EasyChair Preprint

№ 7428

Predicting Critical Genes from Genomic Data Using Artificial Gorilla Troops Optimizer

Rukhsun Ara Parvin, Biswajit Jana and Sriyankar Acharyya

February 7, 2022

1 Introduction

Bioinformatics [1] is the source of tools like databases and algorithms that solve many real-life problems. It is an interdisciplinary study that also uses computers to store and illustrate the biological data. The main purpose is to develop useful & efficient tools that work on biological data. It's all about engineering.

Computational biology [2] is the knowledge or study of biological, behavioural, and social systems via the use of data analysis and theoretical methodologies, mathematical modelling, and computational simulation tools. Molecular biology, genetics, genomics, evolutionary ecology, human anatomy, neurology, and computer science are all included in the broad definition of the topic. It also has roots in mathematics and computer science. In computational biology, biomarkers are important for disease detection. Genetical disorders play a vital role in the progression of various cancers and diseases. Hence, prediction of critical genes is important. The sequencing of the human genome, the creation of precise brain models, and the modelling of biological systems have all been made possible because of computational biology.

The term "biomarkers" [3] refers to the characteristics that can be objectively measured through biological methods. Genes can function as biomarkers. Mutation of critical genes is the cause of many diseases, like cancer. One challenging application for machine learning researchers is the identification of biomarkers or critical genes with high accuracy in large genomics data sets. The gene selection technique [4] plays a significant role in the identification of attractive biomarkers, i.e., critical genes. It is necessary to identify critical genes for the disease. It's very useful in precision medicine.

Discovering the optimal therapy for any disease by deciphering our genetic code is a key component of precision medicine. Precision medicine integrates genetics, biology, and medicine to provide personalised treatment strategies for each patient.

Here, a new population-based metaheuristic algorithm has been applied called the Gorilla Troops Optimizer [6]. It will be used to detect biomarkers, i.e., critical genes for a particular disease, with high classification accuracy. All the non-traditional methods of optimization [7] are called "meta-heuristics" [7]. Optimization means finding the best possible solutions. The Gorilla Troops Optimizer is a meta-heuristic approach based on a nature-inspired population-based algorithm. It is inspired by the social intelligence and behaviour of gorillas. In GTO, each individual is called a " Candidate Gorilla Position Vector" (solution). In Artificial Gorilla Troops, the optimizer search space comprises mainly three types of candidate solutions that are X, GX, and Silverback Gorilla candidate position vectors are generated by each phase that is exploration and exploitation of GTO. X is called the gorilla position vector. GX is called the candidate gorilla position vector. The difference is that GX belongs to another memory of GTO [6]. Silverback is the best solution. This algorithm is stochastic in nature.

In this work, the performance of GTO and MGTO [8] is tested on five selected benchmark functions. The GTO and MGTO are compared by their best objective value. Also, the running

time of the GTO algorithm on MATLAB and PYTHON code is compared on 25 selected benchmark functions. After comparison, the results prove that GTO is superior than other metaheuristics algorithms, but MGTO gives better results than GTO. Furthermore, PYTHON 3 is faster than MATLAB. And all benchmark test functions give the optimal or near optimal result.

The remaining section of the work is arranged as follows. Section 2 contains a literature survey of meta-heuristic algorithms. A problem definition has been presented in section 3. The proposed work of AGTO and MGTO have been discussed in section 4. The Basic AGTO and MGTO with proposed methodologies have been explained in section 5. Section 6 contains experimental results, and Section 7 contains the conclusion and future scope. The work has been concluded by section 8.

2. Literature Survey

In the machine learning research area, various categories of metaheuristic algorithms have been proposed in optimization technique. Most of these meta-heuristic algorithms are inspired by collective behaviours and hunting mechanisms and food searching behaviours of animals. The aim of this section is to explore research areas where different types of metaheuristic algorithms and GTO have been used and study this basic algorithm for solving optimization problems. The animal-based algorithms that use different meta-heuristic approaches have been illustrated below.

Some old (Holland et al. 1960s & 1970s) meta heuristic algorithms like GA (genetic Algorithm) [9] are evolutionary population-based algorithms. According to this, a population is a collection of individuals or chromosomes. Each of the chromosomes is called a candidate solution or an individual. The basic operators of GA are selection, crossover, and mutation. The best solution is considered as that solution (chromosome) which has the lowest fitness value for the minimization problem.

PSO (Particle swarm optimization 90s algorithm, Kennedy et al.1995) [9], which is inspired by the grouping behaviours of birds and fish (particles). PSO is a global optimization population-based algorithm. The best solution is being considered. Particles can move in a D-dimensional search space. And each particle depends on velocity and position. The position of each particle in search space is said to be the solution. The best solution, i.e., best position, is categorised into two parts, such as global best and personal best. "Personal best" is the best position of a particle considering its neighbours. whereas Global Best is the best position considering all particles.

Ants were the inspiration for ACO (Ant Colony Optimization [10], Macro Dorigo, 1992] [9]. The goal of this algorithm is to be useful for finding the shortest path. ACO simulates this behaviour of ants because in real life, ants communicate with each other to search for the best way to reach a destination by spreading pheromones. Each ant actually stores its own position in memory such that other ants can locate the best solution in future iterations. And this exploration will continue until the best root is found.

The WSA (Water Strider Algorithm, Kaveh and Dadras Eslamlou) [9] is a population-based meta heuristic algorithm inspired by the life cycle of water striders.

Another meta-heuristic approach algorithm named "Political Optimizer" [11] is a novel approach to human behaviour based on Askari et al. It is inspired by the poly-stages of politics. In this algorithm, each political party and each constituency are called a solution. It's a multipopulational-based algorithm. It consists of five phases, such as inter-party elections, election campaigns, party switching, election phases, and parliamentary affairs.

In this new era, except for these oldest techniques, a lot of optimization meta-heuristic algorithms with their enhancement have been developed in recent years. Anuj Kumar et al. (2016) proposed System Reliability Optimization [12] using the Gray Wolf Optimizer [12] Algorithm. This GWO algorithm can help to solve any problems that are non-linear and NP-hard in nature. This algorithm is based on grey wolves. It’s inspired by the hunting procedure of wolves.

Artificial Bee Colony (ABC) [7] [11], a population-based metaheuristic algorithm, is a population-based metaheuristic algorithm. This algorithm is inspired by honey bees. Honey bees: ABC imitates the intelligent behaviour of honey bees. ABC consists of three phases, such as employed, onlooker, and scout bee phases. ABC uses this phase to find the best solution. In the first two phases, local search is used. In this phase, ABC selects their food sources by experience and modifies their positions. ABC also takes help from nest mates. But ABC uses randomised nature for finding food sources in the scouts’ phase without any experience and changes their position. ABC has sufficient capacity to balance exploration or diversification and exploitation or intensification operators by local search and global search methods in these three phases and obtain the best solution. One criterion in ABC is that it will forget the t-the position if the t+1th is better.

Simulated annealing [9] [13] is proposed by Kirkpatrick et al. as a local search algorithm. Annealing means cooling material in a heat bath. It follows the way of thermodynamic systems. Temperature is the control parameter here. The initial temperature will always be high. And the final temperature will always be 0. It is based on the state and physical system of that state. SA is a probability-based algorithm. The state will change whether or not it is deepened in profanities. Another new swarm-based metaheuristic algorithm is called Whale Optimization [14]. WOA mimics the hunting mechanism of humpback whales. This prey behaviour of humpback whales is called the "bubble-net feeding method."

Various journals and conferences like International Journal of Intelligent Systems, Willey, July 2021, IEEE, have provided some well-known algorithms like the Artificial Gorilla Troops Optimizer. And the work has been proposed by Benyamin Abdollahzadeh, Farhad Soleimanian Gharehchopogh, and Seyedali Mirjalili [15]. They provided a GTO algorithm inspired by the social behaviours of gorillas. Through their work, it’s proved that GTO is a new comer algorithm which can skip local optima in an easy way. A GTO has recently been proposed for holding global optimization problems. In their proposed algorithm, a total of 52 benchmark functions have been tested and seven engineering problems have been provided. In their proposed work, the performance analysis has been compared with another nine-metaheuristic optimization

algorithm. They conclude that the AGTO algorithm outperforms its competitors MFO (moth-flame optimization), PSO (particle swarm optimization), GWO (gray wolf optimizer), TSA (Tree-Seed Algorithm), PFA (Pathfinder algorithm), and EO (Equilibrium optimizer) in terms of results and convergence criteria. Among them, GTO is the best metaheuristic algorithm. It is also proved by Friedman's test [15] and the Wilcoxon rank-sum test [15].

