the hybrid algorithm, called as DECSA (Dolphin Echolocation and Crow Search Algorithm) for the cluster based energy aware routing optimization problem in wireless sensor networks.

Despite the fact that the echolocation systems applied by bats and dolphins possess a number pros and cons in accordance with the environment and the type of prey focused on, both algorithms (BA and DEA) predominantly depend on the echolocation feature for prey (optimal solution) searching. In straightforward terms, both algorithms (Bat and DEA) use features of the echolocation systems in different formats (sound waves and sonar waves) for prey searching. While they have some benefits to their echo systems, the algorithms also have some flaws in terms of their parameters which limit their searching capability. This echolocation feature encouraged our research team to generate a hybrid model of BA and DEA, which includes the positive features of the *"standard"* BA and DEA algorithms that will smartly handle exploration and exploitation capabilities and thus produce optimal solutions for various optimization problems.

6.4 Optimization Model of the MPLS Network

In this section of the chapter, the optimization model for MPLS network will be presented, in which two conflicted objective functions i, e. minimizing energy consumption and constrained based routing functions will be considered. These objective functions are integrated

with multiple constraints and thus, reflected as MCOP optimization problem in MPLS networks. For the given MCOP MPLS optimization model, the chapter offers a solution in the form of proposed metaheuristic optimization algorithm, which is described in the following sections. Various experiments are conducted for optimization model by proposed algorithm in order to validate algorithm`s performance in different network scenarios. For the considered MCOP MPLS model, the graph theory is employed in which MPLS network is represented as graph G and is dependent on the number of nodes as vertices R and links as edges E, where $G = (R, E)$. The number of r_i^{th} routers are included in the routers set as R_{set} or $R = \{r_i, i = 1,2,3, \dots, n\}$ and for total number of e_i links in the network are the member of link set as E_{set} or $E = \{e_{i_{(i,j)}},\ i = 1,2,3, \dots, n\}$. Each $e_{i_{(i,j)}}$ link is connected between i node to j node in the network. In the considered optimization network model, the traffic demand is received at ingress LSR, which then compute the optimal paths based on the objective functions.

6.4.1 Minimizing Energy Consumption Objective Function

In order to optimize energy consumption in the MPLS network, this objective function is managed into two levels; where the first level introduces the minimization of energy consumption at routers and links level, while the second level minimizes the total link capacity utilization. In simple words, the objective function will optimize the total energy in the network by minimizing both at routers and links level as well as by managing network traffic according to links capacity. Let us consider the capacity of the $e_{i_{(i,j)}}$ link $\in E_{set}$ as $C_{(i,j)}$, where $C_{(i,j)} \geq 0$ and is identified by line cards of i router to j router in the network. The ingress LSR receive k^{th} traffic demand, where $k \in K_{set}$. In the network, the demand k traffic must transfer a volume of ϕ^k from source router $s(k)$ to destination router $d(k)$. The energy consumption of the chassis in r_i^{th} router is denoted as P_i while $g_{(i,j)}$ represents the energy efficiency of the line cards connecting the links in the network. The binary variables z_i and y_e are introduced here for ON/OFF status of r_i router (s) and $e_{i_{(i,j)}}$ links in the network, respectively. In addition to this, for traffic demand k flow over $e_{i_{(i,j)}}$ link is represented by a continuous variable $x_{(i,j)}^k$. In the considered energy aware traffic engineering MPLS network optimization problem consists of

those r_i routers and $e_{i_{(i,j)}}$ links which have minimum energy consumption and then each traffic demand utilize multi-path flow minimizing the total link capacity utilization. The energy optimization based objective function can be formulated as;

$$Fitness\ Function_{Energy\ Consumption} = mini.\left(\sum_{r_i \in R_{set}} P_i\ z_i + \sum_{e_{i_{(i,j)}} \in E_{set}} \sum_{k \in K_{set}} g_{(i,j)} x^k_{(i,j)} \right) \quad (6.\ 4)$$

Subject to constraints;

$$z_i \in \{0, 1\}, \quad \forall\ r_i \in R_{set} \quad \text{Constraint (6.4. 1)}$$

$$y_e \geq z_i + z_j - 1, \quad \forall\ e_{i_{(i,j)}} \in E_{set} \quad \text{Constraint (6.4. 2)}$$

$$y_e \leq z_i, \quad \forall\ e_{i_{(i,j)}} \in E_{set} \quad \text{Constraint (6.4. 3)}$$

$$y_e \leq z_j, \quad \forall\ e_{i_{(i,j)}} \in E_{set} \quad \text{Constraint (6.4. 4)}$$

$$mini.\left(\sum_{e_{i_{(i,j)}} \in E_{set}} \sum_{k \in K_{set}} x^k_{(i,j)} \right) \quad (6.\ 5)$$

Subject to constraints;

$$\sum_{e_{i_{(i,j)}} \in E_{set} + i} x^k_{(i,j)} - \sum_{e_{i_{(i,j)}} \in E_{set} + i} x^k_{(i,j)} = b^k_i \quad , \forall\ r_i \in R_{set}, \forall k \in K_{set} \quad \text{Constraint (6.5. 1)}$$

$$\sum_{k \in K_{set}} x^k_{(i,j)} \leq C_{(i,j)}\ y_e \quad , \quad \forall\ e_{i_{(i,j)}} \in E_{set} \quad \text{Constraint (6.5. 2)}$$

$$x^{k}_{(i,j)} \geq 0, \quad \forall\ e_{i_{(i,j)}} \in E_{set}\ , \quad \forall k \in K_{set}$$ Constraint (6.5. 3)

where

$$b_i^k = \begin{cases} \phi^k, & if\ i = s(k) \\ -\phi^k, & if\ i = d(k) \\ 0, & otherwise \end{cases}$$

Eq. (6. 4) demonstrates the minimum energy consumption objective function at routers and links, whereas the Eq. (6. 5) is used to minimize the total link capacity utilization. Constraint (6.4. 1) identifies that the variable z_i is either 0 or 1 which shows that the router is ON or OFF. From Constraint (6.4. 1) to Constraint (6.4. 4) ensures that all the $e_{i_{(i,j)}}$ links can only be activated between the switched ON routers $i, e.\ s(k)$ router to $d(k)$ of the network. Constraint (6.5. 1) ensures the data flow from one router to another router $i, e.\ s(k) - d(k)$. Constraint (6.5. 2) represents the capacity constraints in which the capacity of each link must be equal to or greater than the data flow over it. While Constraint (6.5. 3) requires the $x^k_{(i,j)}$ variable as non-negative to confirm the traffic flow over the selected link.

6.4.2 Optimizing Constrained based Routing Objective Function

For optimizing the constraint based routing objective function, there are some additional variables that need to be defined for the given MPLS topology. In the MPLS network, the originating or source LSR is denoted as u_l while the termination or destination LSR is presented as v_l. Available bandwidth of the link is μ_l, administration cost of the link is a_l and the maximum allocation multiplier factor of the link is denoted as K_l. In the considered MLPS topology, the ingress LSR is s_i, whereas the egress LSR is d_i. In between ingress LSR and egress LSR, one or more LSPs originate as U and each LSP is the member of LSP set as F. The bandwidth of the LSP is λ_i. The maximum number of hops through LSP is represented by h_i. To confirm that the computed LSP based on the links that belongs to the list set of MPLS network, a variable x_{il} is initiated with the following condition as;

$$x_{il} = \begin{cases} 1, & if\ LSP\ i\ \in F\ is\ routed\ over\ e_{i_{(i,j)}} \in\ E_{set} \\ 0, & otherwise \end{cases}$$

The constrained based routing optimization leads to an objective function that minimizes the sum of administration costs of all the selected links and the total traffic flow over each link. This can be formulated as;

$$Objective\ Function_{Constrained\ Routing}\ = mini.\left(\sum_{e_{i_{(i,j)}} \in E_{set}} \sum_{i \in F} \lambda_i\ x_{il} \right) \quad (6.\ 6)$$

Subject to constraints;

$$\sum_{i \in F} \lambda_i\ x_{il}\ \leq\ \mu_l\, K_l, \quad \forall\ e_{i_{(i,j)}} \in E_{set} \quad \text{Constraint (6.6. 1)}$$

$$\sum_{e_{i_{(i,j)}} \in E_{set}} x_{il}\ \leq\ h_i, \quad \forall\ i\ \in F \quad \text{Constraint (6.6. 2)}$$

$$\sum_{\forall\ e_{i_{(i,j)}} | u_l = n} x_{il}\ =\ 1\,, \quad \forall\ n\ \in U\ \forall\ i | s_i = n \quad \text{Constraint (6.6. 3)}$$

$$\sum_{\forall\ e_{i_{(i,j)}} | v_l = n} x_{il}\ =\ 1\,, \quad \forall\ n\ \in U\ \forall\ i | d_i = n \quad \text{Constraint (6.6. 4)}$$

$$\sum_{\forall\ e_{i_{(i,j)}} | u_l = n} x_{il}\ -\ \sum_{\forall\ e_{i_{(i,j)}} | v_l = n} x_{il}\ = 0\ , \forall\ n\ \in\ R_{set}\, \forall\ i | s_i \neq n\, \forall\ i | d_i \neq n \quad \text{Constraint (6.6. 5)}$$

$$0\ \leq\ x_{il}\ \leq\ 1, \quad \forall\ i\ \in F \quad \text{Constraint (6.6. 6)}$$

Constraint (6.6. 1) approves that the link capacities are not exceeded from the given capacity range. Constraint (6.6. 2) limits the number of LSR hops in an LSP. Constraint (6.6. 3) and Constraint (6.6. 4) confirms that all LSPs in the network are originating and terminating accordingly. Constraint (6.6. 5) ensures that an LSP is routed through intermediate LSRs in the

network, which ensures an end-to-end path across the network. At the end, the Constraint (6.6. 6) describes that the decision variable x_{il} is either 0 or 1 for all LSPs across the network.

6.5 Proposing *Hybrid Bat Dolphin Echolocation (HBD)* Algorithm for the Optimization Problem

Both Bat and DEA algorithms have different strategies of searching the optimal solutions in searching space, but both rely on their echolocation capabilities. In this section of the chapter, a hybrid model of Bat and DEA algorithms is introduced and entitled as *"HBD* (*Hybrid Bat Dolphin Echolocation*)*"* algorithm. The proposed *HBD* algorithm will be described with the support of the pseudo codes of the Algorithms , flow chart, figures, and discussed steps as follow;

Step 6.6. 1: Initialize random positions in searching matrix using Bat algorithm parameters:

This step contains searching matrix initialization, locating random positions, and then computes their position value using fitness function. The following stages will be followed at this initial stage;

i. Define the UB (Upper Bounds) and LB (Lower Bounds) of searching space as searching matrix.

ii. Initialize random positions as $x_i^t, i = 1,2, \dots n$ in searching space with dimensions D_i in iteration t, where $D_i = \{D_1, D_2, \dots D_n\}$.

iii. Initialize random velocities as $v_i^t, i = 1,2, \dots n$.

iv. Define maximum and minimum frequencies ranges as f_{max} and f_{min}

v. Outline the maximum and minimum ranges of loudness parameter as $L_{d_{max}}$ and $L_{d_{min}}$.

vi. Initialize random value of loudness parameter as L_d^t.

vii. Initialize pulse rate as P_r^t.

viii. Compute the fitness function for the initial positions as f_x^t.

ix. Find the best searching agent as $x_{Global-best}^t$.

Step 6.6. 2: Processing Loop of the algorithm;

During processing loop, the algorithm will start converging towards optimal solution in searching space with respect to the number of iterations. The following actions will be taken as;

i. Define the number of iterations / generations in which the algorithm will approach towards optimal solutions in searching space.

ii. Update frequency as f_i^{t+1} using Eq. (6. 7), having $\beta = [0\ ,1]$;

$$f_i^{t+1} = f_{max} + (f_{max} - f_{min})\,\beta \tag{6. 7}$$

iii. Update velocities of each position using Eq. (6. 8) as;

$$v_i^{t+1} = v_i^t + (x_i^t - x_{Global-best}^t)\, f_i^{t+1} \tag{6. 8}$$

iv. Based on updated velocities, update the positions x_i using Eq. (6. 9);

$$x_i^{t+1} = x_i^t + v_i^{t+1} \tag{6. 9}$$

v. To compute the local best position for searching agents, a condition is applied and if this condition is satisfied then the local best position can be updated based on Eq. (6. 10).

$$\boldsymbol{if} \quad rand > P_r^t \quad \boldsymbol{do}$$

$$x_{i,Local-best}^{t+1} = x_i^{t+1} + \varepsilon\, L_d \tag{6. 10}$$

vi. Now compute the fitness function for the updated positions x_i^{t+1}.

Step 6.6. 3: Comparing the fitness function values;

This step check the fitness function values for each searching agent and investigate that whether the fitness function value is improved with respect to searching agent updated position or not. Based on the fitness function values the proposed *HBD* will take some actions, which are described below as;

i. Compare the previous fitness function values f_x^t for each searching agent x_i^{t+1} with its updated fitness function value f_x^{t+1} as;

$$\boldsymbol{if} \quad f_x^t > f_x^{t+1} \quad \boldsymbol{do}$$

$$f_x^t = f_x^{t+1} \tag{6. 11}$$

$$x_i^t = x_i^{t+1} \tag{6. 12}$$

$$\boldsymbol{else\ if} \quad f_x^t < f_x^{t+1} \quad \boldsymbol{do}$$

$$find\ non \tag{6. 13}$$

$$- optimal\ positions\ x_i^{t+1}\ (along\ with\ D_i) against\ \ their\ \ f_x^{t+1}\ \ values$$

In the above if condition the minimum fitness values are considered as optimal values and the position against those values are also taken as improved position of searching agents. Therefore, if the condition is satisfied then the fitness value will be updated with new fitness value as Eq. (6. 11) and also the position of that particular agent will be updated, shown in Eq. (6. 12). Whereas, if the condition is not fulfilled, then according to Eq. (6. 13), that specific searching agent position along with its fitness function value will be figured-out from the searching agents' population.

Step 6.6. 4: Initializing the new positions for searching agents;

In this stage, the unimproved positions x_i^{t+1} (having sub-optimal fitness function values) of searching agents will be updated. For this purpose several steps will be taken as;

i. From the dimensions (D_i) of each searching agent x_i^{t+1}, take the dimension with maximum value as D_{max} and the dimension with minimum value D_{min} .

ii. Take D_{max} as upper bound $i, e.$, $D_{max} = UB$ and lower bound as $D_{min} = LB$.

iii. Initialize the searching space matrix within defined LB and UB for each searching agent.

iv. Each searching agent will initialize random position in the searching space matrix.

Step 6.6. 5: Sorting searching space order in the form of Alternative Matrix of DEA Technique;

During this process, the searching space for each searching agent position will be sorted in an ascending or descending order, this will be done by taking following steps;

i. The searching space for each variable is sorted into one matrix of searching space in an ascending or descending order. This matrix is known as $Alternative\ Matrix$.

ii. From the $Alternative\ Matrix$ take one location as searching agent position x_{ij}^{t+1}.

iii. Initialize the radius R_e of the effective searching agent position x_{ij}^{t+1}. The range of radius should be $1/4^{th}$ of the searching space, and $R_e => from - R_e \ \ to \ \ R_e$.

iv. Within this searching domain or radius ($R_e = \{- R_e \ , \ R_e\}$), find the positions x_{ij}^{t+1} in the j^{th} columns of $Alternative\ Matrix$.

Step 6.6. 6: Apply the fitness function for the updated positions;

At this stage, the updated positions will be checked with fitness function as;

i. After updated positions for searching agents from $Alternative\ Matrix$, the fitness function will be applied again.

ii. Such as, if the updated fitness function is better than the previous fitness function then takes the update position of searching agent as improved position, $else$ go back to Step 6.6. 4.

Step 6.6. 7: Achieving optimal solution;

During this final stage of the algorithm, the archive of optimal solutions is collected along with other important steps, which are explained as follow;

i. Store the optimal fitness function values and consider the update positions as improved positions compared to previous positions of the searching agents.

ii. As the algorithm is coherently converging towards optimal solutions in searching space, then the loudness parameter L_d value will decrease and pulse-rate P_r^t increases, using below mathematical expressions;

$$L_d^{t+1} = \alpha\, L_d^t \tag{6. 14}$$

$$P_r^{t+1} = P_r^t \left[1 - e^{-\gamma(t+1)}\right] \tag{6. 15}$$

Algorithm 6. 3: Pseudo code of Proposed Bat Dolphin Echolocation Algorithm

---

Result: Optimal Solutions from the searching space matrix

Step 6.6. 1: Initialize random positions in searching matrix using Bat algorithm parameters:

1. ***for*** i^{th} searching agents $\boldsymbol{B}$ in each dimension $(D_i, i = \{1,2,3, \ldots . n\})$ ***do***
2. initialize random positions $x_i^t = x_1, x_2, x_3, \ldots x_n$
3. initialize random velocities $v_i^t = v_1, v_2, v_3, \ldots v_n$
4. initialize frequencies (maximum and minimum frequency with ranges) $f_i^t = [f_{max}, f_{min}]$
5. initialize loudness levels (maximum and minimum loudness with ranges) $L_d = [L_{d_{max}}, L_{d_{min}}]$
6. initialize random value of L_d^t
7. initialize random value of P_r^t
8. compute fitness function values as f_x^t for initial random positions of x_i^t
9. compute $x_{Global-best}^t$

10. ***end***

Step 6.6. 2: Processing Loop of the algorithm**;**

11. ***while*** number of iterations $<$ maximum number of iterations ***do***
12. ***for*** f_x^{t+1}, $i = 1{:}N$ in each D_i ***do***
13. update frequency f_i^{t+1} using Eq. (6. 7) with random $\beta = [0\ ,1]$
14. update velocities v_i^{t+1} using Eq. (6. 8)
15. update positions x_i^{t+1} using Eq. (6. 9)
16. ***if*** $rand > P_r^t$ ***do***
17. compute $x_{i,Local-best}^{t+1}$ based on Eq. (6. 10)
18. ***end***
19. compute fitness function values as f_x^{t+1} for updated positions of x_i^{t+1}

Step 6.6. 3: Comparing the fitness function values**;**

20. ***if*** $f_x^t > f_x^{t+1}$ ***do***
21. update f_x^t with f_x^{t+1} as $f_x^t = f_x^{t+1}$ according to Eq. (6. 11)
22. update positions as $x_i^t = x_i^{t+1}$ as given in Eq. (6. 12)
23. ***else if*** $f_x^t < f_x^{t+1}$ ***do***
24. find the unimproved fitness function values from f_x^{t+1}
25. find non-optimal positions x_i^{t+1} against f_x^{t+1} values
26. ***end***

Step 6.6. 4: Initializing the new positions for searching agents**;**

27. **while** $f_x^t > f_x^{t+1}$ *against* x_i^{t+1} **do**

28. take D_{max} and D_{min} from each unimproved x_i^{t+1}

29. take D_{max} value as UB and D_{min} as LB for the searching space

30. initialize searching space with the defined boundaries of UB and LB.