A. Ginidi et al. [16] used GTO to extract parameters from several PV (Power-Voltage) systems. According to PK Krishna et al. [17], clustering has been implemented by the concept of the Gorilla Troops Optimizer. According to their paper, clustering has been implemented through the concept of Gorilla Troops behaviours. As different types of troops are implanted by gorillas, like adult males, they leave their group for adult females and make a separate group or their own cluster. Actually, in this paper, this unique behaviour has been picked up for the formation of clusters. Among various nodes in the Wireless Sensor Network, the one with the highest energy has been chosen as the cluster head. And then, based on which node has the third highest energy, they break away from the cluster and form their own. In this way, they show in their research work how a cluster has been formed by the concept of Gorilla troops' behaviours. Another sensitive point from GTO is that communication between gorillas from one group to another group, whether they move or not, food source, direction, everything takes place through the Silverback Gorilla. Likewise, communication between two nodes is done by the cluster head. M. Abdel et al. [18] applied GTO to PEMFC (efficient parameter estimation algorithm for proton exchange membrane fuel cells). According to their article, according to their article, as PEMFC is non-linear and deterministic in nature, it is not suitable for another optimization algorithm. The proposed algorithm GTO is sufficiently suitable for it. According to their literature, GTO is affected by local optima as well as slow convergence speeds. They used MGTO (Modified Gorilla Troops Optimizer). The exploitation operator has been replaced here only to avoid local optima. They used MGTO for estimating the three PEMFCs, or Parameter Estimation Algorithm for Proton Exchange Membrane Fuel Cells, which are: 250 W, SR-12 stack, and BCS-500W stack. They used this estimated parameter to minimise the error between estimated data points and measured data points as the objective function. They compared the PEMFC results to GTO, MGTO, and another eight optimization algorithms: Slime Mold Algorithm (SMA), Modified Farmaland Fertility Optimization Algorithm (MFFA), Moth-flame Optimization (MFO), Coyote Optimization Algorithm (COA), and Modified Monarh Butterfly Optimization (MMBO). They analyse their performance based on standard deviation or SD, best, average, worst, mean absolute percentage error (MAPE), and mean absolute error (MAE). After comparative analysis, they observed that MGTO is the best compared to other performance metrics. Muhammad Kamal Amjad et al. [19] applied genetic algorithms to FJSSB (Flexible Job Shop Scheduling Problem). This paper is a comprehensive review of the Genetic Algorithm solution to the Flexible Job Shop Scheduling Problem (FJSSP).is difficult to find a reasonable solution to FJSSP as it is a NP-hard problem. Only GA is suitable for it. In another research work, Jatin Garg et al. [20] applied genetic algorithms to their proposed work. According to their work, GA is suitable for ELD problems with non-convex as well as continuous cost functions including various constraints. K. Beulah Suganthy used a genetic algorithm to detect disease in plant leaves. They proposed a software solution for detecting disease in plats by GA. Swati Sharma et al. [21] used genetic algorithms and particle swarm optimization for heart disease prediction. With the help of these two algorithms, they got high predictive accuracy. They also observed GA and PSO have high-level prediction and detection rules for detecting heart disease. The Ant

Colony Optimization algorithm is used by other researchers, Esra Sarac et al. [32] In their work, they developed feature selection system which is an ant colony-based. For web page classification, they used ACO. For this purpose, four types of feature extraction methods have been used, namely, URL, title

JP Sarkar et al. [27] proposed a machine learning integrated ensemble technique for the purpose of feature selection and predicting the most influential biomarkers. And they used NGS (Next Generation Sequencing) data for biomarker identification. They divided their work into two parts, i.e., machine learning integrated ensemble technique for classification accuracy measure of the entire dataset, and they used eight different selection techniques for giving the rank of each feature. As a result, they observed 27 miRNAs (micro–Ribonucleic Acid) biomarkers that are highly accompanied with breast cancer. Also, they used different methods for survival analysis purposes. P. Patowary et al. [28] used a bi-clustering gene selection technique for the biomarker's identification on a microarray dataset. And they found twelve secondary genes that are highly associated with primary genes. P. Giannos et al. [29] proposed a bioinformatics model for identifying the most influenced gene biomarkers for lung cancers. In another research work, PP Debata et al. [30] used a multi-objective metaheuristic algorithm for identifying the most influenced biomarkers or genes in a high throughput cancerous dataset. They used multi-objective particle swarm optimization, i.e., MOPSO, multi-objective genetic algorithms, multi-objective jaya algorithms, i.e., MOCJaya, and multi-objective chaotic jaya, as well as genetic algorithms. After a comparative study, they got high accuracy on the MOCJaya meta-heuristic algorithm. In another research by D. Popovic et al. [31], they used genetic algorithm for biomarker identification for colon cancer. A genetic algorithm helped to select the optimal subset of genes for biomarker identification. They got significant results on it.

3. Problem Definition-

The problem definition in this work is to find critical genes from high dimensional genomic dataset. The brief introduction of gene selection technique and proposed work has explained in section 3.1 and 4.

3.1 Gene selection

The main objective of the gene selection technique is to identify critical genes for future treatment. In computational biology, a lot of critical genes are there. Like tumour suppressor genes, cystic fibrosis genes, down syndrome genes, thalassemia genes for thalassemia disease, HTT genes for huntingtin disease, P53 genes, proto-oncogenes, RB are cancer-causing genes. HER2 genes cause breast cancer disease. Not only this, there are many critical genes which are highly associated with our lives.

Various technologies have failed to detect disease. They do not always measure the individual's risk properly and accurately. Hence, to handle this disease, it needs to concentrate on advanced technologies. In this work, an advanced methodology for identifying interesting biomarkers or genes from high-dimensional genomic datasets will be proposed. The main purpose of biomarker detection and prediction is to aid in the finding of genes, proteins, or other biological indicators that may be related to a specific clinical condition. By automating a portion of this process, wet-lab analysis and clinical trial expenses can be decreased sustainably, motivating a slew of current research initiatives in this regard.

Changes in genetic material can be identified in several states by biomarkers. They can find how the body is measurable. A wide range of biomarkers are in computational biology. Biomarkers help to determine the effect of an investigational drug on people. For cancer cells, cells undergo changes. A cancer biomarker measures the chance of cancer developing, cancer prognosis, or responding to a specific therapy. It can be a decision-maker in oncology, which is the study of cancer. This biomarker is linked with specific cancer pathogenesis. Cancer biomarkers can help to screen for any cancer, help to predict risk, help to develop targeted therapies, and can monitor patient responses to that particular cancer treatment. Novel cancer biomarkers are continuously being researched and identified. Biomarker detection helps with feature selection purposes. It can be used to distinguish between normal cells and cancerous cells. For various purposes, miRNA [22] as a biomarker is impressive in today's research field. Biomarkers can be classified into four types: molecular, psychological, histologic, and radiographic. Biomarkers can be found using the gene selection method, bi-clustering [23] [25], gene regulatory networks (GRNs) [23] [24], etc. The gene selection method is a traditional method for biomarker identification. This technique can be divided into four categories based on how the selection process of genes is associated with the classification process. These types are basically hybrids, wrappers, filters, and embedded approaches. The interesting thing is that biologists concentrate on the identification of biomarkers of disease and also the development of accurate predictive tools. With the help of this gene selection data, one can rank genes based on their classification suitability. According to this technique, genes are classified into classes based on their ability. Biomarkers can be easily found by using a gene selection technique using a feature selection method. Also, which biomarkers are highly associated with any disease that can be found? This can give a lot of information about any disease. Disease detection, as well as various cancerous cells, can be easily found using gene selection techniques and biomarker detection. Selection methods can be defined in two ways, i.e., feature extraction and method. The figure is given below. The Wrapper method of feature selection uses metaheuristic algorithms for the purpose of optimal feature selection. A feature selection technique is applied to the search for the most significant genes to find the optimum subset of genes.

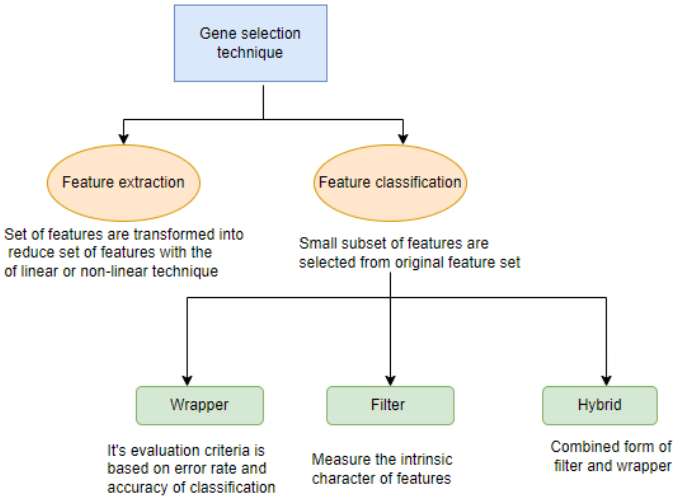


Figure 1- Gene selection technique

DNA (deoxyribonucleic acid) collection [24] The DNA segment is called GRN. These DNA segments interact with each other via RNA (ribonucleic acid) [24] or protein expression data. DNA segments of GRN interact with other substances in the cell as well. With the help of GRNs, biomarkers can be easily found using the feature selection technique.

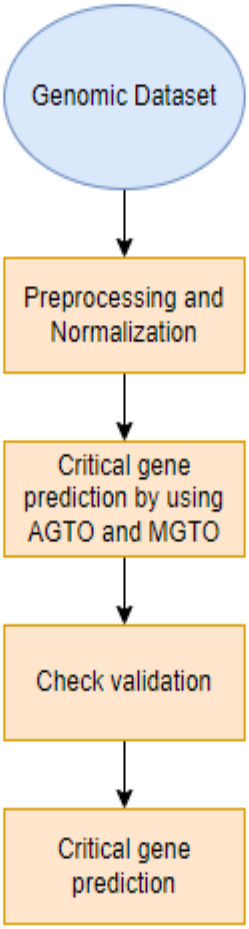
Bi-clustering [25] is a technique that helps to find similar expressions among genes. In a nutshell, it is useful to distinguish genes that are co-related in a subset of conditions that will have similar responses. It is a method that simultaneously groups genes and conditions. If they stay, bi-clustering helps to find distinguishable outline patterns in the matrices of the gene expression data. This technique is an effective technique for identifying interesting biomarkers for high-dimensional genomic datasets. Using this analysis, biomarkers can be found. Except for this technique, a lot of techniques have been proposed in bioinformatics for biomarker or critical genes prediction and detection.

4. Proposed Work

The main goal is to identify critical genes from gene expression profiles for different types of diseases. In bioinformatics many sources are here for collecting genomic datasets. Like, National Centre for Biotechnology Information (NCBI) the national library repository, Gene Expression Omnibus (GEO). The cancer genomes atlas (TCGA) etc. In future dataset will be collected from NCBI if possible. But genomic datasets are backdated. NGS or Next Generation Sequencing marks and determines the DNA or RNA sequencing technology at lower cost. And it has revolutioniz4d the research field of genes. It sequencing billions and millions of small fragments of RNA or DNA nucleotides. So, NGS data of diseases will be collected for future work after experiment with micro-array genomic dataset.

Generally, candidate solutions are generated by meta-heuristic algorithms. An Artificial Gorilla Troops Optimizer has already been developed based on a mathematical example of it, which gives the best candidate solution. Selected genes will be treated as candidate solutions. This algorithm will be applied in bioinformatics for gene selection purposes to make this algorithm stronger. And this is the future work proposed. There are a lot of gene selection methods, but they suffer from low classification accuracy. Hence, a universal feature selection method called the "wrapper approach" will be introduced based on the Gorilla Troops Optimizer. The core of future proposed work is the Gorilla Troops Optimizer, where the most significant and influential biomarkers or genes will be identified from any genomic data with high classification accuracy. For this purpose, the ensemble technique (Support vector machine (SVM), Random Forest (RF), Decision Tree (DT), K Nearest Neighbour (KNN), Nave Bayes (NB)) of machine learning will be applied to the entire set of datasets and classification accuracy will be measured. Also, a feature selection method will be applied to rank the features of entire datasets. The most significant and influenced genes in genomic datasets will be searched by the filter of the feature classification technique, which is followed by the wrapper approach for searching the optimum subset of genes. In bioinformatics, various feature selection techniques are here, like MIM (Mutual Information Maximization), CMIM (Conditional Mutual Information Maximization), CONDRED, etc. Based on this future work, an integrated method will be developed for selecting

genes in any genomic dataset. For this purpose, this meta-heuristic algorithm AGTO has been tested on benchmark function for measuring its superiority and competitiveness by its performance (convergence speed, execution speed). And it gives excellent result on benchmark function. Based on it's performance ability a subset of features will be optimised from the high-throughput genomics datasets. After features are optimised, the performance, i.e., classification accuracy, will also be measured. The optimization process will repeat iteratively for feature optimization. In gene selection, each gene index is a candidate solution. An artificial gorilla troop optimizer will be used for gene selection purposes to generate a gene index. According to their rank or index number, the best solution will be selected accordingly. The future propose work that has been represented below by the flow diagram. How the propose work will be evaluated has been figured below. And how the entire work will be designed that has described in section 3 and 3.1.