31. Initialize random values within this searching space

Step 6.6. 5: Sorting searching space order in the form of Alternative Matrix of DEA Technique;

32. initialize the $Alternative\ Matrix$

33. from the $Alternative\ Matrix$ take one location as searching agent position x_{ij}^{t+1}

34. at x_{ij}^{t+1} take the effective radius R_e

35. within the radius of $R_e = \{-R_e\ ,\ R_e\}$ find the positions of x_{ij}^{t+1} in $Alternative\ Matrix$

Step 6.6. 6: Apply the fitness function for the updated positions**;**

36. update positions x_i^{t+1} from $Alternative\ Matrix$

37. compute the fitness function f_x^{t+1} values for the updated positions x_i^{t+1}

38. **end**

39. minimize loudness L_d^{t+1} using Eq.(5.5) (6. 14)

40. increase pulse rate P_r^{t+1} using Eq. (6. 15)

41. compute average loudness L_d^t

42. **end**

Step 6.6. 7: Achieving optimal solution**;**

43. update $x^t_{Global-best}$

44. store optimal solutions

45. ***end***

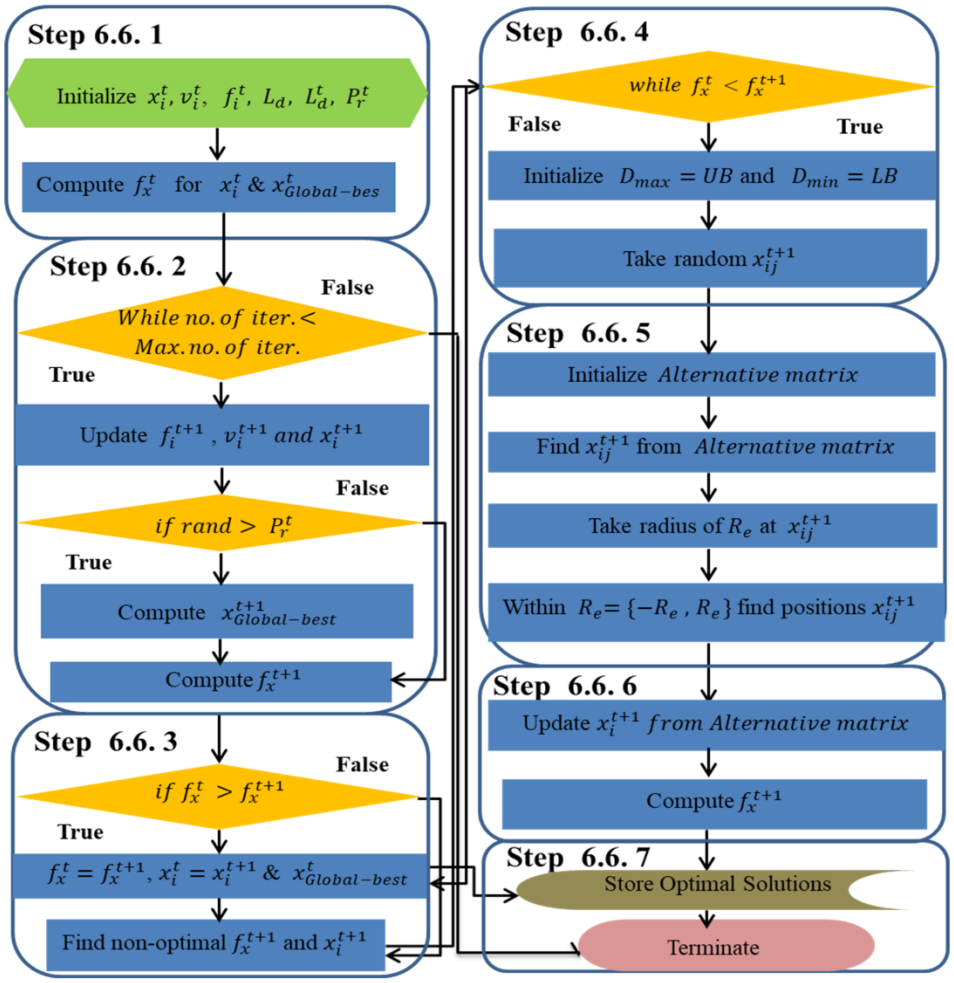


Fig. 6. 1: Proposed *HBD* Flow Chart

6.6 Experimental Assessment of the Proposed *HBD* Algorithm

In order to investigate the performance of the proposed *HBD* algorithm, the experiments were conducted in two categories. The first set of experiments was implemented to solve the MCOP optimization problem in MPLS networks (described in section **Error! Reference source not found.**) while the second set of experiments were performed to analyze the algorithm efficiency in relation to mathematical optimization functions. For all of the experiments, the *HBD* algorithm performance was not only measured in terms of generating optimal solutions for MPLS networks but also to investigate the *HBD* outcomes (as solutions) in comparison to its competitive algorithms. For this purpose, the familiar metaheuristic algorithms were considered for the experiments and they consist of the Bat, DEA, PSO, GWO, and WOA (Whale Optimization) algorithms. To determine the simulation-based results, the *MATLAB 2016a* package is used. During the experimental results analysis, the research focused on analyzing every algorithm's convergence rate in relation to the optimal solutions in the searching space while dealing with the MPLS optimization problem. This was also done for the mathematical optimization functions. On top of that, the statistical data was also collected to inspect the algorithm outcomes. The brief description of the implemented experimental results has been discussed in the later sub-sections.

6.6.1 Obtained Pareto Graphs for the *HBD* Algorithm

The MPLS optimization model (described in section **Error! Reference source not found.**) is based on two objective functions. Therefore it is an essential part of the proposed *HBD* algorithm in terms of offering solutions that do not dominate one another. Simply put, the *HBD* will produce non-dominant solutions for the multiple objective-based optimization model of MPLS network and it will produce results in the form of Pareto Front. For this experiment, six versions of the *HBD* algorithm were arranged as *HBD-n=> HBD-1*, *HBD-2*, *HBD-3*, *HBD-4*, *HBD-5*, and *HBD-6*, where$n = \{1,..6\}$. These versions of *HBD* vary in terms of the number of searching agents and the number of iterations, as illustrated in Table 6. 1. For the considered *HBD* versions, the experiments were conducted over a different number of nodes such as

$D = 10, 40, 80,$ and 120 nodes. The reason for considering this variation is to review the success of the *HBD* algorithm while generating Pareto Front graphs for various scales of network. For the considered experiment, the results were obtained in the form of Pareto Front graphs as shown in Fig. 6. 2 (a−d). To generate the Pareto Fronts present in the various versions of *HBD-n*, the non-dominant solutions were collected during the simulations and then the solutions were connected as points with lines. In 6. 2 the six versions of *HBD-n such as HBD-1, ... HBD-6* successfully generated non-dominant solutions in the form of Pareto fronts for the considered objective (constrained-based routing and energy consumption) functions in the MPLS optimization model for different network sizes. This means that all versions of *HBD-n* (*HBD-1, ...HBD-6*) can generate Pareto fronts on a small to large scale in MPLS networks. The Pareto fronts graphs in Fig. 6. 2 (a−d) also shows that the constrained-based routing and energy consumption objective functions are contradictory to each other. There is still some place for solutions as non-dominant points, which the *HBD* algorithm can successfully produce. In addition to this, the *HBD* algorithm also confirmed that this algorithm has the capability to manage multi-objective optimization problems while generating non-dominant solutions irrespective of the algorithm`s number of iterations and searching agents. The Pareto fronts in Fig. 6. 2 establish the fact that the *HBD* algorithm's performance predominantly does not depend on the number of searching agents and the number of iterations. It has the competence of producing non-dominant solutions for conflicting-based objective functions in any network.

Table 6. 1: Proposed HBD Algorithm Various Versions for Pareto Front Experiment

Version	**Number of Searching Agents**	**Number of Iterations**
$HBD-1$	20	100
$HBD-2$	30	
$HBD-3$	40	
$HBD-4$	50	200
$HBD-5$	60	400

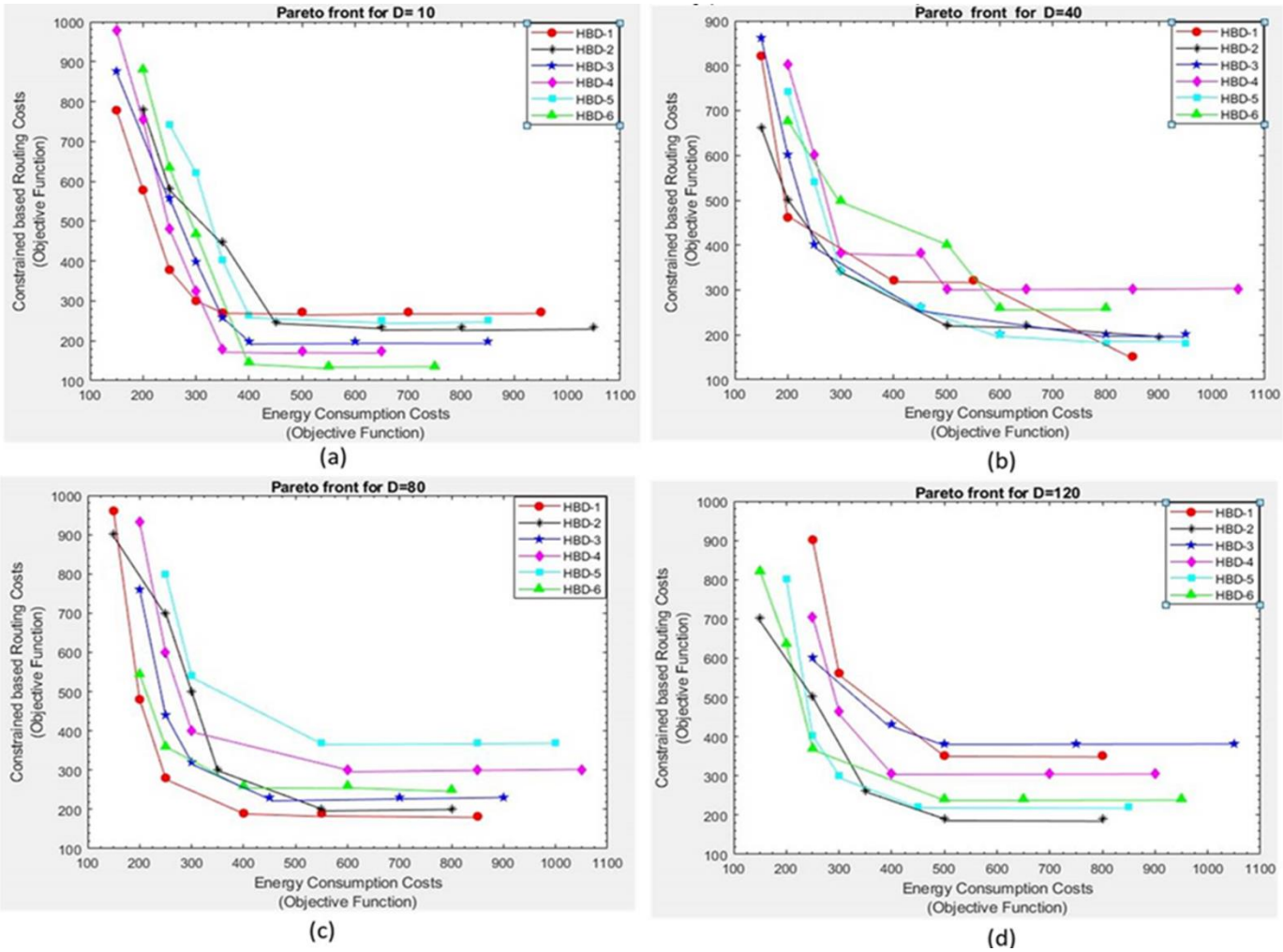


Fig. 6. 2: Pareto front graphs for D = 10, 40, 80, and 120 Nodes MPLS Network

6.6.2 Convergence Analysis of *HBD* Compared to Other Algorithms for MPLS Optimization

The convergence rate reflects the searching capability of algorithms for optimal solutions in the searching domain. It has always been considered an essential parameter used to measure the examining metaheuristic algorithms' performance. Metaheuristic algorithms converge

towards optimal solutions over a number of iterations. If there exists either an exploration or exploitation problem during the iterations, then it affects the convergence of the algorithm. To evaluate the proposed *HBD* algorithm for the considered MPLS optimization problem, the convergence of *HBD* was monitored and compared with its competing metaheuristic (Bat, DEA, PSO, GWO, and WOA) algorithms in terms of convergence over 200 iterations. All of the mentioned algorithms were implemented over a different number of node network$s\ i, e. D = 10, 40, 80$ and 120 nodes as shown in the figures (Fig. 6. 3 (a−d) and Fig. 6. 4 (a−d)) respectively. The experiment used to monitor the convergence of the various node networks was conducted for both objective (constrained-based routing and energy consumption cost) functions. Fig. 6. 3 (a−d) depicts the convergence activity of the HBD, Bat, DEA, GWO, and WOA algorithms for the constrained-based routing cost objective function whereas Fig. 6. 4 portrays the convergence activity of the aforementioned algorithms for the energy consumption cost objective function. Both Fig. 6. 3 (a−d) and Fig. 6. 4 (a−d) were used to make a comparative analysis between the proposed *HBD* algorithm and its rival algorithms (Bat, DEA, PSO, GWO, and WOA). The purpose of this comparison is to investigate the convergence of the *HBD* algorithm while comparing it to the already recognized metaheuristic algorithms for the given MPLS optimization problem.

While exploring the results of Fig. 6. 3 (a−d), it can clearly be observed that the *HBD* algorithm convergences efficiently compare to the other algorithms. As an illustration, Fig. 6. 3 (a) depicts the convergence of *HBD*, Bat, DEA, PSO, GWO, and WOA algorithms for a 10 node network. In this case, *HBD* converges without a problem and it generates optimal solutions during the iterations. In contrast, the other algorithms do not converge during a number of iterations and as a consequence, they generated sub-optimal solutions. For example, the Bat, DEA, PSO, GWO, and WOA algorithms did not converge between 60^{th} to 110^{th}, 30^{th} to 130^{th}, 70^{th} to 120^{th}, 130^{th} to 190^{th}, and 60^{th} to 130^{th}in terms of iterations. Similar findings can be seen in Fig. 6. 3 (b), (c) and (d). However, for Fig. 6. 3 (b, c, and d), the results for Bat, DEA, PSO, GWO, and WOA are worse when compared to Fig. 6. 3 (a). This means that these algorithms have a more damaged convergence when implemented over large scale networks. On the other hand, the *HBD* algorithm has coherent convergence towards optimal solutions from a small to large scale in MPLS networks; hence it produced optimal solutions for the constrained-based routing objective function during the iterations. Fig. 6. 4 (a−d) shows the convergence

activity results for the energy consumption objective functions in a number of node networks. In Fig. 6. 4 (a−d), a similar trend can be observed for the *HBD* algorithm and its competitive algorithms in which *HBD* outclasses its competing algorithms for convergence activity during the iterations compared to the Bat, DEA, PSO, GWO, and WOA algorithms. For example, in Fig. 6. 4 (b), it can be seen that the *HBD* algorithm generates optimal solutions during the iterations, whereas the other aforementioned algorithms had poor convergence in a number of iterations. In Fig. 6. 4 (b), GWO is stuck from 70^{th} iteration until the end. The similar tendency of poor convergence can be witnessed for the Bat, DEA, PSO, and WOA algorithms during various iterations. Similarly in Fig. 6. 4 (a, c, and d), the proposed *HBD* algorithm has proven its superiority in terms of convergence rate compared to the Bat, DEA, PSO, GWO, and WOA algorithms. Fig. 6. 4 (a−d) reveals the better convergence of the proposed *HBD* algorithm compared to its competing algorithms when it comes to optimizing the energy consumption cost objective function. From the findings discussed in 3 (a−d) and Fig. 6. 4 (a−d), it can be concluded that the *HBD* algorithm can converge consistently for optimal solutions in the searching space of both constrained-based objective functions. The other algorithms have poor convergence in terms of the considered optimization problems and therefore they fail to produce optimal solutions.

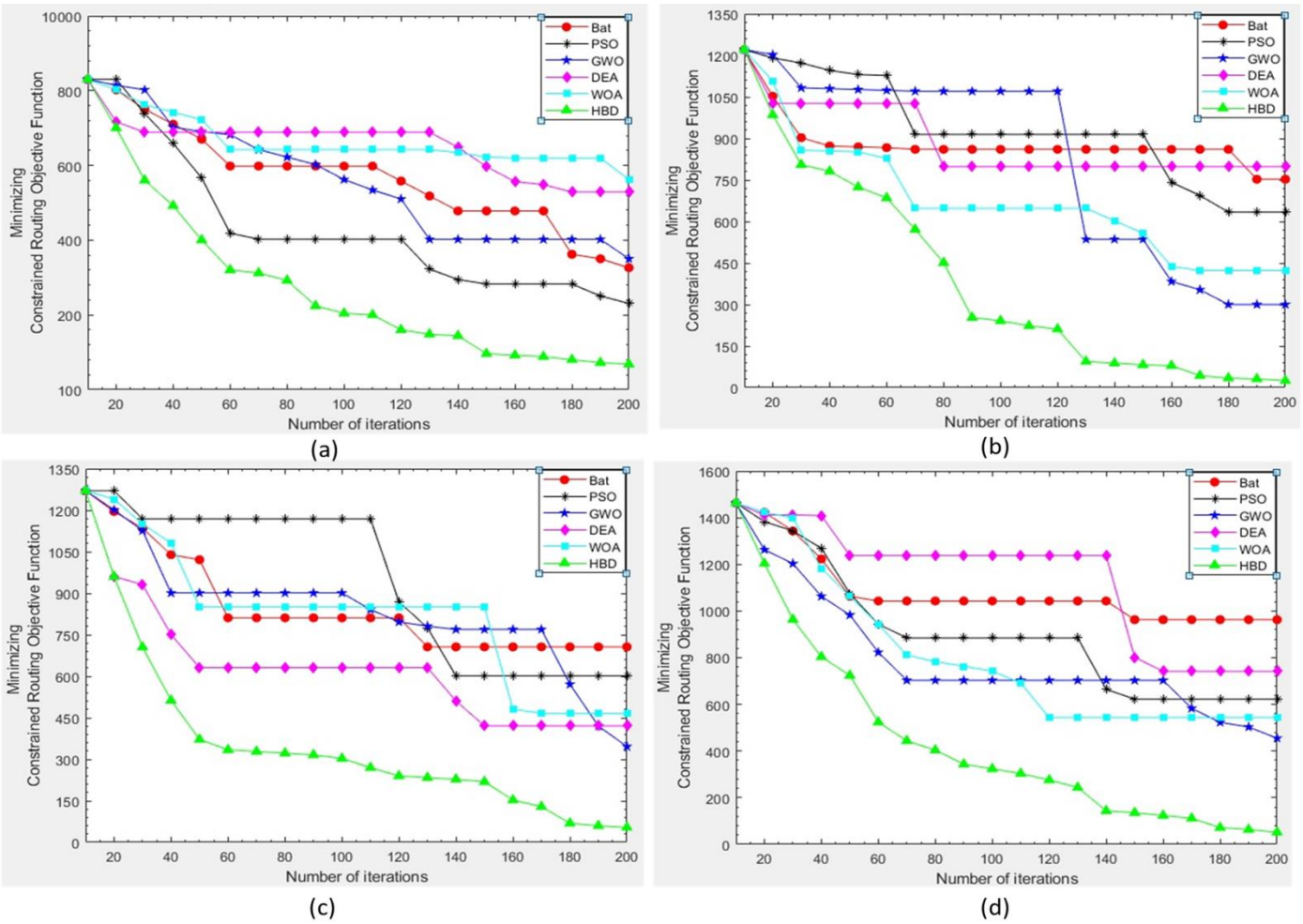


Fig. 6. 2: Convergence of HBD, Bat, PSO, DEA, WOA and GWO Algorithms for Routing Costs Function

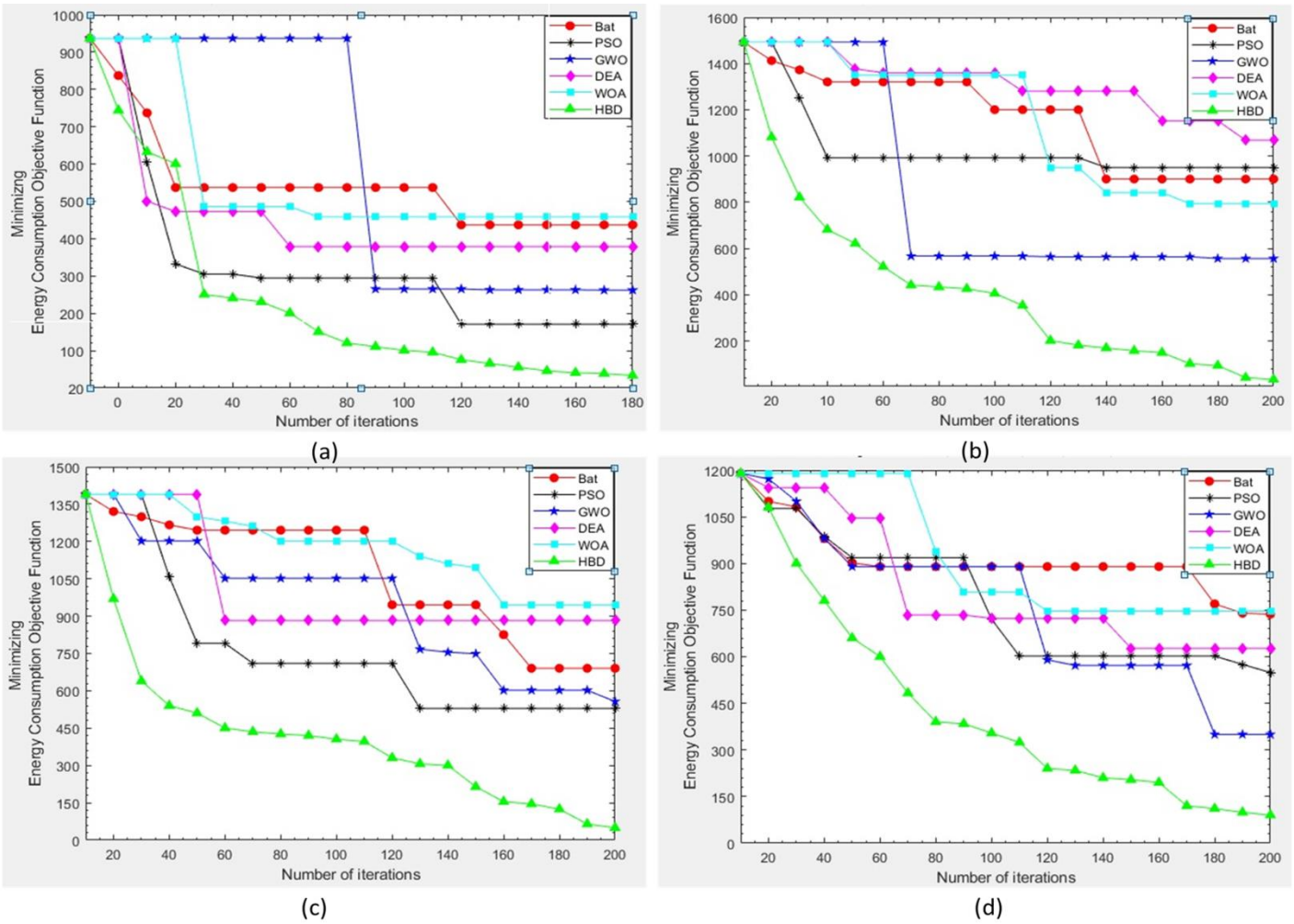


Fig. 6. 3: Convergence of HBD, Bat, PSO, DEA, WOA and GWO for Energy Consumption Function

6.6.3 Statistical Analysis of the Results Obtained for Network Optimization

For advanced research into the *HBD* algorithm`s performance, the data was collected from the experiments and measured using statistical parameters $i,e.$ *Mean*, *Standard Deviation* and *Optimal Solution*. For this experiment, each algorithm (*HBD*, Bat, DEA, PSO, GWO, and WOA) was executed for 100 *Runs*, where each *Run* consists of 100 iterations. We then arranged the data in the form of tables as illustrated in Table 6. 2 and Table 6. 3. Table 6. 2 describes the statistical data results obtained by the constrained-based routing costs objective functions while Table 6. 3 provides the details of the results collected for the energy

consumption cost-objective function. The experiments were conducted over 10, 40, 80, and 120 nodes MPLS networks. For both tables (Table 6. 2 and Table 6. 3), the statistical parameters were employed for an in-depth study of the *HBD* algorithm while comparing its obtained results with the competing algorithms (Bat, DEA, PSO, GWO, and WOA). The *Best* value is considered to be an optimal solution and it is obtained from 100 $Runs$. The Mean values were also calculated for 100 $Runs$. The *Best* and Mean values were used for minimal values as part of the optimal solution. Based on the results given in Table 6. 2 and Table 6. 3, the algorithms are ranked according to ranking formula. This helps to summarize the results for each algorithm in the tables. However, the ranking is only applied to the *Mean* values in the tables.