Flowchart of critical gene prediction from any genomic dataset

Figure 2- Flowchart of critical gene prediction from any genomic dataset

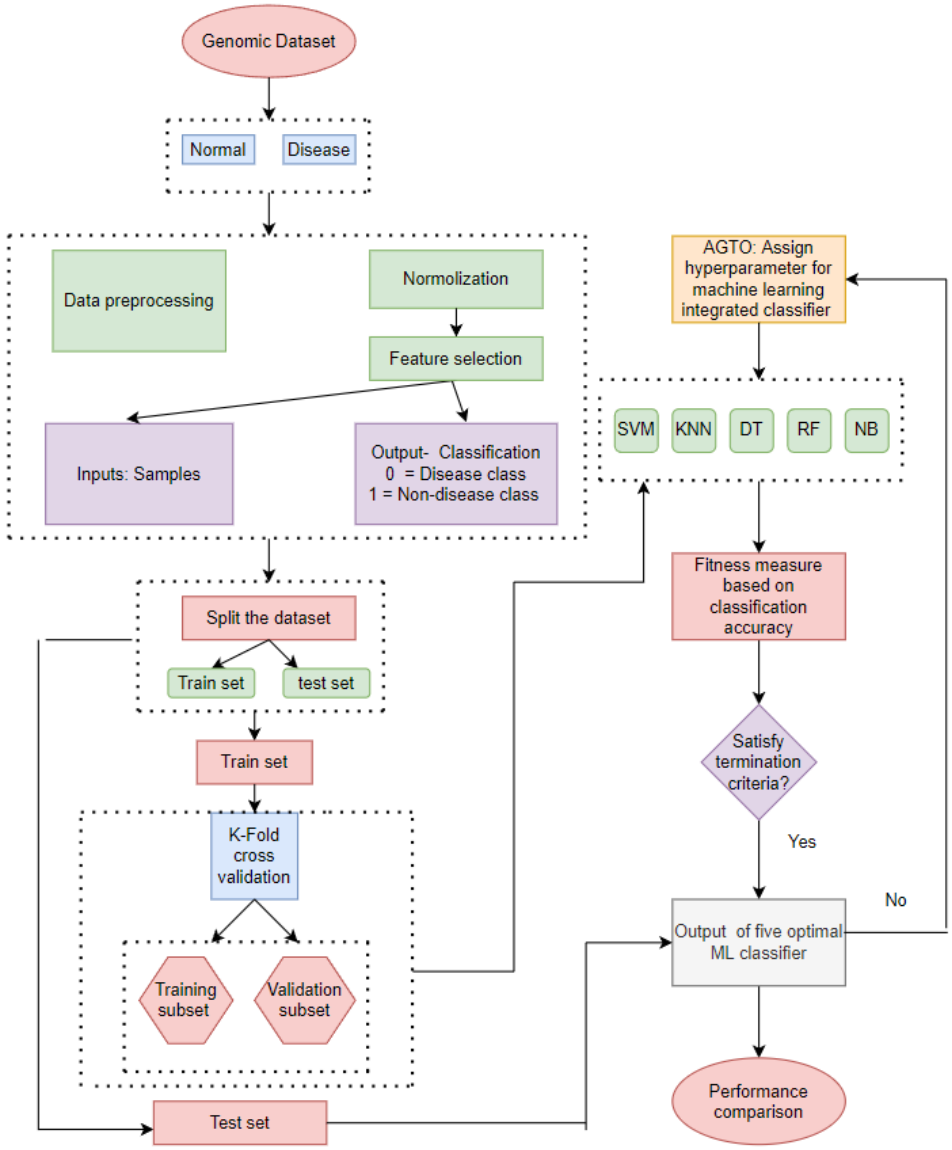


Figure 2.1- Future propose model

4.1 Genomic dataset

For gene prediction purposes, a series matrix file using the.txt format is needed. For txt format to excel, it's necessary to paste into Excel for the purpose of pre-processing data. In the series matrix file, the genes' names are here only. The complete gene information is presented in a GPL (GNU Public Licence Text File) file. Gene selection is very outdated. Researchers are researching some new data. NGS (Next Gene Sequencing) is a popular method now for gene selection purposes.

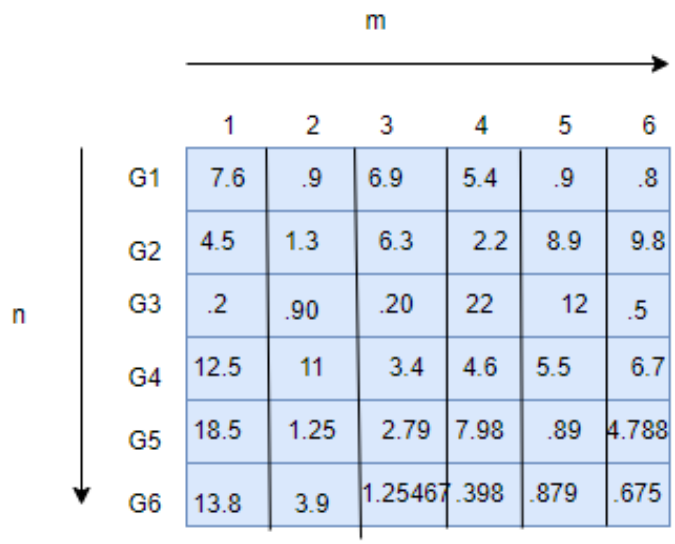


Figure 3- mXn matrix

Assume a matrix of size m x n. In this case, genes = m. Each row is referred to as a gene (G1, G2, G3, G4, G5, G6). Suppose there are 20,000 to 30,000 genes present here. Each gene is called a "feature." And each gene index is considered a candidate solution in the meta-heuristic algorithm. A subset of data called samples Here, mixed samples are present. Mixed samples mean samples of patients and samples of normal people. For their own advantages, researchers separate these two samples of datasets. We have the same names for genes, but the difference is in the gene expression value or protein production rate. Those who will have disease expression values that are different from normal people. From the above dataset, it can be observed that gene expression values or protein point values of genes are different. This type of dataset has been created by DNA micro-array technology. The structural units of genes are called proteins. Due to the abnormality of protein production rate, diseases are formed.

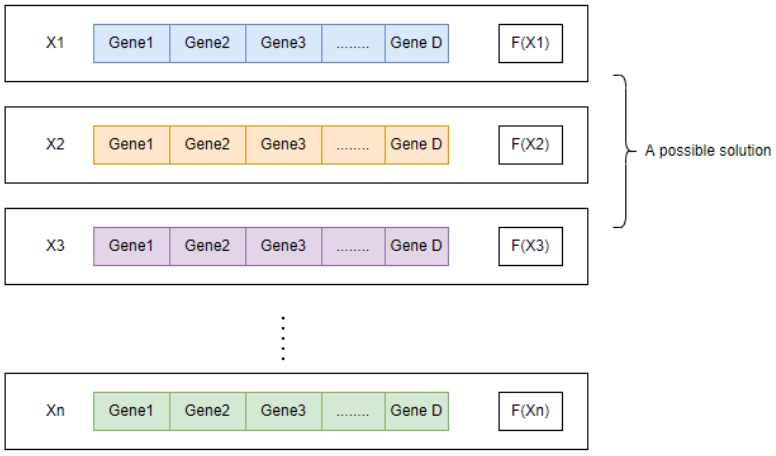
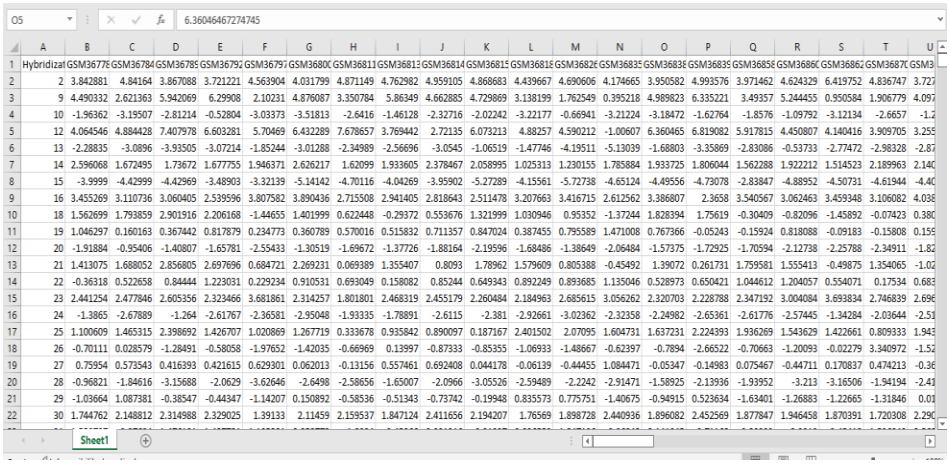


Figure 3.1- Sample population of AGTO

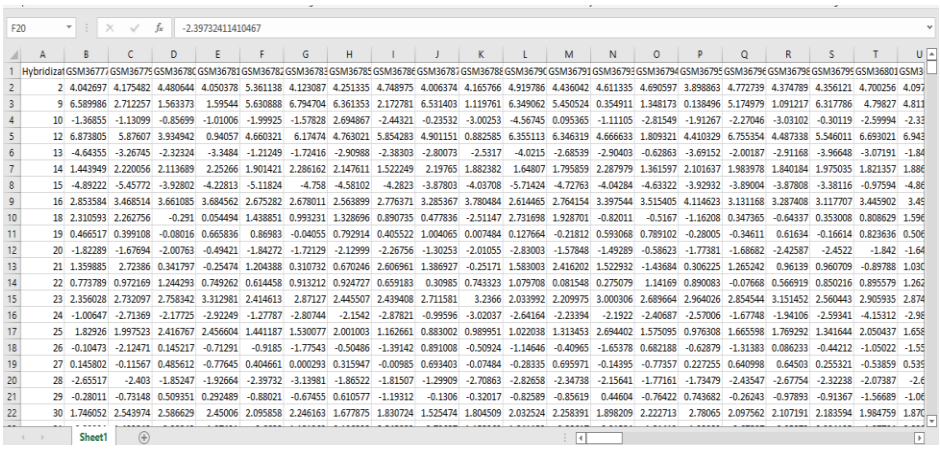
How sample population will be on AGTO has figured in figure 3.1. Here X_i represent a Candidate Gorilla position vector. And F (x_i) represent the fitness value of each gorilla. Each candidate gorillas carries D number of genes. It may be critical or normal.

The future proposed model that has been illustrated in Figure 2 and Figure 2.1 that will be applied to a high-dimensional genomic dataset. This dataset contains some meaningful biological information, like the genome and DNA of an organism, which is presented in the form of micro-arrays. Each micro-array has many features and gene-ids also. An example of a dataset has been given below.



Hybridizat	GSM3677	GSM3678	GSM3678	GSM3679	GSM3680	GSM3681	GSM3681	GSM3681	GSM3682	GSM3683	GSM3683	GSM3683	GSM3685	GSM3686	GSM3686	GSM3687	GSM3
2	3.842881	4.84164	3.867088	3.721221	4.563904	4.031799	4.871149	4.762982	4.959105	4.868683	4.439667	4.690606	4.174665	3.950582	4.993576	3.971462	4.624329	6.419752	4.836747	3.727
9	4.490332	2.621363	5.942069	6.29908	2.10231	4.876087	3.350784	5.86349	4.662885	4.729869	3.138199	1.762549	0.395218	4.989823	6.335221	3.49357	5.244455	0.950584	1.906779	4.097
10	-1.96362	-3.19507	-2.81214	-0.52804	-3.03373	-3.51813	-2.6416	-1.46128	-2.32716	-2.02242	-3.22177	-0.66941	-3.21224	-3.18472	-1.62764	-1.8576	-1.09792	-3.12134	-2.6657	-1.2
12	4.064546	4.884428	7.407978	6.603281	5.70469	6.432289	7.678657	3.769442	2.72135	6.073213	4.88257	4.590212	-1.00607	6.360465	6.819082	5.917815	4.450807	4.140416	3.909705	3.255
13	-2.28835	-3.0896	-3.93505	-3.07214	-1.85244	-3.01288	-2.34989	-2.56696	-3.0545	-1.06519	-1.47746	-4.19511	-5.13039	-1.68803	-3.35869	-2.83086	-0.53733	-2.77472	-2.98328	-2.87
14	2.596068	1.672495	1.73672	1.677755	1.946371	2.626217	1.62099	1.933605	2.378467	2.058995	1.025313	1.230155	1.785884	1.933725	1.806044	1.562288	1.922212	1.514523	2.189963	2.140
15	-3.9999	-4.42999	-4.42969	-3.48903	-3.32139	-5.14142	-4.70116	-4.04269	-3.95902	-5.27289	-4.15561	-5.72738	-4.65124	-4.49556	-4.73078	-2.83847	-4.88952	-4.50731	-4.61944	-4.40
16	3.455269	3.110736	3.060405	2.539596	3.807582	3.890436	2.715508	2.941405	2.818643	2.511478	3.207663	3.416715	2.612562	3.386807	2.3658	3.540567	3.062463	3.459348	3.106082	4.038
18	1.562699	1.793859	2.901916	2.206168	-1.44655	1.401999	0.622448	-0.29372	0.553676	1.321999	1.030946	0.95352	-1.37244	1.828394	1.75619	-0.30409	-0.82096	-1.45892	-0.07423	0.380
19	1.046297	0.160163	0.367442	0.817879	0.234773	0.360789	0.570016	0.515832	0.711357	0.847024	0.387455	0.795589	1.471008	0.767366	-0.05243	-0.15924	0.818088	-0.09183	-0.15808	0.155
20	-1.91884	-0.95406	-1.40807	-1.65781	-2.55433	-1.30519	-1.69672	-1.37726	-1.88164	-2.19596	-1.68486	-1.38649	-2.06484	-1.57375	-1.72925	-1.70594	-2.12738	-2.25788	-2.34911	-1.82
21	1.413075	1.688052	2.856805	2.697696	0.684721	2.269231	0.069389	1.355407	0.8093	1.78962	1.579609	0.805388	-0.45492	1.39072	0.261731	1.759581	1.555413	-0.49875	1.354065	-1.02
22	-0.36318	0.522658	0.84444	1.223031	0.229234	0.910531	0.693049	0.158082	0.85244	0.649343	0.892249	0.893685	1.135046	0.528973	0.650421	1.044612	1.204057	0.554071	0.17534	0.683
23	2.441254	2.477846	2.605356	2.323466	3.681861	2.314257	1.801801	2.468319	2.455179	2.260484	2.184963	2.685615	3.056262	2.320703	2.228788	2.347192	3.004084	3.693834	2.746839	2.696
24	-1.3865	-2.67889	-1.264	-2.61767	-2.36581	-2.95048	-1.93335	-1.78891	-2.6115	-2.381	-2.92661	-3.02362	-2.32358	-2.24982	-2.65361	-2.61776	-2.57445	-1.34284	-2.03644	-2.51
25	1.100609	1.465315	2.398692	1.426707	1.020869	1.267719	0.333678	0.935842	0.890097	0.187167	2.401502	2.07095	1.604731	1.637231	2.224393	1.936269	1.543629	1.422661	0.809333	1.943
26	-0.70111	0.028579	-1.28491	-0.58058	-1.97652	-1.42035	-0.66969	0.13997	-0.87333	-0.85355	-1.06933	-1.48667	-0.62397	-0.7894	-2.66522	-0.70663	-1.20093	-0.02279	3.340972	-1.52
27	0.75954	0.573543	0.416393	0.421615	0.629301	0.062013	-0.13156	0.557461	0.692408	0.044178	-0.06139	-0.44455	1.084471	-0.05347	-0.14983	0.075467	-0.44711	0.170837	0.474213	-0.36
28	-0.96821	-1.84616	-3.15688	-2.0629	-3.62646	-2.6498	-2.58656	-1.65007	-2.0966	-3.05526	-2.59489	-2.2242	-2.91471	-1.58925	-2.13936	-1.93952	-3.213	-3.16506	-1.94194	-2.41
29	-1.03664	1.087381	-0.38547	-0.44347	-1.14207	0.150892	-0.58536	-0.51343	-0.73742	-0.19948	0.835573	0.775751	-1.40675	-0.94915	0.523634	-1.63401	-1.26883	-1.22665	-1.31846	0.01
30	1.744762	2.148812	2.314988	2.329025	1.39133	2.11459	2.159537	1.847124	2.411656	2.194207	1.76569	1.898728	2.440936	1.896082	2.452569	1.877847	1.946458	1.870391	1.720308	2.290

Table 1 Screenshot of Genomic dataset (Series- GSE2023 Normal)