During the assessment of Table 6. 2 and Table 6. 3, the ranking results revealed that the *HBD* algorithm outperforms its competing algorithms by obtaining minimum (optimal) solutions in the form of *Mean* values. This implies that according to the *Mean* values obtained in Table 6. 2 and Table 6. 3 (for both the constrained-based routing and energy consumption cost-objective functions), the *HBD* algorithm produced optimal solutions in the form of *Mean* values from the experiments, meaning that it was had a $Final\ Rank\ = 1$. The second closest rank was obtained by the GWO algorithm for both objective functions in Table 6. 2 and Table 6. 3. In a similar manner, the *Best* (minimum / optimal)values achieved by all algorithms showed the supremacy of *HBD* algorithm in terms of providing significantly better solutions compare to its competing algorithms. The best solutions offered by the HBD algorithm can easily be spotted in Table 6. 2 and Table 6. 3. This clearly proves that the proposed HBD algorithm offers optimal solutions with a substantial difference compared to the Bat, DEA, PSO, GWO, and WOA algorithms. The third parameter of *Standard Deviation* represents the deviated values from the mean. When analyzing the *Standard Deviation* values, it was observed that the *HBD* algorithm has minimal values for this parameter, which in turn shows that for every Run, the random values in the form of solutions do not deviate a lot from the mean values. Metaheuristic algorithms are stochastic in nature and offer random solutions. The minimum S*tandard Deviation* values show that the algorithm offers random solutions with the smallest deviance. In Table 6. 2 and Table 6. 3, it can be seen that the *HBD* algorithm provided the minimum *Standard Deviation* values for the objective functions compared to the Bat, DEA, PSO, GWO, and WOA algorithms.

Table 6. 2: Statistical Data for Optimizing Constrained based Routing Costs Objective Function

	$n = 10$	$n = 40$	$n = 80$	$n = 120$	*Total Rank*	*Final Rank*
HBD						
Best	3.57E+01	3.90E+01	4.17E+01	5.52E+01		
Mean	3.89E+01	4.44E+01	5.42E+01	7.71E+01		
Std	6.23E+01	8.82E+01	1.23E+02	1.48E+02		
Rank	1	1	1	1	4	1
DEO						
Best	1.62E+02	1.72E+02	1.89E+02	2.01E+02		
Mean	1.85E+02	1.89E+02	2.01E+02	2.22E+02		
Std	1.06E+02	1.23E+02	1.32E+02	1.40E+02		
Rank	6	5	5	5	21	5
Bat						
Best	1.54E+02	1.77E+02	1.92E+02	2.03E+02		
Mean	1.66E+02	1.98E+02	2.10E+02	2.40E+02		
Std	1.11E+02	1.53E+02	1.61E+02	1.68E+02		
Rank	5	6	6	6	23	6
GWO						
Best	6.10E+01	8.18E+01	8.89E+01	9.42E+01		
Mean	6.81E+01	9.81E+01	1.01E+02	1.19E+02		
Std	1.56E+02	1.88E+02	1.95E+02	2.11E+02		
Rank	2	2	2	2	8	2
WOA						
Best	9.01E+01	9.76E+01	1.10E+02	1.63E+02		

Mean	1.18E+02	1.20E+02	1.33E+02	1.89E+02		
Std	1.63E+02	1.97E+02	2.00E+02	2.57E+02		
Rank	3	3	3	3	12	3
PSO						
Best	1.12E+02	1.36E+02	1.58E+02	1.89E+02		
Mean	1.46E+02	1.54E+02	1.79E+02	2.02E+02		
Std	2.09E+02	2.56E+02	2.99E+02	3.21E+02		
Rank	4	4	4	4	16	4

Table 6. 3: Statistical Data for Minimizing Energy Consumption Costs Objective Function

	$n = 10$	$n = 40$	$n = 80$	$n = 120$	***Total Rank***	***Final Rank***
HBD						
Best	4.16E+01	4.53E+01	6.02E+01	6.91E+01		
Mean	5.01E+01	7.22E+01	8.01E+01	8.32E+01		
Std	1.36E+02	1.45E+02	1.69E+02	2.12E+02		
Rank	1	1	1	1	4	1
DEO						
Best	2.52E+02	2.61E+02	2.63E+02	2.90E+02		
Mean	2.65E+02	2.68E+02	2.81E+02	3.00E+02		
Std	2.66E+02	2.77E+02	3.21E+02	3.49E+02		
Rank	6	6	5	4	21	6
Bat						
Best	1.89E+02	2.16E+02	2.69E+02	3.30E+02		
Mean	1.94E+02	2.22E+02	2.82E+02	3.45E+02		

Std	4.00E+02	4.20E+02	4.87E+02	5.12E+02		
Rank	4	4	6	6	20	5
GWO						
Best	1.00E+02	1.11E+02	1.42E+02	1.59E+02		
Mean	1.09E+02	1.20E+02	1.52E+02	1.68E+02		
Std	3.77E+02	4.06E+02	4.45E+02	4.99E+02		
Rank	2	2	2	2	8	2
WOA						
Best	1.68E+02	1.90E+02	2.04E+02	2.30E+02		
Mean	1.82E+02	2.01E+02	2.33E+02	2.39E+02		
Std	2.99E+02	3.36E+02	3.41E+02	3.88E+02		
Rank	3	3	3	3	12	3
PSO						
Best	1.90E+02	2.02E+02	2.25E+02	3.02E+02		
Mean	2.00E+02	2.31E+02	2.44E+02	3.11E+02		
Std	1.69E+02	2.06E+02	2.31E+02	3.50E+02		
Rank	5	5	4	5	19	4

6.6.4 Investigation of the *HBD* Algorithm in Relation to Mathematical Functions

Mathematical optimization functions, also known as test or benchmark functions are used to evaluate the optimization algorithms. Researchers working in the field of metaheuristics assess the optimization algorithm characteristics such as the convergence rate, precision and efficiency using benchmark optimization functions. Benchmark functions are useful for testing new algorithms in an unbiased way. Benchmark functions are classified into two groups, namely unimodal and multimodal functions. To examine the *HBD* algorithm`s performance, the experiments were conducted over a number of unimodal and multimodal functions. This experiment was grouped into two parts. The first part of the experiment was implemented over benchmark (unimodal and multimodal) functions to analyze the convergence rate of the *HBD* algorithm and to compare its performance with the Bat, DEA, PSO, GWO, and WOA algorithms. In the later experiments, statistical data was collected and measured using the parameters of *Mean*, *Standard Deviation* and *Optimal Solutions* to examine the *HBD* algorithm`s performance with its competing algorithms. The lists of unimodal and multimodal functions used in the experiments have been given in Table 6. 4.

Table 6. 4: List of Unimodal and Multimodal Optimization Functions

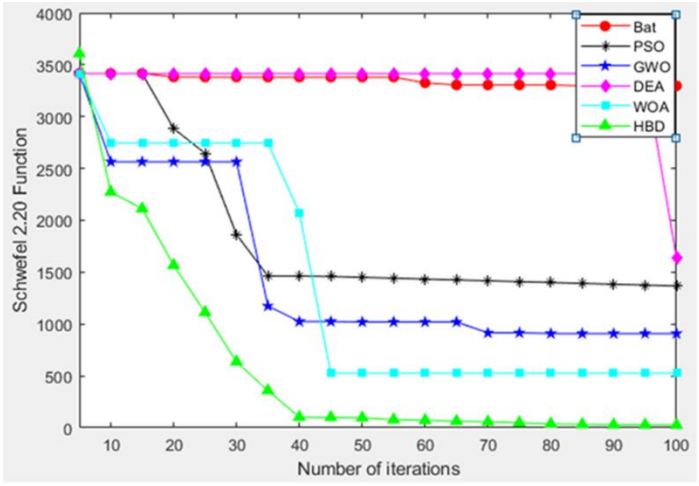
Functions	**Type**	**Math. Expression**	**Range**	**Optimum Value**
Schwefel 2.20	Unimodal	$\sum_{i=1}^{N} \lvert x_i \rvert$	$[-100:100]$	$f_x = 0$
Sphere	Unimodal	$\sum_{i=1}^{N} x_i^2$	$[-5.12 : 5.12]$	$f_x = 0$
Sum Square	Unimodal	$\sum_{i=1}^{N} i x_i^2$	$[-10 : 10]$	$f_x = 0$

Zakharov	Unimodal	$\sum_{i=1}^{N} x_i^2 + (\sum_{i=1}^{N} 0.5ix_i)^2 + (\sum_{i=1}^{N} 0.5ix_i)^4$	$[-5 : 10]$	$f_x = 0$
Elliptic	Unimodal	$\sum_{i=1}^{N} (10^6)^{(i-1)/(N-1)} x_i^2$	$[-100: 100]$	$f_x = 0$
Power	Unimodal	$\sum_{i=1}^{N} \lvert x_i \rvert^{(i+1)}$	$[-1: 1]$	$f_x = 0$
Quartic	Unimodal	$\sum_{i=1}^{N} ix_i^4 + rand(0,1)$	$[-1.28 : 1.28]$	$f_x = 0$
Schwefel 2.22	Unimodal	$\sum_{i=1}^{N} \lvert x_i \rvert + \prod_{i=1}^{N} \lvert x_i \rvert$	$[-100: 100]$	$f_x = 0$
Ackley	Multimodal	$-20exp\left(-0.2\sqrt{\sum_{i=1}^{N} x_i^2}\right) - exp\left(\frac{1}{N}\sum_{i=1}^{N} \cos(2\pi x_i)\right) + 20 + \exp(1)$	$[-32.768: 32.768]$	$f_x = 0$
Griewank	Multimodal	$10N + \sum_{i=1}^{N} \frac{x_i^2}{4000} - \prod_{i=1}^{N} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600: 600]$	$f_x = 0$
Holzman 2	Multimodal	$\sum_{i=1}^{N-1} ix_i^4$	$[-10: 10]$	$f_x = 0$
Hyper Ellipsoid	Multimodal	$\sum_{i=1}^{N} 2^{i-1} x_i^2$	$[-500: 500]$	$f_x = 0$

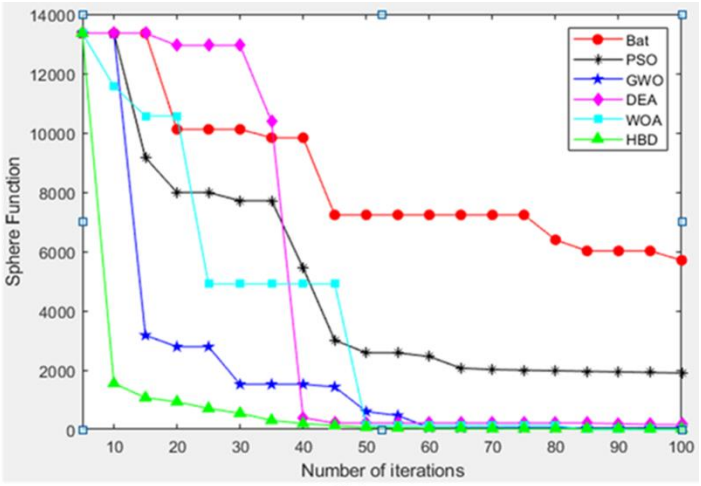
Csendes	Multimodal	$\sum_{i=1}^{N} x_i^6 \left(2 + sin\frac{1}{x_i}\right)$	$[-1:1]$	$f_x = 0$
Rastrigin	Multimodal	$10N + \sum_{i=1}^{N} [x_i^2 - 10\cos(2\pi x_i)]$	$[-5.12 : 5.12]$	$f_x = 0$
Cosine Mixture	Multimodal	$-0.1\sum_{i=1}^{N} \cos(5\pi x_i) - \sum_{i=1}^{N} x_i^2$	$[-1:1]$	$f_x = 0$
Schwefel 2.26	Multimodal	$-\frac{1}{N}\sum_{i=1}^{N} x_i \sin\sqrt{\lvert x_i \rvert}$	$[-500:500]$	$f_x = -418.98$

As mentioned before, the experiments were conducted in order to monitor the convergence of the *HBD* algorithm along with its competing algorithms before making a comparative analysis based on the results obtained in the form of figures (Fig. 6. 5 and Fig. 6. 6). Fig. 6. 5 (a − h) outlines the convergence graph results for the *HBD*, Bat, DEA, PSO, GWO, and WOA algorithms for the unimodal functions (from the list in Table 6. 4). Fig. 6. 6 (a − h) depicts the convergence activity results obtained from the aforementioned algorithms for multimodal functions (Table 6. 4). While reviewing the results in Fig. 6. 5 and Fig. 6. 6, it can be noted that the *HBD* algorithm continues its performance in an impressive way in terms of convergence during the iterations, as shown in Fig. 6. 5 (a − h). It can be observed that for all of the unimodal functions, the *HBD* algorithm successfully obtained an optimal solution, which is $f_x = 0$ (as shown in Table 6. 4). It is also witnessed that the *HBD* algorithm converges faster during the early iterations and it quickly achieves the optimal solutions (values) for the unimodal functions, unlike the Schwefel 2.20 and Elliptic functions, where the convergence is slightly slower compared to the other functions. The other metaheuristic algorithms $i, e.$ Bat, DEA, PSO, GWO, and WOA, have poor convergence during the iterations. Therefore in most cases, these algorithms failed to achieve optimal values. The Bat and DEA algorithms have the worse convergence for all unimodal functions and in many cases, these algorithms extremely failed to converge even during the earlier iterations. PSO, GWO, and WOA showed some convergence during the iterations but still failed to achieve optimal values (solutions), except for the Elliptic, Sum Square, and Sphere functions. Furthermore it can also be noticed that the Bat, DEA, PSO,

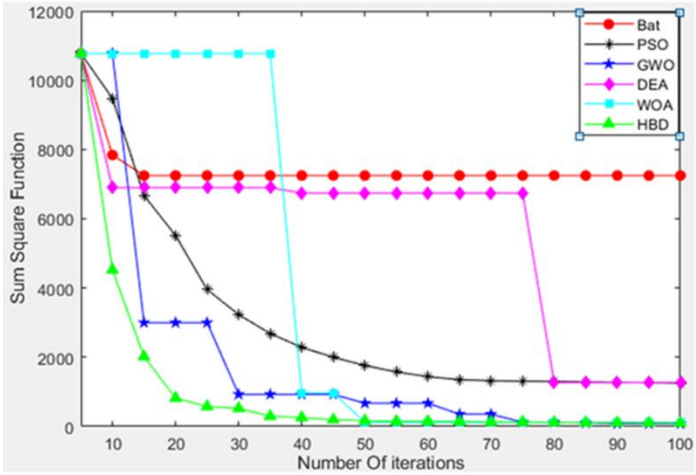
GWO, and WOA algorithms did not converge during certain iterations. As an example of the Schwefel 2.22 function, PSO did not converge between the 20^{th} - 50^{th} iterations. In the same way, Bat, DEA, GWO, and WOA did not converge between the 30^{th} - 100^{th}, 10^{th} $a$to 30^{th}, 10^{th} $-$ to 20^{th}, and the 55^{th} - 100^{th} iterations accordingly. This pattern of poor convergence can be observed for all of the other given unimodal functions as shown in Fig. 6. 4 (a $-$ h). Regarding the graphs of the multimodal functions given in Fig. 6. 5 (a $-$ h), the results of the *HBD* algorithm look identical to the unimodal functions with respect to its performance concerning convergence activity. In Fig. 6. 5 (a $-$ h), the *HBD* algorithm has the best convergence activity compared to the Bat, DEA, PSO, GWO, and WOA algorithms, thus offering an optimal solutions for the considered multimodal functions (from Table 6. 4). In the case of the Schwefel 2.26 and Ackley functions, *HBD* converges towards optimal solutions continuously during the iterations but it couldn't achieve an optimal solution. On the other hand, other competing algorithms drastically failed to converge during the iterations for the Schwefel 2.26 and Ackley functions. Therefore it can be stated that although *HBD* couldn't achieved an optimal solution for the Schwefel 2.26 and Ackley functions, it still maintained its consistent convergence during the iterations compared to its rival algorithms. For the rest of the multimodal functions, the *HBD* algorithm outperforms the rest of the competitive algorithms (Bat, DEA, PSO, GWO, and WOA) in terms of convergence activity as well as obtaining an optimal solution. From both figures (Fig. 6. 5 and Fig. 6. 6), it can be seen that the *HBD* algorithm offers optimal solutions as well as cohesive convergence during the iterations.