Hybridizat	GSM3677	GSM3677	GSM3678	GSM3678	GSM3678	GSM3678	GSM3678	GSM3678	GSM3678	GSM3678	GSM3679	GSM3679	GSM3679	GSM3679	GSM3679	GSM3679	GSM3679	GSM3679	GSM3680	GSM3
2	4.042697	4.175482	4.480644	4.050378	5.361138	4.123087	4.251335	4.748975	4.006374	4.165766	4.919786	4.436042	4.611335	4.690597	3.898863	4.772739	4.374789	4.356121	4.700256	4.097
9	6.589986	2.712257	1.563373	1.59544	5.630888	6.794704	6.361353	2.172781	6.531403	1.119761	6.349062	5.450524	0.354911	1.348173	0.138496	5.174979	1.091217	6.317786	4.79827	4.811
10	-1.36855	-1.13099	-0.85699	-1.01006	-1.99925	-1.57828	2.694867	-2.44321	-0.23532	-3.00253	-4.56745	0.095365	-1.11105	-2.81549	-1.91267	-2.27046	-3.03102	-0.30119	-2.59994	-2.33
12	6.873805	5.87607	3.934942	0.94057	4.660321	6.17474	4.763021	5.854283	4.901151	0.882585	6.355113	6.346319	4.666633	1.809321	4.410329	6.755354	4.487338	5.546011	6.693021	6.943
13	-4.64355	-3.26745	-2.32324	-3.3484	-1.21249	-1.72416	-2.90988	-2.38303	-2.80073	-2.5317	-4.0215	-2.68539	-2.90403	-0.62863	-3.69152	-2.00187	-2.91168	-3.96648	-3.07191	-1.84
14	1.443949	2.220056	2.113689	2.25266	1.901421	2.286162	2.147611	1.522249	2.19765	1.882382	1.64807	1.795859	2.287979	1.361597	2.101637	1.983978	1.840184	1.975035	1.821357	1.886
15	-4.89222	-5.45772	-3.92802	-4.22813	-5.11824	-4.758	-4.58102	-4.2823	-3.87803	-4.03708	-5.71424	-4.72763	-4.04284	-4.63322	-3.92932	-3.89004	-3.87808	-3.38116	-0.97594	-4.86
16	2.853584	3.468514	3.661085	3.684562	2.675282	2.678011	2.563899	2.776371	3.285367	3.780484	2.614465	2.764154	3.397544	3.515405	4.114623	3.131168	3.287408	3.117707	3.445902	3.45
18	2.310593	2.262756	-0.291	0.054494	1.438851	0.993231	1.328696	0.890735	0.477836	-2.51147	2.731698	1.928701	-0.82011	-0.5167	-1.16208	0.347365	-0.64337	0.353008	0.808629	1.596
19	0.466517	0.399108	-0.08016	0.665836	0.86983	-0.04055	0.792914	0.405522	1.004065	0.007484	0.127664	-0.21812	0.593068	0.789102	-0.28005	-0.34611	0.61634	-0.16614	0.823636	0.506
20	-1.82289	-1.67694	-2.00763	-0.49421	-1.84272	-1.72129	-2.12999	-2.26756	-1.30253	-2.01055	-2.83003	-1.57848	-1.49289	-0.58623	-1.77381	-1.68682	-2.42587	-2.4522	-1.842	-1.64
21	1.359885	2.72386	0.341797	-0.25474	1.204388	0.310732	0.670246	2.606961	1.386927	-0.25171	1.583003	2.416202	1.522932	-1.43684	0.306225	1.265242	0.96139	0.960709	-0.89788	1.030
22	0.773789	0.972169	1.244293	0.749262	0.614458	0.913212	0.924727	0.659183	0.30985	0.743323	1.079708	0.081548	0.275079	1.14169	0.890083	-0.07668	0.566919	0.850216	0.895579	1.262
23	2.356028	2.732097	2.758342	3.312981	2.414613	2.87127	2.445507	2.439408	2.711581	3.2366	2.033992	2.209975	3.000306	2.689664	2.964026	2.854544	3.151452	2.560443	2.905935	2.874
24	-1.00647	-2.71369	-2.17725	-2.92249	-1.27787	-2.80744	-2.1542	-2.87821	-0.99596	-3.02037	-2.64164	-2.23394	-2.1922	-2.40687	-2.57006	-1.67748	-1.94106	-2.59341	-4.15312	-2.98
25	1.82926	1.997523	2.416767	2.456604	1.441187	1.530077	2.001003	1.162661	0.883002	0.989951	1.022038	1.313453	2.694402	1.575095	0.976308	1.665598	1.769292	1.341644	2.050437	1.658
26	-0.10473	-2.12471	0.145217	-0.71291	-0.9185	-1.77543	-0.50486	-1.39142	0.891008	-0.50924	-1.14646	-0.40965	-1.65378	0.682188	-0.62879	-1.31383	0.086233	-0.44212	-1.05022	-1.55
27	0.145802	-0.11567	0.485612	-0.77645	0.404661	0.000293	0.315947	-0.00985	0.693403	-0.07484	-0.28335	0.695971	-0.14395	-0.77357	0.227255	0.640998	0.64503	0.255321	-0.53859	0.535
28	-2.65517	-2.403	-1.85247	-1.92664	-2.39732	-3.13981	-1.86522	-1.81507	-1.29909	-2.70863	-2.82658	-2.34738	-2.15641	-1.77161	-1.73479	-2.43547	-2.67754	-2.32238	-2.07387	-2.6
29	-0.28011	-0.73148	0.509351	0.292489	-0.88021	-0.67455	0.610577	-1.19312	-0.1306	-0.32017	-0.82589	-0.85619	0.44604	-0.76422	0.743682	-0.26243	-0.97893	-0.91367	-1.56689	-1.06
30	1.746052	2.543974	2.586629	2.45006	2.095858	2.246163	1.677875	1.830724	1.525474	1.804509	2.032524	2.258391	1.898209	2.222713	2.78065	2.097562	2.107191	2.183594	1.984759	1.870

Table 1.1 Screenshot of Genomic dataset (Series- GSE2023 Tumor**)**

In this genomic dataset, each row describes a gene. The first column represents a gene-id. Every numeric value present in a dataset is called a gene expression value or protein value. This gene expression value, or protein value, is different for everyone. Hence, the characteristics of humans differ from each other. This dataset has been taken from the NCBI (National Centre for Biotechnology Information). The GENE expression profile of the dataset consists of a total of 12,634 genes and 286 breast cancer samples. Out of 286 breast cancer samples, 107 contained

tumour samples and 179 contained non-tumour samples. The advantage is that this type of dataset contains a huge amount of information, but there are many difficulties in working with such types of datasets. As datasets are large in size, they may contain irrelevant data or outliers. Due to high dimensionality, there can be missing values or noisy data. So, it needs to be pre-processed and normalised before applying the gene selection method to it. After pre-processing and normalising the dataset, the meta-heuristic algorithm called the Artificial Gorilla Troops Optimizer will be used for critical gene prediction. After prediction, it's important to use a validation procedure on the resulting data. How the dataset will be pre-processed and normalised has been figured out in figures 2 and 2.1.

5. Methodologies

The methodology of the Artificial Gorilla Troops Optimizer (AGTO) and Modified Gorilla Troops Optimizer (MGTO) is explained below.

5.1 Metaheuristic algorithm

Nowadays, various algorithms concentrate on meta-heuristic techniques [9] [26], as it's not possible to get an exact or proper solution through optimization techniques. This general-purpose optimization technique is not limited to a specific problem. It helps to improve candidate solutions iteratively till an acceptable solution is obtained. The main advantage is that it can find a global optimal solution. We can get a solution closer to the best solution, but not exact one. Like greedy, Divide and Conquer, Dynamic Programming [9] [26] generates a lot of assumptions. The cons are that search space can exponentially grow. Hence, meta-heuristic algorithms are important for solving many complex real-life problems. For solving different types of optimization problems, meta heuristic techniques play a significant role. For optimising problems, optimization algorithms show two types of behaviours, such as deterministic and stochastic [9] [26]. A deterministic optimization algorithm is less practical and less applicable and requires complex calculations. Stochastic algorithms are stochastic in nature. It is more practical and more applicable compared to a deterministic algorithm. It is random in nature. The meta heuristic technique helps to find a near-optimal solution instead of an optimal one. It is computationally faster than the exhaustive search technique. This technique is iterative in nature and involves stochastic operations [26]. It is iterative in nature as it follows different types of iteration for modifying candidate solutions in search space [9] [26]. Usually, our daily life problems aren't solved by optimization techniques. The Meta heuristic technique is one of the techniques that can solve real-life problems. Advanced machine learning techniques employ a meta-heuristic [9] [26] approach that can produce an appropriate result without relying on a variety of assumptions. It has a wide range of applications in various fields like business sectors, research section, intelligent traffic systems, bioinformatics field, marketing sector, engineering, and the medical field, i.e., health care and medicine. A common theme for all meta-heuristic techniques is the exploration or diversification phase [26] as well as the exploitation or intensification phase and providing a superior balance between them [26]. There are some unique characteristics present in meta-heuristic algorithms, such as: this algorithm is simple and very easy to implement [26], the set of rules is straightforward and really smooth to put into effect they outsail local search algorithms [26], in a broad range of applications they can easily be used, there is no need to include any derivative function here [26], and they can avoid local optima by

accepting some bad solutions. Meta-heuristic techniques can be distinguished into two types. It can be global or local-based [26], i.e., a single solution, which contains a single candidate solution, or population-based, which contains multiple candidate solutions.

5.2 List of notations

A list of notations and symbols has been presented below in table 2.

Symbol	Notation
D	Dimension of problem
t	Current iteration number
Maxt	Maximum no of iterations
N	Total number of gorillas (Population size)
X_i	Position of i-th gorilla (Vector of dimension D)
GX_r	Randomly selected r-th gorillas from the entire population
$X_{silverback}$	Silverback position vector (The best gorilla's position vector, best solution)
UB, LB	Lower and upper bound of each element present in D dimensional vector.
rand, r_1, r_2, r_3, r_4, r_5, W, a, b	Random numbers between 0 and 1
p	p is a predefined probability between 0 and 1.
F	Cosine function
l	random number between -1 and 1
g	Variables, scalar value
β	Constant value
N_1	random values in the normal distribution and the problem dimension
N_2	random value in normal distribution
E	The value of E will equal to N_1 If rand ≥0.5 Otherwise, the value of E will equal to N_2 if $rand < 0.5$

Table 2 List of Symbol and notation

5.3 The Artificial gorilla troops optimization algorithm (AGTO)

A new population-based meta-heuristic optimization algorithm called the "Artificial Gorilla Troops Optimizer" which mimics the gorilla group's life behaviour. And this social behaviour has been mathematically formulated. Here, "troops" means groups of gorillas. An adult male gorilla in troops called the Silverback gorilla. Troops consist multiple adult female gorillas also, and their progeny also. A silverback gorilla is normally over 12 years old and is known for the distinctive (silver colour hair) hair on its back during adolescence. Additionally, the silverback serves as the troop's leader, making all decisions, managing conflicts, directing others to food searches, determining group movement, and ensuring group security. Male gorillas between the ages of eight (8) and twelve (12) years are referred to as "blackbacks" gorilla due to their lack of

silver-coloured back hair. They are associated with the silverback and serve as the group's backup defenders. Male and female gorillas, in general, move from their birth group to a second new group. Alternatively, older male gorillas may break away from their original group and form their own battalions through the recruitment of migrating females. However, some male gorillas prefer to remain with the founding troop and follow the silverback. If the silverback perishes, these males may engage in a bloody struggle for group supremacy and mating with adult females. The GTO algorithm's special mathematical model is constructed using the aforementioned concept of gorilla group behaviour in nature.

In AGTO, each candidate gorilla position vector is called a "solution." The sum of these solutions is known as "population". This GTO has many unique features that can be widely used in the future. It uses various mechanisms for optimization purposes. As is the case with other intelligent algorithms, GTO is composed of three major components: initialization, global exploration, and local exploitation, each of which is discussed in detail below.

The algorithm search space comprises mainly three types of candidate solutions. X, GX, and Silverback Gorilla candidate position vectors are generated by each phase that exploration and exploitation of GTO. X is called the gorilla position vector. GX is called the candidate gorilla position vector. The difference is that GX belongs to another memory of GTO. Silverback is the best solution that can be found in each iteration. And only one silverback gorilla will be considered in the whole population when the number of solutions is selected for the optimization process. A total of five strategies are found here, such as migration to an unknown place, migration to a known place, moving towards other gorillas, following the silverback gorilla, and competition for adult females. These five strategies are mimicked and revealed to elucidate the diversification and intensification or the exploration and exploitation of the optimization technique. The first three strategies, i.e., migration to an unknown place, migration to a known place, and moving towards other gorillas, are used in the exploration phase, and the last two strategies, following the silverback gorilla, and competition for adult females, are used in the exploitation phase.

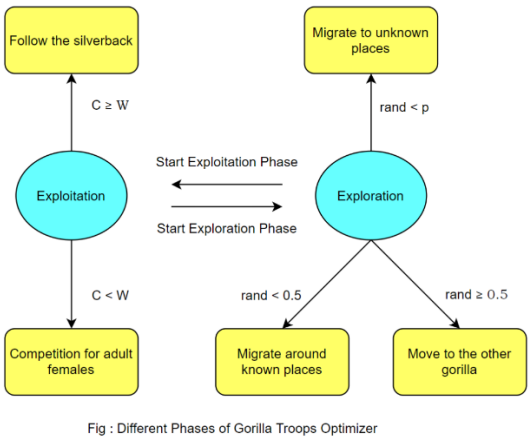


Fig : Different Phases of Gorilla Troops Optimizer

Figure 4- Exploration and Exploitation phase of GTO

5.3.1 Initialization phase

Assume that there are N gorillas in the D-dimensional space. The position of the i-th gorilla the D-dimensional space is represented as $X_i = [x_{i,1}, x_{i,2,\ldots,}x_{i,D}]$, i = 1, 2, 3…, N. In this way the initialization mechanism of gorilla population can be represented as given equation. $X_{N,D} = rand(N, D) * (UB - LB) + LB$. Where UB, LB are upper and lower limits of problem space. rand (N, D) is a representation of matrix with N rows and D columns.

5.3.2 Exploration phase-

In exploration phase, GTO will find the most promising regions by exploring the search space of the optimization technique. This phase includes near-optimal solution.
First, Migration to an unknown place. This mechanism is selected when rand<p where rand is any randomly generated number between 0 & 1, p is user defined parameter. And this mechanism monitors the entire search space well. The mathematical equation is given bellow.

GX(t+1) = (UB – LB) * r_1 + LB when rand < p Equation (1)

Here, p is a predefined probability between 0 and 1. It decides the probability of choosing the movement strategy to an unknown location. GX (t+1) represents the Gorilla's candidate position vector in the next iteration. UB and LB are upper and lower limits of search space r_1 is a random number between 0 and 1.

Second, Migration towards other candidate gorillas. This mechanism is selected when rand≥0.5, where rand is any randomly generated number between 0 and 1. It helps to improve exploration. The mathematical model is,

GX(t+1) = (r_2 – C) * ($GX_{r1}(t)$+L*H when rand ≥ 0.5 Equation (2)

C, L, H are calculated by given equations
C = F * (1 – t/Maxt) Equation (3)
F = cos (2 * (r_4) + 1 Equation (4)
H = Z * X(t) Equation (5)
Z = [-C, C] Equation (6)

Here, rand is random number between 0 and 1. r_2, r_4 is also a random number between 0 and 1. $GX_{r1}(t)$ is randomly selected gorillas position vector at current population. Z is a row vector in D including random numbers generated between -C and C. cos (.) defines the cosine function. C is calculated by above equations (3). t is current iterations and Maxt represents the maximum number of iterations.

Third, Migration to a known location. This mechanism is selected when rand<0.5 where rand is any randomly generated number between 0 and 1. It helps to escape local optima point. This strategy follows the given equation,

$GX(t+1) = X(t) - L * (L*(X(t) - GX_{r2}(t)) + r_3 *(X(t) - GX_{r2}(t)))$ when rand < 0.5 Equation (7)

L is calculated by given equation,

$L = C * l$ Equation (8)

Here, X (t) Current gorilla's position vector of at t iteration. l is random generated number between -1 and 1. And, $GX_{r2}(t)$ is randomly selected gorillas position vector at current population.

rand>0.5 means there have 50% chance to movement other gorillas. rand<0.5 means there have 50% chance to movement to a known location. At the end of the exploration phase the cost values of all newly generated candidate gorillas i.e., GX (t+1)-th solutions are calculated. And if cost of F(GX) < F(X) means current gorilla position vector will update in to GX position vector. Else, X position vector will still in to memory. Here F is actually a fitness function. In this phase $X_{silverback}$ is the optimal solution.

5.3.3 Exploitation phase

After exploring Search space, solutions will go in exploitation phase to get best solution. In this phase, better solution (gorilla candidate position vector) can be found. Based on the updated candidate gorilla position vector in exploration phase, a new population will be made & from that a best solution will be picked up which is called the best solution (Silverback gorilla). This silverback gorilla takes all decision as he is the leader. Two behaviors have been followed here. The behaviors are given below.

First, Follow the Silverback. This mechanism is followed when C≥W. This mechanism is used to simulate the gorilla behavior. The mathematical equation is given below.

$GX(t+1) = L * M * (X(t) - X_{silverback}) + X(t)$ when C≥W Equation (9)

M and g are calculated by given equation,

$M = (|\frac{1}{N}\sum_{i=1}^{N} X_i(t)|^g |^{\frac{1}{g}}|)$ Equation (10)

$g = 2^L$ Equation (11)

Here, W a parameter which needs to be set before optimization operation ranging from [0,1]. $X_{silverback}$ represents the best candidate gorillas position vector. C is calculated by above equation (3) and L is calculated by above equation (8). M is the average value of Candidate Gorilla's position vector. Whatever may be the value of M whether +M or -M, only the +M is taken into account (i.e., absolute value of M will be considered). M can be calculated by above equation (10). Where,
X_i *is the* Position of i-th gorilla (Vector of dimension D). g is calculated by equation (11). N is population size i.e., total number of gorillas.

Second, Competition for the adult females. This mechanism is followed when C<W. Here W is user defined parameter and that will be set before Optimization technique.