Schwefel 2.20 Function
Number of iterations
Bat
PSO
GWO
DEA
WOA
HBD

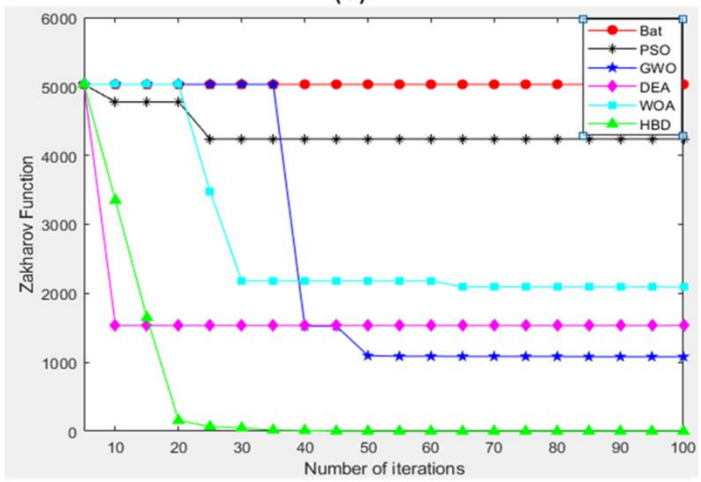
(a)


Sphere Function
Number of iterations
Bat
PSO
GWO
DEA
WOA
HBD

(b)


Sum Square Function
Number Of iterations
Bat
PSO
GWO
DEA
WOA
HBD

(c)


Zakharov Function
Number of iterations
Bat
PSO
GWO
DEA
WOA
HBD

(d)

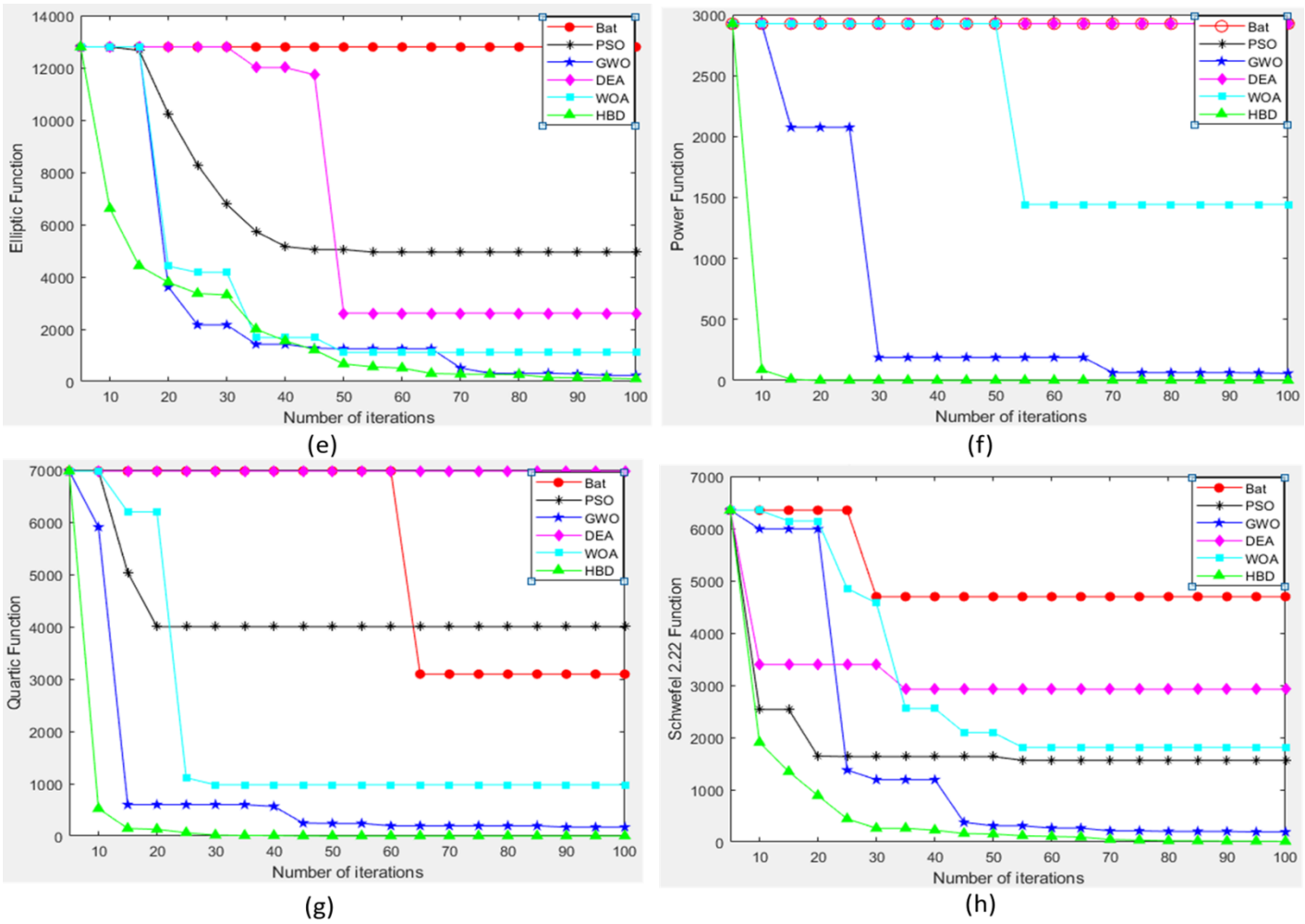
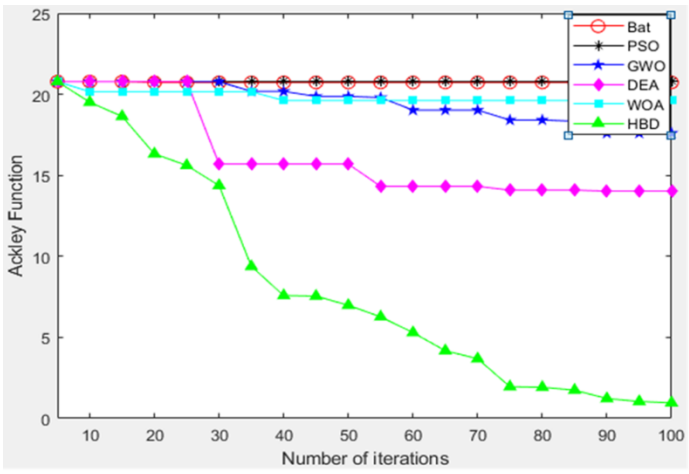
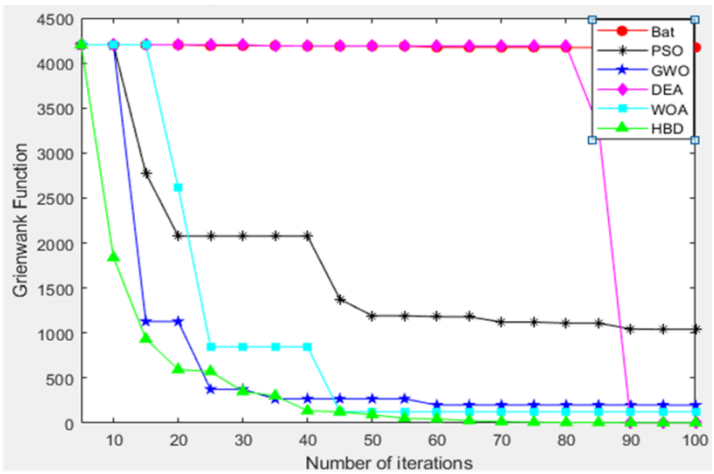


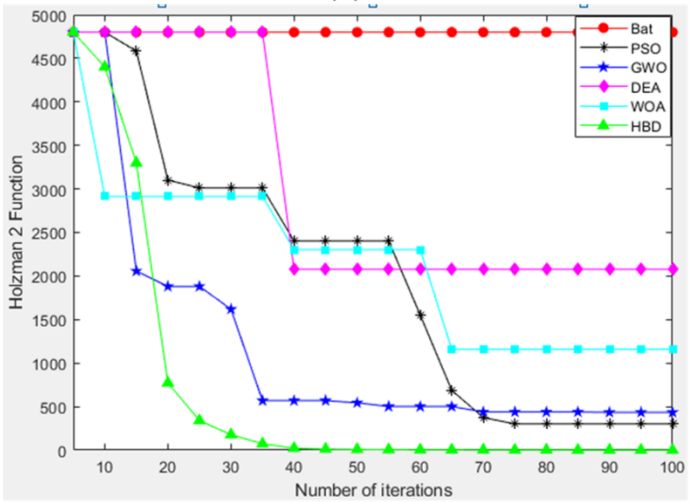
Fig. 6. 4: Convergence of HBD, Bat, PSO, DEA, WOA and GWO for Unimodal Functions


Ackley Function
Number of iterations
Bat
PSO
GWO
DEA
WOA
HBD

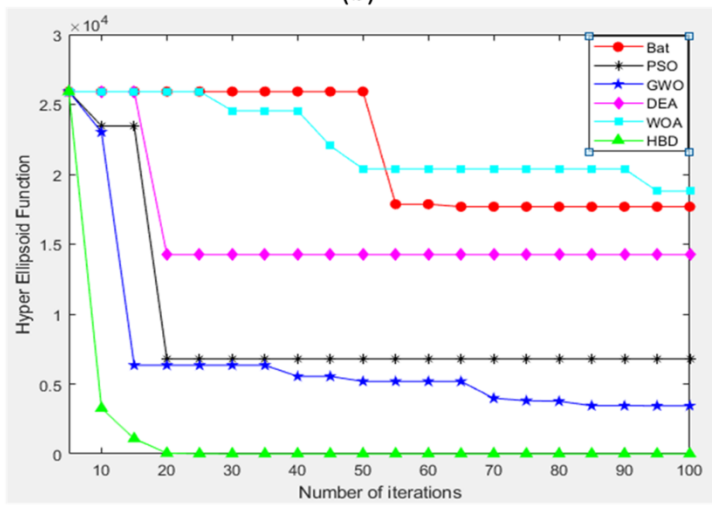
(a)


Grienwank Function
Number of iterations
Bat
PSO
GWO
DEA
WOA
HBD

(b)


Holzman 2 Function
Number of iterations
Bat
PSO
GWO
DEA
WOA
HBD

(c)


Hyper Ellipsoid Function
Number of iterations
Bat
PSO
GWO
DEA
WOA
HBD

(d)

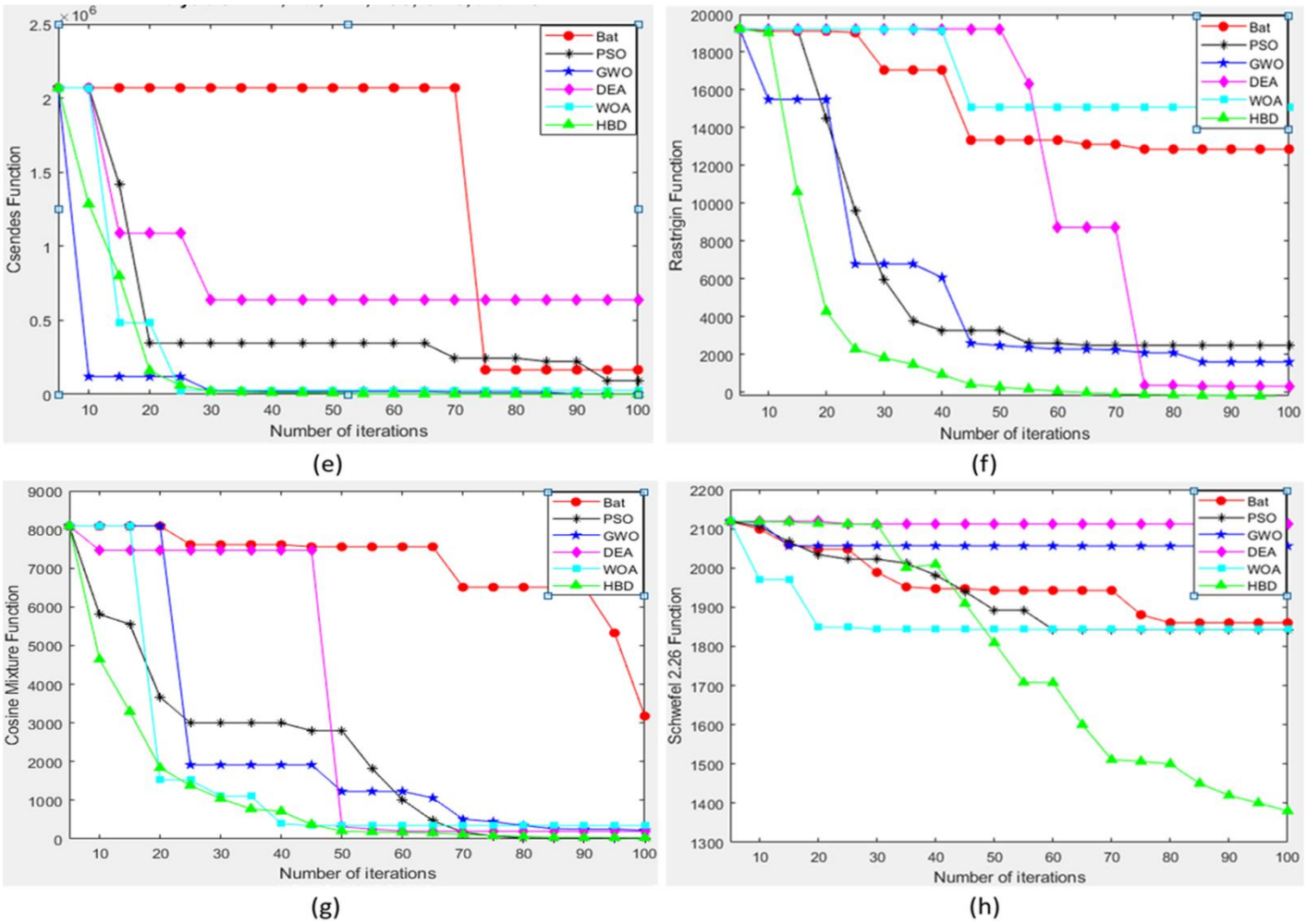


Fig. 6. 5: Convergence of HBD, Bat, PSO, DEA, WOA and GWO for Multimodal Functions

6.6.5 Analyzing the Statistical Data Obtained from the Unimodal and Multimodal Functions

For the in-depth analysis of *HBD* and its competitive algorithms over the unimodal and multimodal functions, the algorithms were simulated over 100 *Runs* for $10, 30, 100, 500$ and 1000 dimensions. The gathered data was explored using statistical parameters $i, e.$ *Means*, *Standard Deviation* and *Optimal Solutions* (values). The experimental data for unimodal functions has been presented in Table 6. 5 – Table 6. 12, while the multimodal function results are given in Table 6. 13 – Table 6. 20 (Appendix). For the experiments, the unimodal and multimodal functions are considered in Table 6. 4. While studying the results obtained for the

unimodal functions (Table 6. 5 − Table 6. 12), the proposed *HBD* algorithm has successfully generated optimal solutions, with respect of *Means*, *Standard Deviation* and *Optimal solutions*, for all unimodal functions. The ranking formula is applied to the *Mean* results of the tables in order to distinguish between each algorithm`s performance with a better understanding. The optimal results for each function have been highlighted in bold in the tables. The *HBD* algorithm ranks 1^{st} in the $Final\ Rank = 1$ in all tables (Table 6. 5 − Table 6. 12) for unimodal functions. In other words, the *HBD* algorithm has clearly shown its dominance in terms of generating optimal solutions compared to its rival algorithm in the tables (Table 6. 5 − Table 6. 12). For the results in Table 6. 13 − Table 6. 20, although the *HBD* algorithm is ranked 1^{st} as the $Final\ Rank = 1$ in all multimodal functions, in some functions *HBD* couldn't generated optimal solutions, specifically for the larger dimensions. For example, for the $n = 500$ and $n = 1000$ Ackley function, optimal values are not achieved by *HBD* but still, the algorithm confirms its ranking 1^{st} in the table. Similar trends can be seen for the Cosine Mixture and Schwefel 2.26 algorithms. However, the *HBD* algorithm has again shown its effective and impressive performance by offering optimal solutions for most cases of multimodal function. From the results of the tables (Table 6. 5 − Table 6. 12 and Table 6. 13 − Table 6. 20), it can be concluded that the *HBD* algorithm offers optimal solutions as seen in the statistical data obtained from the experiments. Thus it has proven its effective convergence effects in the form of proving better results.

6.7 Summary

This chapter offers a hybrid algorithm of the Bat and DEA algorithms, named the *"HBD* (*Hybrid Bat Dolphin Echolocation*)*"* algorithm. The purpose of offering this new hybrid algorithm is to come with a metaheuristic algorithm that can produce optimal solutions by maintaining the balance between the exploration and exploitation processes. The proposed *HBD* algorithm emphasized over echolocation features of Bat and DEA algorithms. To test the proposed *HBD* algorithm performance, two phases of experiment were implemented. One stage of the experiment was to apply the *HBD* algorithm over an application such as MPLS optimization and the second type of experiment was planned to cover the unimodal and multimodal functions. For the MPLS optimization, the MCOP-based MPLS optimization model

is presented in the chapter, in which two constrained--based objective functions are considered. The constrained-based routing costs and energy consumption costs, in addition to the objective functions and MCOP-based problem formulation, are presented in section 6.3. For the presented MPLS optimization problem, the proposed *HBD* algorithm is applied in order to produce an optimal solution. A number of experiments were conducted and comparative analyses were made using the Bat, DEA, PSO, GWO, and WOA algorithms. As the MPLS optimization model was based on multiple objectives, the *HBD* algorithm was applied to the optimization model to generate non-dominant solutions in the form of the Pareto front, as shown in Fig. 6. 2. Detailed comparative analysis was conducted in the experiments using the convergence parameters (Fig. 6. 3 and Fig. 6. 4) and statistical parameters such as *Mean, Standard Deviation and optimal solutions* (Table 6. 2 and Table 6. 3). From the results in Fig. 6. 3, Fig. 6. 4, Table 6. 2 and Table 6. 3, the proposed *HBD* algorithm showed a quick and effective convergence rate that supported the algorithm in terms of generating optimal solutions. In the experiments on the unimodal and multimodal functions, the results in Fig. 6. 5, Fig. 6. 6, Table 6. 5 – Table 6.12, and Table 6. 13 – Table 6. 20 clearly present the efficiency of the proposed *HBD* algorithm in terms of the convergence rate as well as the statistical parameters. It can be observed in the results of the chapter that the Bat and DEA algorithms had a worse convergence rate for the MPLS optimization problem and also for the benchmark functions, whereas the *HBD* algorithm impressively offered optimal solutions for the MCOP-based optimization model of MPLS networks and for the benchmark functions.

Chapter 7

Obtained Results and Future Work

7.1 Summary of Obtained Results

Advanced applications implemented over MPLS networks require better traffic management in order to avoid network congestion and to offer maximum throughput. Therefore, for MPLS optimization, the MCOP based optimization models were proposed consisting of multiple objectives based functions along with constraints. For such complex optimization problems, the novel metaheuristic algorithms were developed and analyzed. The contributions of this thesis are summarized below;

7.1.1 *PMLG-PSO* for Resource Reservation and Traffic Load Balancing Costs Functions

To address the MCOP based MPLS optimization model a modified version of PSO algorithm is proposed as *PMLG-PSO* algorithm. For MPLS optimization, the resource reservation and traffic load balancing costs objective functions are mathematically formulated as an optimization problem. The *PMLG-PSO* algorithm introduced new concept of six registers (shown in Table 4. 1), used as a technique to counter the exploration problem in the algorithm. The improved version is successfully applied to the considered MPLS optimization model that generates both Pareto based solutions together with optimal solutions for both objective functions. The experimental results of *PMLG-PSO* in the form of convergence activity and statistical measures (as *Mean*, *Standard Deviation* and *Optimal solutions*) are comprehensively compared with PSO, Bat, APSO and DA algorithms, in which *PMLG-PSO* clearly showed its dominance with remarkable outcomes. To analyze the effects of stochastic nature of *PMG-PSO* algorithm, the Krushkal Wallis test was also implemented that ensures the algorithm does not suffer from abrupt solutions for various numbers of network sizes and iterations.

7.2.1 *ABAT* for Routing Delay and Load Balancing Costs Function

A novel version of *"ABAT* (*Adjustable Bat*)*"* algorithm implanted with *loudness optimizer* is proposed for the optimization of routing delay and load balancing costs objective functions. The mathematical formulation of both objective functions are presented as MCOP based optimization model of MPLS networks. The *ABAT* was designed to offer both Pareto front based solutions along with optimal solutions. Various version of *ABAT* are presented as $ABAT-1, ABAT-2, ABAT-3, ABAT-4,$ and $ABAT-5$ with different loudness levels to highlight the role of loudness parameter in the convergence of algorithm. For this purpose the number of experimental setups were designed and implemented, which shows that the $L_{d_{max}}$ with higher value can improve te algorithm`s performance. Furthermore, the *LO* is introduced as new parameter in *ABAT* which improved the algorithm convergence rate and thus, generated optimal solutions for the given optimization problem. A comprehensive comparative analysis of *ABAT* has done with other popular algorithms i,e. Bat, CBA, PSO and DEA algorithms. The results in the form of convergence activity, generating optimal solutions and statistical data analysis showed that *ABAT* that was implemented with *LO* have brought a significant improvement in the Bat algorithm. Therefore, the results offered by *ABAT* are much superior to its competing algorithms for MPLS optimization problem.

7.3.1 *HBD* for Energy Consumption and Constrained based Routing Costs Functions

The echolocation is the common and important feature in both Bat and Dolphin Echolocation algorithms, which plays a substantial role for the convergence of the mentioned algorithm and thus helps to generate optimal solutions for complex optimization problems. Taking the beneficial features of both (Bat and DEA) algorithms, a novel algorithm named as *HBD* is developed and offered for the described MCOP based optimization model of MPLS networks. For the given optimization problem, energy consumption and constrained based routing is mathematically formulated. The proposed *HBD* algorithm has provided not only the

optimal solutions for energy consumption and constrained based routing costs functions but also had the capability to generate non-dominant solutions in the form of Pareto front graphs. The significance of the *HBD* was also proved by applying the offered algorithm over mathematical optimization functions in order to test the capability of *HBD* while comparing with Bat, DEA, PSO, GWO, and WOA algorithms. The number of experiments was conducted for testing the convergence activity of *HBD* algorithm over mentioned objective functions for MPLS networks as well as on benchmark functions. For in depth analysis, the statistical parameters were also employed to compare the *HBD* efficiency with its competing algorithms, for which *HBD* has clearly shown its effective performance compared to other aforementioned algorithms. From the results (in the form of figures and tables), the performance of *HBD* algorithm was comprehensively investigated and thus, it can be concluded that *HBD* algorithm has come as a effective approach that has successfully solved the MPLS optimization problem and also checked over number of unimodal and multimodal benchmark functions.

7.2 Future Work

The main focus of the thesis was to offer novel metaheuristic optimization algorithms as improved versions for the developed MCOP based model of MPLS networks. For this research work, PSO, Bat and DEA algorithms were considered, in which an extensive research is presented in each algorithms and the improved version is offered. However, there is still a room of research in this area, which can be considered for future research direction.

PSO algorithm has developed long ago, yet it maintains its popularity due to its flexible parameters that plays a twisting role for the improvement in algorithm`s performance. In addition to this, the PSO belongs to a family of swarm based algorithms, which depends on number of searching agents and its two parameters i, e. cognitive and social behavior feature are very important and can be considered for future research work for algorithm improvement. Beside this, the presented *PMLG-PSO* is applied for the MCOP based optimization problem of MPLS network; however this *PMLG-PSO* can also be used for different objective functions for MPLS network optimization. A comprehensive analysis of *PMLG-PSO* over benchmark functions can also be an interesting future research work direction. All these directions of research can enhance the proposed *PMLG-PSO* performance.

In this thesis an extensive research work has presented on the Bat algorithm, in which significance of *loudness parameter* is highlighted and then offered a *LO* implanted over *ABAT* algorithm that enhanced algorithm`s performance. However, the optimal value of $L_{d_{max}}$ and $L_{d_{min}}$ during the initial stage of the Bat algorithm can be a challenging task for future research. In addition to this, there are other parameters such as frequency and pulse rate in the algorithm that can also be taken for further consideration in order to understand the impact of these parameters and can improve over algorithm`s performance. The proposed *ABAT* can also be applied over benchmark mark functions for an extensive research analysis.

The proposed *HBD* algorithm is applied for MPLS optimization as well as for benchmark functions. As future research direction, the parameters in *HBD* algorithm can be play a vital role for performance improvement in the algorithm. For example, the *LO* parameter of *ABAT* can be utilized in *HBD* and then extensively studied its influence over the algorithm`s performance.

In addition to abovementioned, the optimization of MPLS network is presented and tested using simulation tool of *Matlab 2016,* this work can be deployed over MPLS based routers using the described objective functions along with proposed algorithms (*PMLG-PSO*, *ABAT*, and *HBD*). Furthermore, the presented algorithms can also be applied for the optimization of other applications such as vehicle routing problem, feature selection problem in machine learning algorithms, transportation problems and so on. Metaheuristic algorithms consist of a long list of algorithms that requires a comprehensive research for not only apply these techniques for various applications but also to enhance algorithm performance by offering improved versions.

References

[1] Tso, F., Jouet, S. and Pezaros, D., 2016. Network and server resource management strategies for data centre infrastructures: A survey. *Computer Networks*, 106, pp.209-225.

[2] Zeng, Y., Wang, D., Liu, W. and Xiong, A., 2009. An approximation algorithm for weak vertex cover problem in IP network traffic measurement. *IEEE International Conference on Network Infrastructure and Digital Content,*.

[3] Puska, A., Nogueira, M. and Santos, A., 2014. Unwanted traffic characterization on IP networks by low interactive honeypot. *10th International Conference on Network and Service Management (CNSM) and Workshop,*.

[4] Pavlou, G., n.d. Traffic engineering and quality of service management for IP-based next generation networks. *2004 IEEE/IFIP Network Operations and Management Symposium (IEEE Cat. No.04CH37507),*.

[5] Howarth, M., Boucadair, M., Flegkas, P., Wang, N., Pavlou, G., Morand, P., Coadic, T., Griffin, D., Asgari, A. and Georgatsos, P., 2006. End-to-end quality of service provisioning through inter-provider traffic engineering. *Computer Communications,* 29(6), pp.683-702.

[6] Song Xu and Jialei Wu, 2010. A QoS guaranteed MPLS multicast scheme in IPv6 network. *2010 2nd International Conference on Computer Engineering and Technology,*.

[7] Sridhar, S. and Arunnehru, J., 2018. Traffic Engineering: An Application of MPLS L3 VPN Technology. *2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI),*.

[8] Hussain, I., n.d. Overview of MPLS technology and traffic engineering applications. *International Conference on Networking and Communication, 2004. INCC 204.,*.

[9] Murakami, M. and Koike, Y., 2014. Highly Reliable and Large-Capacity Packet Transport Networks: Technologies, Perspectives, and Standardization. *Journal of Lightwave Technology*, 32(4), pp.805-816.

[10] Minei, I. and Lucek, J., 2008. *MPLS - Enabled Applications - Emerging Developments And New Technologies - 2E*. John Wiley & Sons.

[11] Ash, G., 2007. *Traffic Engineering And Qos Optimization Of Integrated Voice & Data Networks*. Amsterdam: Elsevier/Morgan Kaufmann Publishers.

[12] Köhler, S. and Binzenhöfer, A., 2003. MPLS traffic engineering in OSPF networks—A combined approach. *Providing Quality of Service in Heterogeneous Environments, Proceedings of the 18th International Teletraffic Congress - ITC-18*, pp.21-30.

[13] Dahai Xu, Mung Chiang and Rexford, J., 2011. Link-State Routing With Hop-by-Hop Forwarding Can Achieve Optimal Traffic Engineering. *IEEE/ACM Transactions on Networking*, 19(6), pp.1717-1730.

[14] Li, F. and Chen, J., 2012. MPLS Traffic Engineering Load Balance Algorithm Using Deviation Path. *2012 International Conference on Computer Science and Service System*,.

[15] Awduche, D. and Jabbari, B., 2002. Internet traffic engineering using multi-protocol label switching (MPLS). *Computer Networks*, 40(1), pp.111-129.

[16] Haßlinger, G. and Schnitter, S., 2003. Optimized Traffic Load Distribution in MPLS Networks. *Operations Research/Computer Science Interfaces Series*, pp.125-141.

[17] Garroppo, R., Giordano, S. and Tavanti, L., 2010. A survey on multi-constrained optimal path computation: Exact and approximate algorithms. *Computer Networks*, 54(17), pp.3081-3107.

[18] Girão-Silva, R., Craveirinha, J., Clímaco, J. and Captivo, M., 2015. Multiobjective routing in multiservice MPLS networks with traffic splitting — A network flow

approach. *Journal of Systems Science and Systems Engineering*, 24(4), pp.389-432.

[19] Neely, M., 2010. Stochastic Network Optimization with Application to Communication and Queueing Systems. *Synthesis Lectures on Communication Networks*, 3(1), pp.1-211.

[20] Yang, X., 2010. *Engineering Optimization*. Hoboken, NJ: Wiley.

[21] Ming, M., Wang, R. and Zhang, T., 2019. Evolutionary Many-Constraint Optimization: An Exploratory Analysis. *Lecture Notes in Computer Science*, pp.165-176.

[22] Ehrgott, M., 2004. *Multicriteria Optimization*. New York: Springer.

[23] Dokeroglu, T., Sevinc, E., Kucukyilmaz, T. and Cosar, A., 2019. A survey on new generation metaheuristic algorithms. *Computers & Industrial Engineering*, 137, p.106040.

[24] Yang, X., 2013. Metaheuristic Optimization: Nature-Inspired Algorithms and Applications. *Studies in Computational Intelligence*, pp.405-420.

[25] Hussain, K., Salleh, M., Cheng, S. and Shi, Y., 2018. On the exploration and exploitation in popular swarm-based metaheuristic algorithms. *Neural Computing and Applications*, 31(11), pp.7665-7683.

[26] Morales-Castañeda, B., Zaldívar, D., Cuevas, E., Fausto, F. and Rodríguez, A., 2020. A better balance in metaheuristic algorithms: Does it exist?. *Swarm and Evolutionary Computation*, 54, p.100671.

[27] Giambene, G., 2014. IP-Based Networks and Future Trends. *Queuing Theory and Telecommunications*, pp.129-262.

[28] Ergen, M., 2008. Basics of All-IP Networking. *Mobile Broadband*, pp.67-106.

[29] Sun, Z., 2011. IP Networking and Future Evolution. *Network Performance Engineering*, pp.951-978.

[30] Minei, I. and Lucek, J., 2011. *MPLS-Enabled Applications*. Chichester, West Sussex,

England: Wiley.

[31] Farrel, A. and Bryskin, I., 2006. *GMPLS*. San Francisco, CA: Morgan Kaufmann.

[32] Bruce S., D. A. Farrel., 2008. MPLS Next Steps, The Morgan Kaufmann Series in Networking, Elsevier Publisher.

[33] De Ghein, L., n.d. *MPLS Fundamentals*. Cisco Publishers.

[34] Osborne E., Simha A., 2002. *Traffic Engineering With MPLS*. Cisco Press.

[35] Medhi, D. and Ramasamy, K., 2018. Routing and Traffic Engineering using MPLS. *Network Routing*, pp.766-785.

[36] Hodzic, H. and Zoric, S., 2008. Traffic engineering with constraint based routing in MPLS networks. *IEEE 50th International Symposium ELMAR,*.

[37] Dahai Xu, Mung Chiang and Rexford, J., 2011. Link-State Routing With Hop-by-Hop Forwarding Can Achieve Optimal Traffic Engineering. *IEEE/ACM Transactions on Networking*, 19(6), pp.1717-1730.

[38] Tychogiorgos, G. and Leung, K., 2014. Optimization-based resource allocation in communication networks. *Computer Networks*, 66, pp.32-45.

[39] Maggie Xiaoyan Cheng., Yingshu Li., Ding-Zhu Du., Combinatorial Optimization in Communication Networks, Springer Publisher.

[40] Gu, F., Liu, Z., Cheung, Y. and Liu, H., 2017. Optimization of Spectrum-Energy Efficiency in Heterogeneous Communication Network. *Lecture Notes in Computer Science*, pp.821-832.

[41] Lemeshko, O. and Yeremenko, O., 2018. Linear optimization model of MPLS Traffic Engineering Fast ReRoute for link, node, and bandwidth protection. *2018 14th International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET),*.

[42] Onety, R., Tadei, R., Neto, O. and Takahashi, R., 2013. Multiobjective optimization of MPLS-IP networks with a variable neighborhood genetic algorithm. *Applied Soft Computing*, 13(11), pp.4403-4412.

[43] Kochenderfer, M. and Wheeler, T., 2019. *Algorithms For Optimization*. The MIT Press Cambridge, Massachusetts London, England.

[44] Neumann, F. and Witt, C., 2013. *Bioinspired Computation In Combinatorial Optimization*. Berlin: Springer Berlin.

[45] Du, K. and Swamy, M., 2010. *Search And Optimization By Metaheuristics*. Birkhäuser.

[46] Dinh T. Luc., 2016. *Multiobjective Linear Programming: An Introduction*. 1st ed. Springer International Publishing.

[47] Garroppo, R., Giordano, S. and Tavanti, L., 2010. A survey on multi-constrained optimal path computation: Exact and approximate algorithms. *Computer Networks*, 54(17), pp.3081-3107.

[48] Yang, X. and Karamanoglu, M., 2020. Nature-inspired computation and swarm intelligence: a state-of-the-art overview. *Nature-Inspired Computation and Swarm Intelligence*, pp.3-18.

[49] Saroj, Vashishtha, J., Goyal, P. and Ahuja, J., 2018. A Novel Fitness Computation Framework for Nature Inspired Classification Algorithms. *Procedia Computer Science*, 132, pp.208-217.

[50] Khosravy, M., 2020. *Frontier Applications Of Nature Inspired Computation*. Singapore: Springer Nature.

[51] Mavrovouniotis, M., Li, C. and Yang, S., 2017. A survey of swarm intelligence for dynamic optimization: Algorithms and applications. *Swarm and Evolutionary Computation*, 33, pp.1-17.

[52] Beg, A. and Islam, M., 2016. Advantages and limitations of genetic algorithms for

clustering records. *2016 IEEE 11th Conference on Industrial Electronics and Applications (ICIEA)*,.

[53] Zhan, S., Lin, J., Zhang, Z. and Zhong, Y., 2016. List-Based Simulated Annealing Algorithm for Traveling Salesman Problem. *Computational Intelligence and Neuroscience*, 2016, pp.1-12.

[54] Cuevas, E., Osuna, V. and Oliva, D., n.d. *Evolutionary Computation Techniques: A Comparative Perspective*.

[55] Babu, B. V., Gujarathi, Ashish M,. 2017. *Evolutionary Computation : Techniques And Applications*. Apple Academic Press Inc.

[56] Affenzeller, M., Beham, A., Kofler, M., Kronberger, G., Wagner, S. and Winkler, S., 2010. Metaheuristic Optimization. *Hagenberg Research*, pp.103-155.

[57] Samir, M., Salim, C. and Ahmed, T., 2012. A meta-exact approach to improve intensification in combinatorial optimization problems. *2012 International Conference on Information Technology and e-Services*,.

[58] Talbi, E., 2009. *Metaheuristics: From Design To Implementation*. Hoboken, NJ: Wiley.

[59] Hromkovič, J., 2011. *Algorithmics For Hard Problems*. Berlin: Springer.

[60] Haddad, B., 2017. *Meta-Heuristic And Evolutionary Algorithms For Engineering Optimization*. John Wiley & Sons.

[61] Hussain, K., Mohd Salleh, M., Cheng, S. and Shi, Y., 2018. Metaheuristic research: a comprehensive survey. *Artificial Intelligence Review*, 52(4), pp.2191-2233.

[62] Rajakumar, R., Dhavachelvan, P. and Vengattaraman, T., 2016. A survey on nature inspired meta-heuristic algorithms with its domain specifications. *2016 International Conference on Communication and Electronics Systems (ICCES)*,.

[63] Chen, P. and Zeng, Z., 2020. Developing two heuristic algorithms with metaheuristic

algorithms to improve solutions of optimization problems with soft and hard constraints: An application to nurse rostering problems. *Applied Soft Computing*, 93, p.106336.

[64] Glover, F., 1989. Tabu Search—Part I. *ORSA Journal on Computing*, 1(3), pp.190-206.

[65] Konstantinov, S., Diveev, A., Balandina, G. and Baryshnikov, A., 2019. Comparative Research of Random Search Algorithms and Evolutionary Algorithms for the Optimal Control Problem of the Mobile Robot. *Procedia Computer Science*, 150, pp.462-470.

[66] Lourenço, H., Martin, O. and Stützle, T., 2018. Iterated Local Search: Framework and Applications. *Handbook of Metaheuristics*, pp.129-168.

[67] De Carvalho, M., 2012. A generalization of the Solis–Wets method. *Journal of Statistical Planning and Inference*, 142(3), pp.633-644.

[68] Hansen, P. and Mladenović, N., 2018. Variable Neighborhood Search. *Handbook of Heuristics*, pp.759-787.

[69] Voudouris, C. and Tsang, E., 2003. Guided Local Search. *Handbook of Metaheuristics*, pp.185-218.

[70] Hussain, K., Mohd Salleh, M., Cheng, S. and Shi, Y., 2017. Comparative Analysis of Swarm-Based Metaheuristic Algorithms on Benchmark Functions. *Lecture Notes in Computer Science*, pp.3-11.

[71] Quan, H. and Shi, X., 2009. Evolutionary Swarm Algorithm Based on Evolutionary Swarm Model. *2009 WRI Global Congress on Intelligent Systems,*.

[72] Ishibuchi, H., Hitotsuyanagi, Y., Tsukamoto, N. and Nojima, Y., 2008. Use of Heuristic Local Search for Single-Objective Optimization in Multiobjective Memetic Algorithms. *Parallel Problem Solving from Nature – PPSN X*, pp.743-752.

[73] Gawiejnowicz, S., 2020. Local search and meta-heuristic algorithms. *Monographs in Theoretical Computer Science. An EATCS Series*, pp.329-351.

[74] Liu, Q., Li, X., Liu, H. and Guo, Z., 2020. Multi-objective metaheuristics for discrete optimization problems: A review of the state-of-the-art. *Applied Soft Computing*, 93, p.106382.

[75] Caserta, M. and Voß, S., 2009. Metaheuristics: Intelligent Problem Solving. *Matheuristics*, pp.1-38.

[76] Bianchi, L., Dorigo, M., Gambardella, L. and Gutjahr, W., 2008. A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing*, 8(2), pp.239-287.

[77] Dokeroglu, T., Sevinc, E., Kucukyilmaz, T. and Cosar, A., 2019. A survey on new generation metaheuristic algorithms. *Computers & Industrial Engineering*, 137, p.106040.

[78] Ramos-Figueroa, O., Quiroz-Castellanos, M., Mezura-Montes, E. and Schütze, O., 2020. Metaheuristics to solve grouping problems: A review and a case study. *Swarm and Evolutionary Computation*, 53, p.100643.

[79] Kennedy, J. and Eberhart, R., 1995. Particle swarm optimization. *Proceedings of ICNN'95 - International Conference on Neural Networks,*.

[80] Lambora, A., Gupta, K. and Chopra, K., 2019. Genetic Algorithm- A Literature Review. *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon),*.

[81] Yang, X., 2010. A New Metaheuristic Bat-Inspired Algorithm. *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*, pp.65-74.

[82] Mandal, A. and Dehuri, S., 2020. A Survey on Ant Colony Optimization for Solving Some of the Selected NP-Hard Problem. *Learning and Analytics in Intelligent Systems*, pp.85-100.

[83] Kaveh, A. and Farhoudi, N., 2013. A new optimization method: Dolphin echolocation.

Advances in Engineering Software, 59, pp.53-70.

[84] Mirjalili, S., Mirjalili, S. and Lewis, A., 2014. Grey Wolf Optimizer. *Advances in Engineering Software*, 69, pp.46-61.

[85] Joshi, A., Kulkarni, O., Kakandikar, G. and Nandedkar, V., 2017. Cuckoo Search Optimization- A Review. *Materials Today: Proceedings*, 4(8), pp.7262-7269.

[86] Yuce, B., Packianather, M., Mastrocinque, E., Pham, D. and Lambiase, A., 2013. Honey Bees Inspired Optimization Method: The Bees Algorithm. *Insects*, 4(4), pp.646-662.

[87] Xu, J. and Zhang, J., 2014. Exploration-exploitation tradeoffs in metaheuristics: Survey and analysis. *Proceedings of the 33rd Chinese Control Conference*,.

[88] Tilahun, S., 2019. Balancing the Degree of Exploration and Exploitation of Swarm Intelligence Using Parallel Computing. *International Journal on Artificial Intelligence Tools*, 28(03), p.1950014.

[89] Salleh, M., Hussain, K., Cheng, S., Shi, Y., Muhammad, A., Ullah, G. and Naseem, R., 2018. Exploration and Exploitation Measurement in Swarm-Based Metaheuristic Algorithms: An Empirical Analysis. *Advances in Intelligent Systems and Computing*, pp.24-32.

[90] Hussain, K., Salleh, M., Cheng, S. and Shi, Y., 2018. On the exploration and exploitation in popular swarm-based metaheuristic algorithms. *Neural Computing and Applications*, 31(11), pp.7665-7683.

[91] Morales-Castañeda, B., Zaldívar, D., Cuevas, E., Fausto, F. and Rodríguez, A., 2020. A better balance in metaheuristic algorithms: Does it exist?. *Swarm and Evolutionary Computation*, 54, p.100671.

[92] Patel, M., Kabat, M. and Tripathy, C., 2014. A hybrid ACO/PSO based algorithm for QoS multicast routing problem. *Ain Shams Engineering Journal*, 5(1), pp.113-120.

[93] Khamsawang, S., Wannakarn, P. and Jiriwibhakorn, S., 2010. Hybrid PSO-DE for

solving the economic dispatch problem with generator constraints. *2010 The 2nd International Conference on Computer and Automation Engineering (ICCAE)*,.

[94] Garg, H., 2016. A hybrid PSO-GA algorithm for constrained optimization problems. *Applied Mathematics and Computation*, 274, pp.292-305.

[95] Wang, G., Gandomi, A., Alavi, A. and Hao, G., 2013. Hybrid krill herd algorithm with differential evolution for global numerical optimization. *Neural Computing and Applications*, 25(2), pp.297-308.

[96] Zhang, Z., Dong, Y. and Gao, T., 2016. A Hybrid Method Based on Cuckoo Search and Krill Herd Optimization with Differential Evolution. *2016 13th Web Information Systems and Applications Conference (WISA)*,.

[97] Zhang X., Duan H., and Jin J., 2008. DEACO: Hybrid Ant Colony Optimization with Differential Evolution. *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*,.

[98] Lin, W., 2010. A GA–DE hybrid evolutionary algorithm for path synthesis of four-bar linkage. *Mechanism and Machine Theory*, 45(8), pp.1096-1107.

[99] Nemati, S., Basiri, M., Ghasem-Aghaee, N. and Aghdam, M., 2009. A novel ACO–GA hybrid algorithm for feature selection in protein function prediction. *Expert Systems with Applications*, 36(10), pp.12086-12094.

[100] Wolpert, D. and Macready, W., 1997. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1), pp.67-82.

[101] Mirjalili, S., 2016. SCA: A Sine Cosine Algorithm for solving optimization problems. *Knowledge-Based Systems*, 96, pp.120-133.

[102] Doolittle, E., Kerivin, H. and Wiecek, M., 2018. Robust multiobjective optimization with application to Internet routing. *Annals of Operations Research*, 271(2), pp.487-525.

[103] Girão-Silva, R., Craveirinha, J. and Clímaco, J., 2014. Stochastic hierarchical

multiobjective routing model in MPLS networks with two service classes: an experimental study on imprecision and uncertainty issues. *Journal of Uncertainty Analysis and Applications*, 2(1).

[104] El-Alfy, E., Mujahid, S. and Selim, S., 2013. A Pareto-based hybrid multiobjective evolutionary approach for constrained multipath traffic engineering optimization in MPLS/GMPLS networks. *Journal of Network and Computer Applications*, 36(4), pp.1196-1207.

[105] Zhang, Y., Wang, S. and Ji, G., 2015. A Comprehensive Survey on Particle Swarm Optimization Algorithm and Its Applications. *Mathematical Problems in Engineering*, 2015, pp.1-38.

[106] Seixas Gomes de Almeida, B. and Coppo Leite, V., 2019. Particle Swarm Optimization: A Powerful Technique for Solving Engineering Problems. *Swarm Intelligence - Recent Advances, New Perspectives and Applications*,.

[107] Sun, J., Lai, C. and Wu, X., 2016. Particle Swarm Optimisation: Classical and Quantum Perspective.

[108] Masood, M., Fouad, M., Kamal, R., Glesk, I. and Khan, I., 2019. An Improved Particle Swarm Algorithm for Multi-Objectives Based Optimization in MPLS/GMPLS Networks. *IEEE Access*, 7, pp.137147-137162.

[109] Akjiratikarl, C., Yenradee, P. and Drake, P., 2007. PSO-based algorithm for home care worker scheduling in the UK. *Computers & Industrial Engineering*, 53(4), pp.559-583.

[110] Du, Z., Li, S., Sun, Y. and Li, N., 2017. Adaptive particle swarm optimization algorithm based on levy flights mechanism. *2017 Chinese Automation Congress (CAC)*,.

[111] Aziz, M., Taib, M. and Hussin, N., 2010. Assignments acceptance strategy in a Modified PSO Algorithm to elevate local optima in solving class scheduling problems. *2010 6th International Colloquium on Signal Processing & its Applications*,.

[112] Mousavi, S., Moghadasi, M. and Fazekas, G., 2017. Dynamic resource allocation using combinatorial methods in Cloud: A case study. *2017 8th IEEE International Conference on Cognitive Infocommunications (CogInfoCom),*.

[113] Liang, J., Qin, A., Suganthan, P. and Baskar, S., 2006. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Transactions on Evolutionary Computation*, 10(3), pp.281-295.

[114] Dong, C., Wang, G., Chen, Z. and Yu, Z., 2008. A Method of Self-Adaptive Inertia Weight for PSO. *2008 International Conference on Computer Science and Software Engineering*,.

[115] Ghodousian, A. and Parvari, M., 2017. A modified PSO algorithm for linear optimization problem subject to the generalized fuzzy relational inequalities with fuzzy constraints (FRI-FC). *Information Sciences*, 418-419, pp.317-345.

[116] Hossain, M., Pota, H., Squartini, S. and Abdou, A., 2019. Modified PSO algorithm for real-time energy management in grid-connected microgrids. *Renewable Energy*, 136, pp.746-757.

[117] Yan, Y., Zhang, R., Wang, J. and Li, J., 2018. Modified PSO algorithms with "Request and Reset" for leak source localization using multiple robots. *Neurocomputing*, 292, pp.82-90.

[118] Choudhary, A., Kumar, M., Gupta, M., Unune, D. and Mia, M., 2019. Mathematical modeling and intelligent optimization of submerged arc welding process parameters using hybrid PSO-GA evolutionary algorithms. *Neural Computing and Applications*, 32(10), pp.5761-5774.

[119] Mahapatra, S., Badi, M. and Raj, S., 2019. Implementation of PSO, it's variants and Hybrid GWO-PSO for improving Reactive Power Planning. *2019 Global Conference for Advancement in Technology (GCAT)*,.

[120] Liu, Y., Dai, J., Zhao, S., Zhang, J., Shang, W., Li, T., Zheng, Y., Lan, T. and Wang, Z.,

2020. Optimization of five-parameter BRDF model based on Hybrid GA-PSO algorithm. *Optik*, p.164978.

[121] Salehizadeh, S., Yadmellat, P. and Menhaj, M., 2009. Local Optima Avoidable Particle Swarm Optimization. *2009 IEEE Swarm Intelligence Symposium*,.

[122] Al-Bahrani, L. and Chandra Patra, J., 2018. Multi-gradient PSO algorithm for optimization of multimodal, discontinuous and non-convex fuel cost function of thermal generating units under various power constraints in smart power grid. *Energy*, 147, pp.1070-1091.

[123] Parsopoulos, K. and Vrahatis, M., 2005. Unified Particle Swarm Optimization for Solving Constrained Engineering Optimization Problems. *Lecture Notes in Computer Science*, pp.582-591.

[124] Mendes, R., Kennedy, J. and Neves, J., 2004. The Fully Informed Particle Swarm: Simpler, Maybe Better. *IEEE Transactions on Evolutionary Computation*, 8(3), pp.204-210.

[125] Peram, T., Veeramachaneni, K. and Mohan, C., n.d. Fitness-distance-ratio based particle swarm optimization. *Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS'03 (Cat. No.03EX706)*,.

[126] Masar, M. and Zelenka, J., 2012. Modification of PSO algorithm for the purpose of space exploration. *2012 IEEE 16th International Conference on Intelligent Engineering Systems (INES)*,.

[127] Kai-Shixu and Yi, Y., 2015. A new particle swarm algorithm by modifying its topology structure. *2015 International Conference on Machine Learning and Cybernetics (ICMLC)*,.

[128] Chen, D., Chen, J., Jiang, H., Zou, F. and Liu, T., 2014. An improved PSO algorithm based on particle exploration for function optimization and the modeling of chaotic systems. *Soft Computing*, 19(11), pp.3071-3081.

[129] Liu, T., Li, L., Shao, G., Wu, X. and Huang, M., 2020. A novel policy gradient algorithm with PSO-based parameter exploration for continuous control. *Engineering Applications of Artificial Intelligence*, 90, p.103525.

[130] Liu, H. and Han, F., 2017. A Modified Standard PSO-2011 with Robust Search Ability. *Communications in Computer and Information Science*, pp.207-222.

[131] Rudhrra Priyaa, A., Rini Tonia, E. and Manikandan, N., 2020. Resource Scheduling Using Modified FCM and PSO Algorithm in Cloud Environment. *Second International Conference on Computer Networks and Communication Technologies*, pp.697-704.

[132] Hernafi, Y., Ben Ahmed, M. and Bouhorma, M., 2017. ACO and PSO Algorithms for Developing a New Communication Model for VANET Applications in Smart Cities. *Wireless Personal Communications*, 96(2), pp.2039-2075.

[133] Chaudhry, R., Tapaswi, S. and Kumar, N., 2018. Forwarding Zone enabled PSO routing with Network lifetime maximization in MANET. *Applied Intelligence*, 48(9), pp.3053-3080.

[134] Goudarzi, S., Hassan, W., Anisi, M., Soleymani, A., Sookhak, M., Khan, M., Hashim, A. and Zareei, M., 2017. ABC-PSO for vertical handover in heterogeneous wireless networks. *Neurocomputing*, 256, pp.63-81.

[135] Udhayakumar, S., Raja Kumar, R. and Indumathi, P., 2018. Network Selection in Wireless Heterogeneous Network. *2018 International Conference on Recent Trends in Electrical, Control and Communication (RTECC)*.

[136] Craveirinha, J., Girão-Silva, R. & Clímaco, J. , 2008. A meta-model for multiobjective routing in MPLS networks. Central European Journal of Operations Research, 16(1):79–105.

[137] Girão-Silva, R., Craveirinha, J. & Clímaco, J., 2009. Hierarchical multiobjective routing in Multiprotocol Label Switching networks with two service classes – A heuristic solution. International Transactions in Operational Research, 16(3):275–305.

[138] Craveirinha, J. M. F., Clímaco, J. C. N., Pascoal, M. M. B. & Martins, L. M. R. A., 2007). Traffic splitting in MPLS networks – A hierarchical multicriteria approach. Journal of Telecommunications and Information Technology, (4):3–10.

[139] Messac, A., Ismail-Yahaya, A. & Mattson, C. A., 2003. The normalized normal constraint method for generating the Pareto frontier. Structural and Multidisciplinary Optimization, 25(2):86–98.

[140] Mavrotas, G., 2009. Effective implementation of the ε-constraint method in Multi-Objective Mathematical Programming problems. Applied Mathematics and Computation, 213(2):455–465.

[141] S.-H. Kim, K.-N. Chang, S. Kim, 2000. A channel allocation for cellular mobile radio systems using simulated annealing, Telecommunication Systems 14, 95–106.

[142] M. Randall, G. McMahon, S. Sugden, 2002. A simulated annealing approach to communication network design, Journal of Combinatorial Optimization 6, 55–65.

[143] E. Amaldi, A. Capone, F. Malucelli, 2003. Planning UMTS base station location: optimization models with power control and algorithms, IEEE Transactions on Wireless Communications 2, 939–952.

[144] L.A. Cox, J.R. Sanchez, 2000. Designing least-cost survivable wireless backhaul networks, Journal of Heuristics 6, 525–540.

[145] A. Fink, G. Schneidereit, S. Voss, 1999. Solving general ring network design problems by meta-heuristics, in: M. Laguna, J.L. Gonza´lez (Eds.), Computing Tools for Modeling, Optimization and Simulation, Kluwer, Dordrecht, pp. 91–113.

[146] B. Gendron, J.-Y. Potvin, P. Soriano, 2000. Diversification strategies in local search for a nonbifurcated network loading problem, European Journal of Operational Research 142, 231–241.

[147] V. Gabrel, A. Knippel, M. Minoux, 2003. A comparison of heuristics for the discrete

cost multicommodity network optimization problem, Journal of Heuristics 9, 429–445.

[148] M.G.C. Resende, C.C. Ribeiro, 2003. A GRASP with path-relinking for private virtual circuit routing, Networks 41 (2003) 104–114.

[149] M. Armony, J.C. Klincewicz, H. Luss, M.B. Rosenwein, 2000. Design of stacked self-healing rings using a genetic algorithm, Journal of Heuristics 6, 85–105.

[150] L.S. Buriol, M.G.C. Resende, C.C. Ribeiro, M. Thorup, 2005. A hybrid genetic algorithm for the weight setting problem in OSPF/IS-IS routing, Networks 46, 36–56.

[151] G.N. Varela, M.C. Sinclair, 1999. Ant colony optimisation for virtualwavelength- path routing and wavelength allocation, in: P.J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, A. Zalzala (Eds.), Proceedings of the Congress on Evolutionary Computation, vol. 3, IEEE Press, Washington, DC, pp. 1809–1816.

[152] O. Wittner, P.E. Heegaard, B. Helvik, 2003. Scalable distributed discovery of resource paths in telecommunication networks using cooperative ant-like agents, in: Proceedings of the Congress on Evolutionary Computation, Canberra.

[153] Li, B. and Wada, K., 2011. Communication latency tolerant parallel algorithm for particle swarm optimization. *Parallel Computing*, 37(1), pp.1-10.

[154] Soo, K., Siu, Y., Chan, W., Yang, L. and Chen, R., 2007. Particle-Swarm-Optimization-Based Multiuser Detector for CDMA Communications. *IEEE Transactions on Vehicular Technology*, 56(5), pp.3006-3013.

[155] Jordehi, A., 2015. Enhanced leader PSO (ELPSO): A new PSO variant for solving global optimisation problems. *Applied Soft Computing*, 26, pp.401-417.

[156] Khan, S., Yang, S. and Ur Rehman, O., 2017. A global particle swarm optimization algorithm applied to electromagnetic design problem. *International Journal of Applied Electromagnetics and Mechanics*, 53(3), pp.451-467.

[157] Mehmood, Y., 2020. *Novel Particle Swarm Optimization Algorithm For Multimodal*

Optimization Problems By Enhancing The Robustness And Diversity.

[158] Chen, K., Zhou, F., Wang, Y. and Yin, L., 2018. An ameliorated particle swarm optimizer for solving numerical optimization problems. *Applied Soft Computing*, 73, pp.482-496.

[159] Adsawinnawanawa, E. and Kruatrachue, B., 2020. Mutation Variations in Improving Local Optima Problem of PSO. *Recent Advances in Information and Communication Technology 2020*, pp.149-158.

[160] Mehmood, Y., Aziz, N., Riaz, F., Iqbal, H. and Shahzad, W., 2018. PSO-based clustering techniques to solve multimodal optimization problems: A survey. *2018 1st International Conference on Power, Energy and Smart Grid (ICPESG)*.

[161] Sánchez-García, J., Reina, D. and Toral, S., 2019. A distributed PSO-based exploration algorithm for a UAV network assisting a disaster scenario. *Future Generation Computer Systems*, 90, pp.129-148.

[162] Pluhacek, M., Senkerik, R., Viktorin, A. and Kadavy, T., 2017. Exploring the shortest path in PSO communication network. *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*.

[163] Yu, X., Bai, C., Xu, H., Sun, W., Yang, L., Zheng, H. and Hu, W., 2020. A modified PSO assisted blind modulation format identification scheme for elastic optical networks. *Optics Communications*, p.126280.

[164] Shi, L., Zhan, Z., Yuan, H., Li, J. and Zhang, J., 2017. Distributed co-evolutionary particle swarm optimization using adaptive migration strategy. *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*,.

[165] Burke, E. and Kendall, G., 2014. *Search Methodologies*. Boston, MA: Springer US.

[166] Zitzler, E., Laumanns, M. and Bleuler, S., 2004. A Tutorial on Evolutionary Multiobjective Optimization. *Lecture Notes in Economics and Mathematical Systems*,

pp.3-37.

[167] Zhan, Z. and Zhang, J., n.d. Adaptive Particle Swarm Optimization. *Ant Colony Optimization and Swarm Intelligence*, pp.227-234.

[168] *An Introduction to Statistical Analysis in Research*, 2017. Kruskal-Wallis. pp.353-391.

[169] Masood, M., Fouad, M. and Glesk, I., 2017. Proposing bat inspired heuristic algorithm for the optimization of GMPLS networks. *2017 25th Telecommunication Forum (TELFOR)*,.

[170] Masood, M., Fouad, M. and Glesk, I., 2018. Pareto Based Bat Algorithm for Multi Objectives Multiple Constraints Optimization in GMPLS Networks. *The International Conference on Advanced Machine Learning Technologies and Applications (AMLTA2018)*, pp.33-41.

[171] Ponce, P., Mantilla, A., Molina, A., Paez, A., Tello, T., Zepeda, S., 2016. Swarm-based nature-inspired algorithm and genetic algorithms for optimizing a sun tracker trajectory. Appl. Artif. Intell. 30 (2), 97–124.

[172] Ayadi, O., Masmoudi, M., Ben Ameur, M., Masmoudi, F., 2017. A new PSO-based algorithm for two-dimensional non-guillotine non-oriented cutting stock problem. Appl. Artif. Intell. (23673), 1–18

[173] Rehman, M., Zamli, K. and Nasser, A., 2020. An Improved Genetic Bat algorithm for Unconstrained Global Optimization Problems. *Proceedings of the 2020 9th International Conference on Software and Computer Applications*.

[174] Wang, X., Wang, W. and Wang, Y., 2013. An Adaptive Bat Algorithm. *Intelligent Computing Theories and Technology*, pp.216-223.

[175] Yang, X., Deb, S., Fong, S., 2014. Bat Algorithm is better than intermittent search strategy. Multiple-Valued Logic Soft Computer 22 (3), 223–237.

[176] Alomari, O.A., Khader, A.T., Al-Betar, M.A., Abualigah, L.M., 2017. MRMR BA: a

hybrid gene selection algorithm for cancer classification. J. Theory Appl. Inf. Technol. 95 (12), 2610–2618.

[177] Arora, S., Singh, S., 2013. A conceptual comparison of firefly algorithm, bat algorithm and cuckoo search. In: 2013 International Conference on Control, Computing, Communication and Materials, ICCCCM 2013, (Iccccm), pp.1–4.

[178] Talal, R., 2014. Comparative Study between the (BA) Algorithm and (PSO) Algorithm to Train (RBF) Network at Data Classification, 92 (5), pp.16–22.

[179] Wang, G., Lu, M., Zhao, X., Wang, E.G., Lu, M., Zhao, X., 2016. An Improved Bat Algorithm with Variable Neighborhood Search for Global. Optimization, 1773– 1778.

[180] Ramli, M., Abas, Z., Desa, M., Abidin, Z. and Alazzam, M., 2019. Enhanced convergence of Bat Algorithm based on dimensional and inertia weight factor. *Journal of King Saud University - Computer and Information Sciences*, 31(4), pp.452-458.

[181] Yang, X., 2011. Bat Algorithm for Multiobjective Optimization. Int. J. Bio-Inspired Comput. 3 (5), 267–274.

[182] Komarasamy, G., Wahi, A., 2012. An optimized k-means clustering technique using bat algorithm. Eur. J. Sci. Res. 84 (2), 263–273.

[183] Khan, K., Nikov, A., Sahai, A., 2011. A fuzzy bat clustering method for ergonomic screening of office workspaces. Adv. Intell. Comput. 101, 59–66.

[184] Nakamura, R.Y.M., Pereira, L.A.M., Costa, K.A., Rodrigues, D., Papa, J.P., Yang, X.S., 2012. BBA: A binary bat algorithm for feature selection. Brazilian Symp. Computer Graphic Image Process., 291–297

[185] Fister, I., Brest, J., Yang, X.S., 2015. Modified bat algorithm with quaternion representation. IN: 2015 IEEE Congress on Evolutionary Computation, CEC 2015– Proceedings, (September), pp.491–498.

[186] Yilmaz, S., Kucuksille, E.U., 2013. Improved Bat Algorithm (IBA) on Continuous

Optimization Problems. Lecture Notes Softw. Eng. 1 (3), 279–283.

[187] Lin, J., Chou, C., Yang, C., Tsai, H., 2010. A Chaotic Levy Flight Bat Algorithm for Parameter Estimation in Nonlinear Dynamic Biological Systems. J. Comput. Inf. Technol. 2 (2), 56–63.

[188] Xie, J., Zhou, Y., Chen, H., 2013. A novel bat algorithm based on differential operator and Levy flights trajectory. Comput. Intell. Neurosci.

[189] Jamil et al., 2013. Improved bat algorithm for global optimization. Appl. Soft Computer pp. 2013.

[190] Li, L., Zhou, Y., 2014. A novel complex-valued bat algorithm. Neural Comput. Appl. 25 (6), 1369–1381.

[191] Ali, E.S., 2014. Optimization of Power System Stabilizers using BAT search algorithm. Int. J. Electr. Power Energy Syst. 61 (2014), 683–690.

[192] Alihodzic, A., Tuba, E., Tuba, M., n.d. An Upgraded Bat Algorithm for Tuning Extreme Learning Machines for Data Classification, pp.125–126.

[193] Rizk-Allah, R.M., Hassanien, A.E., 2017. New binary bat algorithm for solving 0–1 knapsack problem. Complex Intelligent Syst.

[194] Yahya, Nafrizuan Mat, M.O.T., 2017. A modified bats echolocation-based algorithm for solving constrained optimisation problems. Int. J. Bio-Inspired Comput. 10 (1), 12–23.

[195] Yildizdan, G. and Baykan, Ö., 2020. A novel modified bat algorithm hybridizing by differential evolution algorithm. *Expert Systems with Applications*, 141, p.112949.

[196] Yue, X. and Zhang, H., 2020. Modified hybrid bat algorithm with genetic crossover operation and smart inertia weight for multilevel image segmentation. *Applied Soft Computing*, 90, p.106157.

[197] Yang, X., Gandomi, A.H., 2012. Bat algorithm: a novel approach for global engineering

optimization. Eng. Comput. 29 (5), 464–483.

[198] Iztok Fister Jr., Simon Fong, Janez Brest, I.F., 2014. A Novel Hybrid Self-Adaptive Bat Algorithm. The Scientific World Journal, 2014 (Article ID 709738), pp. 12.

[199] Fister, I., Fister, D., Yang, X.S., 2013. A hybrid bat algorithm. Elektrotehniski Vestnik/ Electrotechnical Rev. 80 (1–2), 1–7.

[200] Lihong Guo, Gai-Ge Wang, Heqi Wang, D.W., 2013. An Novel Hybrid Bat Algorithm with Harmony Search for Global Numerical Optimization. The Scientific World Journal, 2013 (Article ID 125625), pp. 9.

[201] Sathya, M.R., Ansari, M.M.T., 2015. Electrical Power and Energy Systems Load frequency control using Bat inspired algorithm based dual mode gain scheduling of PI controllers for interconnected power system. Int. J. Electr. Power Energy Syst. 64, 365–374.

[202] Naderi, M., Khamehchi, E., 2017. Application of DOE and metaheuristic bat algorithm for well placement and individual well controls optimization. J. Nat. Gas Sci. Eng.

[203] Gandomi, A. and Yang, X., 2014. Chaotic bat algorithm. *Journal of Computational Science*, 5(2), pp.224-232.

[204] Yılmaz, S., Kucuksille, E. and Cengiz, Y., 2014. Modified Bat Algorithm. *Electronics and Electrical Engineering*, 20(2).

[205] Ustun, D., Carbas, S. and Toktas, A., 2020. Multi-objective Optimization of Engineering Design Problems Through Pareto-Based Bat Algorithm. *Springer Tracts in Nature-Inspired Computing*, pp.19-43.

[206] Lu, Q., Ren, Y., Jin, H., Meng, L., Li, L., Zhang, C. and Sutherland, J., 2020. A hybrid metaheuristic algorithm for a profit-oriented and energy-efficient disassembly sequencing problem. *Robotics and Computer-Integrated Manufacturing*, 61, p.101828.

[207] Kyroudi, A., Petersson, K., Ozsahin, E., Bourhis, J., Bochud, F. and Moeckli, R., 2020. Exploration of clinical preferences in treatment planning of radiotherapy for prostate cancer using Pareto fronts and clinical grading analysis. *Physics and Imaging in Radiation Oncology*, 14, pp.82-86.

[208] Liang, J., Liu, Y. and Xue, Y., 2020. Preference-driven Pareto front exploitation for bloat control in genetic programming. *Applied Soft Computing*, 92, p.106254.

[209] Grygar, D. and Fabricius, R., 2019. An Efficient Adjustment of Genetic Algorithm for Pareto Front Determination. *Transportation Research Procedia*, 40, pp.1335-1342.

[210] Omid Bozorg-Haddad, 2018. Advanced Optimization by Nature-Inspired Algorithms. *Studies in Computational Intelligence*

[211] Darwish, A., 2018. Bio-inspired computing: Algorithms review, deep analysis, and the scope of applications. *Future Computing and Informatics Journal*, 3(2), pp.231-246.

[212] Brabazon, A., O'Neill, M. and McGarraghy, S., 2015. Chemically Inspired Algorithms. *Natural Computing Algorithms*, pp.479-498.

[213] Demirol, D., Oztemiz, F. and Karci, A., 2018. Performance Comparison of Physics Based Meta-Heuristic Optimization Algorithms. *2018 International Conference on Artificial Intelligence and Data Processing (IDAP),*.

[214] Kaipa, K. and Ghose, D., 2017. Glowworm Swarm Optimization. *Studies in Computational Intelligence,*.

[215] Cuevas, E., Fausto, F. and González, A., 2019. Metaheuristics and Swarm Methods: A Discussion on Their Performance and Applications. *Intelligent Systems Reference Library*, pp.43-67.

[216] Gowri, R. and Rathipriya, R., 2018. Non-Swarm Plant Intelligence Algorithm: BladderWorts Suction (BWS) Algorithm. *2018 International Conference on Circuits and Systems in Digital Enterprise Technology (ICCSDET),*.

[217] Gholizadeh, S., Danesh, M. and Gheyratmand, C., 2020. A new Newton metaheuristic algorithm for discrete performance-based design optimization of steel moment frames. *Computers & Structures*, 234, p.106250.

[218] Salih, S., Alsewari, A., Al-Khateeb, B. and Zolkipli, M., 2018. Novel Multi-swarm Approach for Balancing Exploration and Exploitation in Particle Swarm Optimization. *Advances in Intelligent Systems and Computing*, pp.196-206.

[219] Shadravan, S., Naji, H. and Bardsiri, V., 2019. The Sailfish Optimizer: A novel nature-inspired metaheuristic algorithm for solving constrained engineering optimization problems. *Engineering Applications of Artificial Intelligence*, 80, pp.20-34.

[220] Mirjalili, S. and Lewis, A., 2016. The Whale Optimization Algorithm. *Advances in Engineering Software*, 95, pp.51-67.

[221] Au, W., 1993. Comparison Between the Sonar of Bats and Dolphins. *The Sonar of Dolphins*, pp.242-265.

[222] Simmons, J., Houser, D. and Kloepper, L., 2014. Localization and Classification of Targets by Echolocating Bats and Dolphins. *Biosonar*, pp.169-193.

[223] Yovel, Y., Geva-Sagiv, M. and Ulanovsky, N., 2011. Click-based echolocation in bats: not so primitive after all. *Journal of Comparative Physiology A*, 197(5), pp.515-530.

[224] Mohl, B. and Surlykke, A., 1989. Detection of sonar signals in the presence of pulses of masking noise by the echolocating bat,Eptesicus fuscus. *Journal of Comparative Physiology A*, 165(1), pp.119-124.

[225] Kaveh, A., 2016. Dolphin Echolocation Optimization. *Advances in Metaheuristic Algorithms for Optimal Design of Structures*, pp.161-197.

[226] Griffin DR. Listening in the dark: the acoustic orientation of bats and men. New Haven (CT), Cambridge (MA): Yale University Press, Biological Laboratories, Harvard University; 1958. p. 413.

[227] Au, W., 1993. The Sonar of Dolphins. Springer-Verlag New York.

[228] AU, W., 1997. Echolocation in dolphins with a dolphin-bat comparison. *Bioacoustics*, 8(1-2), pp.137-162.

[229] Moss, C., Chiu, C. and Moore, P., 2014. Analysis of Natural Scenes by Echolocation in Bats and Dolphins. *Biosonar*, pp.231-256.

[230] Au, W. and Hastings, M., 2008. Echolocation in Marine Mammals. *Principles of Marine Bioacoustics*, pp.501-564.

[231] Kim, Hyun, 2010. Bat Intelligent Hunting Optimization with Application to Multiprocessor Scheduling. Electronic Thesis or Dissertation. Case Western Reserve University, *OhioLINK Electronic Theses and Dissertations Center*.

[232] Bora, T.C., Coelho, L.D.S., Lebensztajn, L., 2012. Bat-inspired optimization approach for the brushless DC wheel motor problem. IEEE Trans. Magn. 48 (2), 947–950.

[233] Biswal, S., A. K. Barisal, A. Behera, and T. Prakash. 2013. Optimal power dispatch using bat algorithm. InInternational conference on energy efficient technologies for sustainability, 1018–1023. ICEETS/IEEE.

[234] Niknam,T., R. A. Abarghooee, M. Zare, and B. B. Firouzi. 2013. Reserve constrained dynamicenvironmental/economic dispatch: A new multiobjective self-adaptive learning bat algorithm.IEEESystems Journal7:763–775.

[235] Kumaravel, G., and C. Kumar. 2012. Design of self tuning PI controller for STATCOM using Batsecholocation algorithm based neural Controller. InInternational conference on advances in engineering,science and management (ICAESM), 276–281. IEEE.

[236] Othman, Z., Sulaiman, S., Musirin, I., Omar, A. and Shaari, S., 2018. Dolphin Echolocation — Based sizing algorithm for stand-alone photovoltaic system. *2018 IEEE International Conference on Applied System Invention (ICASI),*.

[237] *Lenin, Kanagasabai et al., 2014. Dolphin Echolocation Algorithm for Solving Optimal Reactive Power Dispatch Problem.*

[238] Arjmand, M., Sheikhi Azqandi, M. and Delavar, M., 2015. Hybrid Improved Dolphin Echolocation and Ant Colony Optimization for Optimal Discrete Sizing of Truss Structures. *Journal of Rehabilitation in Civil Engineering*.

[239] Mahesh, N. and Vijayachitra, S., 2018. DECSA: hybrid dolphin echolocation and crow search optimization for cluster-based energy-aware routing in WSN. *Neural Computing and Applications*, 31(S1), pp.47-62.

Appendix

Table 6. 5: Schwefel 2.20 (Unimodal) Function

	$n = 10$	$n = 30$	$n = 100$	$n = 500$	$n = 1000$	*Total Rank*	*Final Rank*
HBD							
Best	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**		
Mean	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**		
Std	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**		
Rank	1	1	1	1	1	5	1
DEO							
Best	1.4502e+01	1.0153e+02	4.2116e+02	8.3222e+02	1.2569e+03		
Mean	1.5281e+01	1.0618e+02	4.2137e+02	8.3408e+02	1.2624e+03		
Std	1.6193e+00	1.6772e+00	5.1355e-01	2.3761e+00	9.6632e+00		
Rank	6	6	6	6	6	30	6
Bat							
Best	4.0582e+00	9.8142e+00	4.1868e+01	2.3526e+02	6.1088e+02		
Mean	4.4255e+00	1.0247e+01	4.2848e+01	2.3622e+02	6.1122e+02		
Std	5.6236e-01	5.6488e-01	8.0142e-01	6.5698e-01	6.4279e-01		
Rank	5	4	4	4	5	22	4
GWO							
Best	1.1010e+00	1.8811e+00	3.7406e+00	3.8701e+01	1.1055e+02		
Mean	1.8754e+00	3.7644e+00	1.1435e+01	8.0186e+01	1.8854e+02		
Std	7.9568e-01	3.6381e+00	9.1353e+00	4.0617e+01	7.8094e+01		

Rank	3	2	2	2	3	12	2
WOA							
Best	1.3668e+00	4.9866e+00	1.1800e+01	8.4798e+01	1.1003e+02		
Mean	1.8636e+00	6.2258e+00	1.6009e+01	8.7426e+01	1.4036e+02		
Std	6.7660e-01	2.3594e+00	6.7086e+00	8.5499e+00	4.3902e+01		
Rank	2	3	3	3	2	13	3
PSO							
Best	2.2656e+00	2.2951e+01	5.1323e+01	2.8721e+02	5.8160e+02		
Mean	2.6368e+00	2.5054e+01	5.1588e+01	2.8749e+02	5.8319e+02		
Std	4.2723e-01	1.4886e+00	5.1588e+01	8.6113e-01	4.8746e+00		
Rank	4	5	5	5	4	23	5

Table 6. 6: Sphere (Unimodal) Function

	$n = 10$	$n = 30$	$n = 100$	$n = 500$	$n = 1000$	***Total Rank***	***Final rank***
HBD							
Best	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**		
Mean	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**		
Std	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**		
Rank	2.5	2.5	2.5	1	1	9.5	1
DEO							
Best	1.3818e+00	1.8589e+02	6.0469e+02	1.4330e+03	8.2363e+03		
Mean	1.4261e+00	1.8589e+02	6.0469e+02	1.4460e+03	8.2515e+03		
Std	1.4333e-01	5.8320e-14	1.7626e+01	2.4000e+01	2.7036e+01		

Rank	5	5	5	5	6	26	5
Bat							
Best	1.6072e+00	5.5992e+02	6.7557e+02	3.5755e+04	8.0325e+03		
Mean	1.6072e+00	6.0272e+02	6.9004e+02	3.5755e+04	8.0325e+03		
Std	2.6780e-15	6.7076e+01	6.2325e+00	4.8800e+02	8.9556e+01		
Rank	6	6	6	6	5	29	6
GWO							
Best	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	6.5518e-01	1.3949e+00		
Mean	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	8.5689e-01	1.6634e+00		
Std	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	4.3039e-01	1.6634e+00		
Rank	2.5	2.5	2.5	2	2	11.5	2
WOA							
Best	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	1.6562e+02	1.2729e+00		
Mean	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	1.7916e+02	1.8411e+00		
Std	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	1.3820e+01	1.0830e+00		
Rank	2.5	2.5	2.5	4	3	14.5	3.5
PSO							
Best	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	2.2446e+00	4.4851e+00		
Mean	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	2.2446e+00	4.4851e+00		
Std	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	0.0000e+00	0.0000e+00		
Rank	2.5	2.5	2.5	3	4	14.5	3.5

Table 6. 7: Sum Square (Unimodal) Function

	$n = 10$	$n = 30$	$n = 100$	$n = 500$	$n = 1000$	*Total Rank*	*Final Rank*
HBD							
Best	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**		
Mean	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**		
Std	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**		
Rank	2.5	1.5	1	1	1	7	1
DEO							
Best	1.0822e+01	6.0138e+03	1.2556e+04	3.9207e+04	4.0087e+04		
Mean	1.2307e+01	6.5006e+03	1.2984e+04	3.9448e+04	4.0137e+04		
Std	2.3630e+00	1.1458e+02	7.9254e+02	2.7274e+02	1.5575e+02		
Rank	5	6	6	6	6	29	6
Bat							
Best	1.0285e+01	4.2283e+01	1.3922e+03	3.7776e+04	3.8200e+04		
Mean	1.4866e+01	4.4912e+01	1.3965e+03	3.7776e+04	3.8200e+04		
Std	1.8878e+00	5.6030e+00	4.4743e+00	7.4650e-12	7.4650e-12		
Rank	6	4	4	5	5	24	5
GWO							
Best	**0.0000e+00**	1.0771e+00	6.4480e+01	1.2249e+02	1.6123e+02		
Mean	**0.0000e+00**	5.8101e+00	8.1445e+01	1.7018e+02	4.1973e+02		
Std	**0.0000e+00**	8.5710e+00	1.9307e+01	5.9608e+01	2.9999e+02		
Rank	2.5	3	3	3	2	13.5	3
WOA							

Best	**0.0000e+00**	**0.0000e+00**	2.5974e+00	2.9584e+01	1.1084e+03		
Mean	**0.0000e+00**	**0.0000e+00**	5.3803e+00	7.9738e+01	1.3134e+03		
Std	**0.0000e+00**	**0.0000e+00**	4.9763e+00	1.1067e+02	1.3134e+03		
Rank	2.5	1.5	2	2	3	11	2
PSO							
Best	**0.0000e+00**	5.5657e+01	3.2872e+03	3.4276e+04	3.6811e+04		
Mean	**0.0000e+00**	5.6597e+01	3.3169e+03	3.4276e+04	3.6811e+04		
Std	**0.0000e+00**	4.2030e+00	9.1454e+01	7.4650e-12	7.4650e-12		
Rank	2.5	5	5	4	4	20.5	4

Table 6. 8: Zakharov (Unimodal) Function

	$n = 10$	$n = 30$	$n = 100$	$n = 500$	$n = 1000$	***Total Rank***	***Final Rank***
HBD							
Best	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**		
Mean	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**		
Std	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**		
Rank	1.5	1	1	1	1	5.5	1
DEO							
Best	1.2615e+01	3.7556e+02	4.1898e+01	1.6336e+02	3.2992e+02		
Mean	1.3018e+01	8.0047e+02	4.6944e+01	1.6580e+02	3.3444e+02		
Std	1.8032e+00	1.3078e+03	2.2564e+01	2.2629e+00	5.8575e+00		
Rank	5	6	4	4	4	23	4
Bat							

Best	8.5003e+01	3.8137e+01	7.2866e+02	4.4884e+03	2.5958e+03		
Mean	8.7679e+01	3.9040e+01	7.2910e+02	6.7167e+04	2.7143e+03		
Std	5.5648e+00	6.0628e-01	1.6825e+00	7.5923e+04	7.0369e+01		
Rank	6	4	6	6	5	27	6
GWO							
Best	6.6657e-01	1.2145e+00	2.3611e+00	1.4771e+01	3.4559e+01		
Mean	1.1096e+00	2.2557e+00	3.9488e+00	4.0443e+01	7.5536e+01		
Std	8.8551e-01	1.5967e+00	2.4622e+00	6.0426e+00	1.8092e+01		
Rank	3	3	2	2	3	13	3
WOA							
Best	**0.0000e+00**	1.0949e+00	8.3388e+00	4.2513e+01	2.3264e+01		
Mean	**0.0000e+00**	1.2648e+00	1.0802e+01	5.8707e+01	3.3578e+01		
Std	**0.0000e+00**	5.2311e-01	3.2681e+00	1.5028e+01	2.5190e+01		
Rank	1.5	2	3	3	2	11.5	2
PSO							
Best	6.5138e+00	4.2671e+01	9.9343e+01	2.8985e+02	2.6249e+03		
Mean	6.7738e+00	4.8479e+01	1.3652e+02	3.2349e+03	5.6788e+04		
Std	5.3228e-01	2.5978e+01	1.6625e+02	9.0645e+03	1.2621e+05		
Rank	4	5	5	5	6	25	5

Table 6. 9: Elliptic (Unimodal) Function

	$n = 10$	$n = 30$	$n = 100$	$n = 500$	$n = 100$	*Total Rank*	*Final Rank*
HBD							
Best	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**		
Mean	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**		
Std	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**		
Rank	1.5	1	1	1	1	5.5	1
DEO							
Best	6.2742e+00	3.8263e+02	2.4299e+02	2.6488e+02	5.6487e+02		
Mean	6.5947e+00	3.9926e+02	2.8767e+02	2.8672e+02	5.6581e+02		
Std	5.1253e-01	2.6070e+01	1.9256e+01	9.4684e+00	2.8881e+00		
Rank	5	6	5	4	4	24	5
Bat							
Best	4.5690e+01	2.7239e+02	8.4498e+02	4.1755e+03	1.3541e+03		
Mean	4.5690e+01	2.7239e+02	8.4498e+02	4.1755e+03	1.3541e+03		
Std	7.2900e-15	5.8320e-14	0.0000e+00	0.0000e+00	2.3328e-13		
Rank	6	5	6	6	5	28	6
GWO							
Best	**0.0000e+00**	7.2487e-01	1.2872e+00	1.3401e+01	2.1851e+01		
Mean	**0.0000e+00**	1.0929e+00	1.9850e+00	1.4974e+01	2.6601e+01		
Std	**0.0000e+00**	6.3042e-01	1.6621e+00	6.4829e+00	8.3313e+00		
Rank	1.5	2	2	2	2	9.5	2
WOA							

Best	1.0099e+00	6.0873e+00	7.2294e+00	5.7419e+01	1.5059e+02		
Mean	1.3336e+00	7.0595e+00	7.9467e+00	5.9292e+01	1.9915e+02		
Std	5.0732e-01	2.2424e+00	9.0321e-01	5.7646e+00	8.6297e+01		
Rank	3	3	3	3	3	15	3
PSO							
Best	3.6677e+00	1.8080e+01	6.3021e+01	1.2191e+03	1.6861e+03		
Mean	3.9991e+00	1.8242e+01	6.3021e+01	1.2555e+03	1.7434e+03		
Std	7.9024e-01	5.0024e-01	0.0000e+00	1.1200e+02	1.7628e+02		
Rank	4	4	4	5	6	23	4

Table 6. 10: Power (Unimodal) Function

	$n = 10$	$n = 30$	$n = 100$	$n = 500$	$n = 1000$	***Total Rank***	***Final Rank***
HBD							
Best	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**		
Mean	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**		
Std	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**		
Rank	3.5	3	2	1	1	10.5	1
DEO							
Best	**0.0000e+00**	1.9230e+00	4.9128e+00	3.2789e+01	1.2206e+02		
Mean	**0.0000e+00**	1.9230e+00	4.9128e+00	3.2976e+01	1.2696e+02		
Std	**0.0000e+00**	2.2781e-16	9.1125e-16	5.7616e-01	3.3956e+00		
Rank	3.5	6	6	5	6	26.5	6
Bat							

Best	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	3.3515e+01	7.4519e+01		
Mean	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	3.3515e+01	7.4519e+01		
Std	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	7.2900e-15	1.4580e-14		
Rank	3.5	3	2	6	5	19.5	5
GWO							
Best	**0.0000e+00**	**0.0000e+00**	7.1533e-01	4.2645e-02	5.1727e+00		
Mean	**0.0000e+00**	**0.0000e+00**	1.6075e+00	7.3714e+00	3.0888e+01		
Std	**0.0000e+00**	**0.0000e+00**	1.7632e+00	1.1570e+01	3.1604e+01		
Rank	3.5	3	5	2	3	16.5	3.5
WOA							
Best	**0.0000e+00**	**0.0000e+00**	4.0392e-02	3.7556e+00	5.9533e-01		
Mean	**0.0000e+00**	**0.0000e+00**	9.2445e-01	1.0519e+01	7.9151e+00		
Std	**0.0000e+00**	**0.0000e+00**	2.1363e+00	1.2018e+01	2.1569e+01		
Rank	3.5	3	4	3	2	15.5	2
PSO							
Best	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	3.2340e+01	7.3263e+01		
Mean	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	3.2340e+01	7.3263e+01		
Std	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	0.0000e+00	1.4580e-14		
Rank	3.5	3	2	4	4	16.5	3.5

Table 6. 11: Quartic (Unimodal) Function

	$n = 10$	$n = 30$	$n = 100$	$n = 500$	$n = 1000$	*Total Rank*	*Final Rank*
HBD							
Best	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**		
Mean	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**		
Std	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**		
Rank	2	1.5	1.5	1.5	1	7.5	1
DEO							
Best	5.2668e+00	1.0966e+01	3.6412e+01	3.4013e+01	1.4269e+02		
Mean	6.7014e+00	1.0966e+01	4.2476e+01	3.4817e+01	1.4483e+02		
Std	2.0057e+00	3.6450e-15	1.9344e+00	6.1973e-01	1.5328e+00		
Rank	5	4	4	3	3	19	4
Bat							
Best	4.9819e+00	7.1722e+01	1.7716e+03	1.8280e+02	6.8221e+02		
Mean	7.6560e+00	7.5375e+01	1.7773e+03	1.8280e+02	6.8221e+02		
Std	6.2941e-01	1.1245e+01	1.7344e+01	5.8320e-14	3.4992e-13		
Rank	6	5	6	4	6	27	6
GWO							
Best	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	1.8659e+00		
Mean	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	2.6130e+00		
Std	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	1.2461e+00		
Rank	2	1.5	1.5	1.5	2	8.5	2
WOA							

Best	**0.0000e+00**	2.2074e-01	4.7551e+00	1.3429e+02	8.4767e+01		
Mean	**0.0000e+00**	9.7479e-01	1.5504e+01	5.5660e+02	1.5403e+02		
Std	**0.0000e+00**	1.5571e+00	1.9331e+01	5.9276e+02	1.2308e+02		
Rank	2	3	3	5	4	17	3
PSO							
Best	1.2741e+00	7.7403e+01	4.2080e+02	1.3218e+03	3.7434e+02		
Mean	1.6330e+00	7.9249e+01	4.2316e+02	1.3218e+03	3.7434e+02		
Std	1.6050e+00	8.2530e+00	7.2623e+00	0.0000e+00	5.8320e-14		
Rank	4	6	5	6	5	26	5

Table 6. 12: Schwefel 2.22 (Unimodal) Function

	$n = 10$	$n = 30$	$n = 100$	$n = 500$	$n = 1000$	***Total Rank***	***Final Rank***
HBD							
Best	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**		
Mean	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**		
Std	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**		
Rank	1	1	1	1	1	5	1
DEO							
Best	5.1157e+00	1.4576e+01	4.4305e+01	1.1658e+02	1.8582e+02		
Mean	5.5501e+00	1.4576e+01	4.4773e+01	1.1739e+02	1.8678e+02		
Std	5.4579e-01	9.1125e-15	6.8723e-01	1.1289e+00	8.8798e-01		
Rank	6	5	4	4	4	23	4.5
Bat							

Best	3.2532e+00	1.3819e+01	5.4180e+01	3.0409e+02	6.3577e+02		
Mean	3.5147e+00	1.3819e+01	5.4180e+01	3.0409e+02	6.3577e+02		
Std	6.1547e-02	3.6450e-15	7.2900e-15	5.8320e-14	3.4992e-13		
Rank	4	4	5	5	5	23	4.5
GWO							
Best	7.0457e-01	3.0115e+00	1.6732e+01	2.7531e+01	7.3676e+01		
Mean	2.7981e+00	4.7072e+00	2.1040e+01	3.7775e+01	1.0108e+02		
Std	2.2234e+00	2.5544e+00	6.5052e+00	2.0313e+01	3.0484e+01		
Rank	3	2	2	2	2	12	2
WOA							
Best	1.1395e+00	5.9077e+00	1.4975e+01	8.1781e+01	1.0057e+02		
Mean	1.8616e+00	6.6394e+00	1.8696e+01	9.1483e+01	1.2709e+02		
Std	1.0057e+00	1.7875e+00	7.4614e+00	1.2010e+01	3.9494e+01		
Rank	2	3	2	3	3	13	3
PSO							
Best	3.7744e+00	2.0481e+01	7.9690e+01	4.4834e+02	9.0195e+02		
Mean	4.3051e+00	2.1307e+01	8.0640e+01	4.5054e+02	9.0850e+02		
Std	6.8288e-01	1.3369e+00	2.9239e+00	6.7790e+00	2.0146e+01		
Rank	5	6	6	6	6	29	6

Table 6. 13: Ackley (Multimodal) Function

	$n = 10$	$n = 30$	$n = 100$	$n = 500$	$n = 1000$	*Total Rank*	*Final Rank*
HBD							
Best	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	1.8948e+01	1.7727e+01		
Mean	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	1.9704e+01	2.1546e+01		
Std	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	3.1848e+01	4.6889e+01		
Rank	1	1	1	3	4	10	1
DEO							
Best	4.6611e+00	1.2232e+01	1.7506e+01	1.7725e+01	1.9013e+01		
Mean	7.1403e+00	1.3103e+01	1.7898e+01	1.9399e+01	2.1023e+01		
Std	2.3762e+00	1.3754e+00	1.1064e+00	5.9558e-01	4.1066e+00		
Rank	2	2	5	2	3	14	2
Bat							
Best	1.2898e+01	1.9314e+01	1.5040e+01	2.1105e+01	2.2774e+01		
Mean	1.4398e+01	1.9642e+01	1.5561e+01	2.5220e+01	2.5188e+01		
Std	1.1018e+00	5.8331e-01	7.2986e-01	2.2478e+00	1.8984e+00		
Rank	4	4	2	6	5	21	5
GWO							
Best	1.6600e+01	1.9228e+01	1.4602e+01	1.9206e+01	1.9916e+01		
Mean	1.8078e+01	1.9970e+01	1.5880e+01	1.9206e+01	2.0076e+01		
Std	1.7602e+00	5.6164e-01	2.1800e+00	4.0699e-01	1.3216e-01		
Rank	6	6	3	1	2	18	3
WOA							
Best	1.3181e+01	1.9264e+01	1.6513e+01	2.0141e+01	1.8058e+01		

Mean	1.5148e+01	1.9770e+01	1.7044e+01	2.0141e+01	1.8936e+01		
Std	2.9518e+00	6.0863e-01	1.0735e+00	2.0141e+01	3.6498e-01		
Rank	5	5	4	5	1	20	4
PSO							
Best	1.1794e+01	1.6411e+01	1.8699e+01	1.9325e+01	2.4654e+01		
Mean	1.1917e+01	1.7363e+01	1.9189e+01	1.9748e+01	2.6296e+01		
Std	5.4997e-01	9.7662e-01	8.7354e-01	4.6348e-01	1.9147e+00		
Rank	3	3	6	4	6	22	6

Table 6. 14: Griewank (Multimodal) Function

	$n = 10$	$n = 30$	$n = 100$	$n = 500$	$n = 1000$	***Total Rank***	***Final Rank***
HBD							
Best	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**		
Mean	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**		
Std	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**		
Rank	3.5	3.5	2.5	2	2	13.5	1
DEO							
Best	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	5.1335e+00	9.4908e+00		
Mean	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	6.9099e+00	1.0370e+01		
Std	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	1.9746e+00	1.3622e+00		
Rank	3.5	3.5	2.5	5	5	19.5	4
Bat							
Best	**0.0000e+00**	**0.0000e+00**	3.0919e+01	4.1262e+01	1.9983e+02		
Mean	**0.0000e+00**	**0.0000e+00**	3.1630e+01	4.1262e+01	2.0191e+02		

Std	**0.0000e+00**	**0.0000e+00**	8.2005e-01	6.7856e-01	5.5271e+00		
Rank	3.5	3.5	6	6	6	25	6
GWO							
Best	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**		
Mean	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**		
Std	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**		
Rank	3.5	3.5	2.5	2	2	13.5	2
WOA							
Best	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**		
Mean	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**		
Std	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**		
Rank	3.5	3.5	2.5	2	2	13.5	2
PSO							
Best	**0.0000e+00**	**0.0000e+00**	6.1880e-01	3.9488e+00	7.0197e+00		
Mean	**0.0000e+00**	**0.0000e+00**	6.2105e-01	3.9488e+00	7.7770e+00		
Std	**0.0000e+00**	**0.0000e+00**	5.4909e-03	8.9760e-01	1.6711e+00		
Rank	3.5	3.5	5	4	4	20	5

Table 6. 15: Holzman 2 (Multimodal) Function

	$n = 10$	$n = 30$	$n = 100$	$n = 500$	$n = 1000$	***Total Rank***	***Final Rank***
HBD							
Best	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**		
Mean	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**		
Std	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**		

Rank	3	1.5	1	1	1	7.5	1
DEO							
Best	**0.0000e+00**	7.9041e+00	1.6591e+01	9.5795e+01	2.1101e+03		
Mean	**0.0000e+00**	7.9041e+00	1.7216e+01	9.5795e+01	2.1335e+03		
Std	**0.0000e+00**	1.8225e-15	9.8410e-01	2.7779e+00	1.4436e+01		
Rank	3	6	5	4	6	24	5
Bat							
Best	1.0664e+00	5.5395e+00	1.7145e+01	9.8904e+01	1.9850e+02		
Mean	1.0664e+00	5.5395e+00	1.7145e+01	9.8904e+01	1.9850e+02		
Std	2.2781e-16	2.7338e-15	7.2900e-15	2.9160e-14	5.8320e-14		
Rank	6	5	4	6	5	26	6
GWO							
Best	**0.0000e+00**	**0.0000e+00**	6.9494e-01	9.3845e+00	2.5341e+01		
Mean	**0.0000e+00**	**0.0000e+00**	6.8443e+00	1.5161e+01	3.2339e+01		
Std	**0.0000e+00**	**0.0000e+00**	7.4470e+00	3.3659e+00	1.0627e+01		
Rank	3	1.5	2	2	2	10.5	2
WOA							
Best	**0.0000e+00**	2.2007e-01	2.3935e+00	6.7464e+01	1.0178e+02		
Mean	**0.0000e+00**	1.8104e+00	7.0434e+00	7.9656e+01	1.0178e+02		
Std	**0.0000e+00**	2.8564e+00	7.2873e+00	1.3778e+01	3.9327e+01		
Rank	3	3	3	3	3	15	3
PSO							
Best	**0.0000e+00**	5.3554e+00	1.9813e+01	9.6978e+01	1.9637e+02		
Mean	**0.0000e+00**	5.3554e+00	1.9813e+01	9.6978e+01	1.9637e+02		
Std	**0.0000e+00**	1.8225e-15	3.6450e-15	4.3740e-14	8.7480e-14		

Rank	3	4	6	5	4	22	4

Table 6. 16: Hyper Ellipsoid (Multimodal) Function

	$n = 10$	$n = 30$	$n = 100$	$n = 500$	$n = 1000$	***Total Rank***	***Final Rank***
HBD							
Best	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**		
Mean	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**		
Std	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**		
Rank	1	1	1	1	1	5	1
DEO							
Best	1.4521e+01	2.7466e+01	2.1244e+01	2.7027e+01	7.3912e+01		
Mean	2.5119e+01	2.7466e+01	2.2805e+01	3.0134e+01	9.8227e+01		
Std	4.3705e+00	7.2900e-15	2.8844e+00	1.8405e+00	1.9075e+01		
Rank	5	3	3	2	2	15	2.5
Bat							
Best	4.0885e+01	3.9281e+01	1.0581e+02	1.3046e+02	4.1661e+02		
Mean	4.1300e+01	3.9281e+01	1.0581e+02	1.3046e+02	4.1661e+02		
Std	5.2188e-01	7.2900e-15	2.9160e-14	2.9160e-14	1.1664e-13		
Rank	6	5	5	3	4	23	5
GWO							
Best	1.0128e+00	2.4041e+00	1.6030e+01	7.7147e+01	9.6209e+01		
Mean	9.0460e+00	8.7784e+01	1.2209e+02	3.6733e+02	7.1516e+02		
Std	1.1365e+01	2.1685e+02	2.2020e+02	7.7087e+02	1.9631e+03		
Rank	3	6	6	5	6	26	6

WOA							
Best	1.4765e+00	6.6727e+00	1.0511e+01	6.0385e+01	2.1388e+02		
Mean	2.9242e+00	2.6525e+01	1.0511e+01	5.7573e+02	2.5463e+02		
Std	4.4343e+00	3.3087e+01	3.5277e+02	5.7573e+02	8.4263e+01		
Rank	2	2	2	6	3	15	2.5
PSO							
Best	9.3032e+00	3.5658e+01	7.5172e+01	1.5599e+02	4.5090e+02		
Mean	9.3032e+00	3.5658e+01	7.5172e+01	1.5599e+02	4.5090e+02		
Std	9.3032e+00	7.2900e-15	1.4580e-14	2.9160e-14	0.0000e+00		
Rank	4	4	4	4	5	21	4

Table 6. 17: Csendes (Multimodal) Function

	$n = 10$	$n = 30$	$n = 100$	$n = 500$	$n = 1000$	***Total Rank***	***Final Rank***
HBD							
Best	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**		
Mean	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**		
Std	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**		
Rank	3.5	2	1	1	1	8.5	1
DEO							
Best	**0.0000e+00**	4.5826e+00	1.9670e+01	1.2032e+02	2.4178e+02		
Mean	**0.0000e+00**	4.5826e+00	1.9670e+01	1.2456e+02	2.4500e+02		
Std	**0.0000e+00**	9.1125e-16	7.2900e-15	4.0436e+00	8.5933e+00		
Rank	3.5	6	6	5	4	24.5	5

Bat							
Best	**0.0000e+00**	3.4314e+00	8.2447e+00	1.1330e+02	2.4634e+02		
Mean	**0.0000e+00**	3.6649e+00	1.1483e+01	1.1399e+02	2.4634e+02		
Std	**0.0000e+00**	2.0792e-01	4.5274e+00	7.0414e-01	2.9160e-14		
Rank	3.5	4	3	4	5	19.5	4
GWO							
Best	**0.0000e+00**	**0.0000e+00**	1.4087e+00	8.4378e+00	9.7667e+00		
Mean	**0.0000e+00**	**0.0000e+00**	5.7182e+00	2.9452e+01	4.0140e+01		
Std	**0.0000e+00**	**0.0000e+00**	8.3490e+00	3.9623e+01	7.2687e+01		
Rank	3.5	2	2	2	2	11.5	2
WOA							
Best	**0.0000e+00**	**0.0000e+00**	1.6102e+01	2.2509e+01	4.9638e+01		
Mean	**0.0000e+00**	**0.0000e+00**	1.6102e+01	5.0826e+01	1.0421e+02		
Std	**0.0000e+00**	**0.0000e+00**	7.2266e-01	5.0826e+01	9.6974e+01		
Rank	3.5	2	4	3	3	15.5	3
PSO							
Best	**0.0000e+00**	4.0775e+00	1.6467e+01	1.2758e+02	2.6050e+02		
Mean	**0.0000e+00**	4.0775e+00	1.6467e+01	1.2758e+02	2.6050e+02		
Std	**0.0000e+00**	9.1125e-16	0.0000e+00	1.4580e-14	0.0000e+00		
Rank	3.5	5	5	6	6	25.5	6

Table 6. 18: Rastrigin (Multimodal) Function

	$n = 10$	$n = 30$	$n = 100$	$n = 500$	$n = 1000$	*Total Rank*	*Final Rank*
HBD							
Best	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**		
Mean	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**		
Std	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**		
Rank	1	1	1	1	1	5	1
DEO							
Best	2.2400e+02	9.1178e+02	2.0528e+03	1.1245e+04	2.0083e+04		
Mean	2.2591e+02	9.1327e+02	2.0917e+03	1.1658e+04	2.0114e+04		
Std	2.1675e+00	4.5714e+00	6.8985e+01	1.6095e+03	1.0266e+02		
Rank	6	6	6	6	6	30	6
Bat							
Best	4.5606e+00	1.6274e+01	3.6254e+01	1.6699e+02	3.2838e+02		
Mean	1.9951e+01	5.2467e+01	1.2453e+02	6.2353e+02	5.9907e+02		
Std	2.0071e+01	8.8019e+01	1.1145e+02	9.6681e+02	1.2106e+03		
Rank	4	4	3	2	2	15	3
GWO							
Best	1.0023e+02	3.0648e+02	1.0000e+03	5.0313e+03	1.0701e+04		
Mean	1.0451e+02	3.0648e+02	1.0163e+03	6.1129e+03	1.2257e+04		
Std	7.8657e+00	6.8825e+01	6.7338e+01	1.8125e+03	3.1571e+03		
Rank	5	5	5	5	5	25	5
WOA							
Best	2.0132e+00	6.8061e+00	4.1214e+01	3.0809e+01	2.4393e+02		

Mean	3.1201e+00	1.6518e+01	2.4538e+02	7.8960e+02	1.8769e+03		
Std	4.9501e+00	2.0507e+01	2.8545e+02	2.4307e+03	2.6181e+03		
Rank	2	3	4	4	4	17	4
PSO							
Best	7.0407e+00	1.0879e+01	4.2093e+01	3.4880e+02	1.3784e+03		
Mean	9.2378e+00	1.4938e+01	5.3374e+01	6.2930e+02	1.8038e+03		
Std	9.2378e+00	1.8154e+01	5.0450e+01	6.3695e+02	6.6997e+02		
Rank	3	2	2	3	3	13	2

Table 6. 19: Cosine Mixture (Multimodal) Function

	$n = 10$	$n = 30$	$n = 100$	$n = 500$	$n = 1000$	***Total Rank***	***Final Rank***
HBD							
Best	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	1.2419e+00	6.5648e+01		
Mean	**0.0000e+00**	1.0054e+00	3.1500e+00	3.2065e+01	9.9826e+00		
Std	**0.0000e+00**	2.1151e+00	6.4854e+00	3.8619e+01	8.9327e+01		
Rank	3	1	1	1	1	7	1
DEO							
Best	1.4935e+00	7.8637e+00	2.8223e+01	1.5004e+02	3.0995e+02		
Mean	1.6437e+00	7.8637e+00	2.8223e+01	1.5395e+02	3.1226e+02		
Std	1.7033e-01	1.8225e-15	1.4580e-14	3.6451e+00	3.1282e+00		
Rank	6	6	6	5	6	29	6
Bat							
Best	**0.0000e+00**	5.0331e+00	2.6067e+01	1.5085e+02	2.0213e+02		
Mean	**0.0000e+00**	5.5002e+00	2.6810e+01	1.5122e+02	2.0775e+02		

Std	**0.0000e+00**	4.4445e-01	1.8257e-01	6.6435e-01	2.4489e+01		
Rank	3	4	3	3	4	17	3
GWO							
Best	**0.0000e+00**	2.1876e+00	1.4821e+00	3.6711e+01	8.9966e+01		
Mean	**0.0000e+00**	2.7727e+00	7.5920e+00	5.2575e+01	1.6331e+02		
Std	**0.0000e+00**	1.3506e+00	9.2225e+00	3.6366e+01	9.0490e+01		
Rank	3	2	2	2	3	12	2
WOA							
Best	**0.0000e+00**	4.8056e+00	2.4579e+01	1.5170e+02	1.0192e+02		
Mean	**0.0000e+00**	5.1003e+00	2.8201e+01	1.5755e+02	1.2932e+02		
Std	**0.0000e+00**	4.7747e-01	3.3628e+00	6.0077e+00	6.8305e+01		
Rank	3	3	5	6	2	19	4
PSO							
Best	**0.0000e+00**	5.5970e+00	2.7515e+01	1.5187e+02	3.0542e+02		
Mean	**0.0000e+00**	5.5970e+00	2.7515e+01	1.5187e+02	3.0542e+02		
Std	**0.0000e+00**	9.1125e-16	3.6450e-15	5.8320e-14	1.1664e-13		
Rank	3	5	4	4	5	21	5

Table 6. 20: Schwefel 2.26 (Multimodal) Function

	$n = 10$	$n = 30$	$n = 100$	$n = 500$	$n = 1000$	***Total Rank***	***Final Rank***
HBD							
Best	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00		
Mean	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00		
Std	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00		

Rank	2.5	1	1	1	1	6.5	1
DEO							
Best	3.9932e+00	3.2889e+01	1.2883e+02	8.1819e+02	9.9686e+02		
Mean	4.9415e+00	3.6367e+01	1.3035e+02	8.2340e+02	9.9786e+02		
Std	8.7109e-01	2.3361e+00	3.2878e+00	9.3682e+00	2.8670e+00		
Rank	6	6	5	6	6	29	6
Bat							
Best	3.2929e+00	1.1312e+01	1.3048e+02	3.1977e+02	8.2245e+02		
Mean	3.5400e+00	1.1679e+01	1.3659e+02	3.2440e+02	8.2880e+02		
Std	5.0779e-01	1.1284e+00	1.4388e+00	1.7259e+00	2.9903e+00		
Rank	5	4	6	4	5	24	5
GWO							
Best	0.0000e+00	7.9770e-01	3.4857e+00	9.7608e+00	4.0634e+01		
Mean	0.0000e+00	1.4726e+00	1.0907e+01	2.1480e+01	1.0935e+02		
Std	0.0000e+00	9.4046e-01	1.0162e+01	1.8619e+01	9.8839e+01		
Rank	2.5	2	2	2	2	10.5	2
WOA							
Best	0.0000e+00	3.0738e+00	1.9247e+01	2.6046e+01	1.6955e+02		
Mean	0.0000e+00	3.6453e+00	1.9698e+01	5.2189e+01	1.9794e+02		
Std	0.0000e+00	1.3015e+00	2.0167e+00	4.4561e+01	3.1661e+01		
Rank	2.5	3	3	3	3	14.5	3
PSO							
Best	0.0000e+00	1.5255e+01	7.3255e+01	3.6363e+02	7.5390e+02		
Mean	0.0000e+00	1.5468e+01	7.3645e+01	3.6987e+02	7.6182e+02		
Std	0.0000e+00	9.4985e-01	1.7432e+00	1.7205e+01	2.4372e+01		

Rank	2.5	5	4	5	4	20.5	4