GX(t+1) = $X_{silverback} - (X_{silverback} * Q - X(t) * Q)$ * A when C < W Equation (12)

Mathematical equation of Q and A are given bellow

Q = 2*$r_5 - 1$ Equation (13)

A = β * E Equation (14)

E = N_1 $if\ rand \geq 0.5$
N_2 $if\ rand < 0.5$ Equation (15)

X (t) is current position vector of gorillas. Q is a scalar value which simulate the impact force of gorillas that is calculated by above equation (13). r_5 is a random number in between 0 and 1. β is constant value. $N_1$is a vector which contains random values in the normal distribution and the problem dimensions. N_2 is a scaler which is random value in normal distribution. E could be calculated by above equation (15). A represents a coefficient vector to indicate the degree of violence in case of conflicts. It can be calculated by equation (14). At the end of exploitation phase, the cost values of all newly generated candidate gorillas i.e., GX (t+1)-th solutions are calculated. And if the cost if F(GX)<F(X) means current gorilla position vector will update in to GX position vector. Else, X position vector will still in to memory. Here F is actually a fitness function. In this phase $X_{silverback}$ is the optimal solution within the all-candidate solutions. The steps of the entire algorithm of AGTO are presented below.

5.3.4 The Algorithm of AGTO

Set AGTO parameters Maxt, N, β, p, w
Initialize X_i $(i = 1\ to\ N)$
Calculate the fitness value for each X
While t = 1 to Maxt do
    C = F * (1 – t/Maxt)
    F = cos (2 * (r_4) + 1
    L = C * l
    for i = 1: N do
        if rand < p do
            GX(t+1) = (UB – LB) * r_1 + LB
        else
            if rand >= 0.5 do
                GX(t+1) = (r_2 – C) * ($GX_{r1}(t)$+L*H
            else
                GX(t+1) = X(t) – L* (L*(X(t) - $GX_{r2}(t)$)+ r_3 *(X(t) $-GX_{r2}(t)$))
            end
        end
    end
    for i = 1: N do
        if $F(GX_i) < F(X_i)$ do
            $X_i = GX_i$
            $X_{silverback} = GX_i$

else
$GX_i = X_i$
$X_{silverback} = X_i$
end
end
for i = 1: N do
if rand C >= W do
M = ($|\frac{1}{N}\sum_{i=1}^{N} X_i(t)|^{g}|^{\frac{1}{g}}|$)
g = 2^L
GX(t+1) = L * M * (X(t) - $X_{silverback}$) +X(t)
else
if rand >= 0.5
E = N_1
else
E = N_2
end
Q = 2*$r_5 - 1$
A = β * E
GX(t+1) = $X_{silverback} - (X_{silverback} * Q - X(t) * Q)$ * A
end
end
for i = 1: N do
if $F(GX_i) < F(X_i)$ do
$X_i = GX_i$
$X_{silverback} = GX_i$
else
$GX_i = X_i$
$X_{silverback} = X_i$
end
end
t=t+1
end while
return $X_{silverback}, silverback\ score$

5.3.5 AGTO Flow diagram

The flow diagram of the Artificial Gorilla Troops Optimizer with various strategies (Exploration and Exploitation phase) has been figured below.

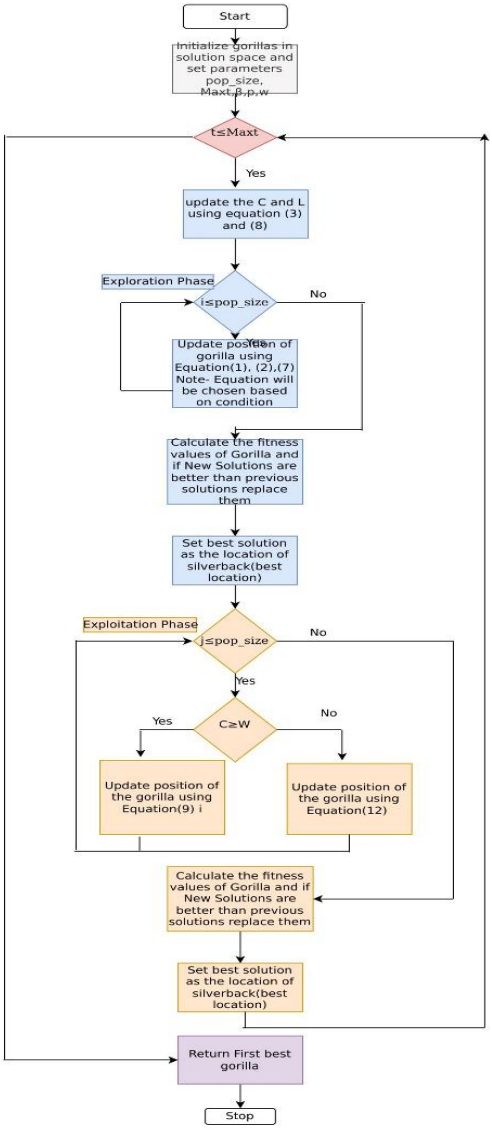


Figure 5- AGTO flow diagram

5.4 Modified Gorilla Troops Optimizer (MGTO)

In GTO, the exploitation phase is still suffering from slow convergence speed. As in GTO, a random number is used significantly, which can increase the step size of the optimization process. And that's why the new solution might be far from the best solution. It might involve the desired near-optimal solution but not be involved in the optimum solution. The equality between intensification and diversification might reduce the performance of the entire GTO algorithm, and hence, in some cases, higher exploration and/or exploitation operators are needed. But this gap can also be handled by MGTO, or modified Gorilla Troops Optimizer, which can give the desired optimum solution. In MGTO, the exploitation operator is replaced by another one. MGTO aids in exploring more promising search regions around the best solution. And one amazing thing is that other places in the promising search regions might be able to give the

desired optimal solution as well as an optimal solution. In MGTO, they will choose randomly the Gorilla position vector instead of random numbers in between 0 and 1. A GTO can fall into premature convergence. The changes are for the exploitation phase only. So, the new exploitation strategies are based on two new mechanisms, where the first one is mathematically explained below.

$$GX(t+1) = X_{silverback} + \beta \times (X_r(t) - X(t)) + (1-\beta) \times (X_{r1}(t) - X_{r2}(t)) \quad \text{when } C \leq 0.5 \qquad \text{Equation (16)}$$

The new second mechanism is also mathematically described.

$$GX(t+1) = X_r(t) + a \times (X_{r1}(t) - X(t)) + b \times (X_{r2}(t) - X_{r3}(t)) \quad \text{when } C > 0.5 \qquad \text{Equation (17)}$$

Here, $X_r(t), X_{r1}(t), X_{r2}(t), X_{r3}(t)$are randomly selected candidate gorilla position vector from current population. a, b are random numbers between 0 and 1. GX (t+1), $X_{silverback}, \beta, C$ are already defined in AGTO portion. The steps of the algorithm of MGTO are presented below.

5.4.1 The Algorithm of AGTO

Set MGTO parameters Maxt, N, β, p, w
Initialize X_i $(i = 1\ to\ N)$
Calculate the fitness value for each X
While t = 1 to Maxt do
    C = F * (1 – t/Maxt)
    F = cos (2 * (r_4) + 1
    L = C * l
    for i = 1: N do
        if rand < p do
            GX(t+1) = (UB – LB) * r_1 + LB
        else
            if rand >= 0.5 do
                GX(t+1) = (r_2 – C) * ($GX_{r1}(t)$+L*H
            else
                GX(t+1) = X(t) – L* (L*(X(t) - $GX_{r2}(t)$)+ r_3 *(X(t) −$GX_{r2}(t)$))
            end
        end
    end
    for i = 1: N do
        if $F(GX_i) < F(X_i)$ do
            $X_i = GX_i$
            $X_{silverback} = GX_i$
        else
            $GX_i = X_i$
            $X_{silverback} = X_i$
        end
    end
    for i = 1: N do
        if C =<0.5 do
            $G(t+1) = X_{silverback} + \beta \times (X_r(t) - X(t)) + (1-\beta) \times (X_{r1}(t) - X_{r2}(t))$

else

$$G(t+1) = X_r(t) + a \times (X_{r1}(t) - X(t)) + b \times (X_{r2}(t) - X_{r3}(t))$$

end

end

for i = 1: N do

if $F(GX_i) < F(X_i)$ do

$X_i = GX_i$

$X_{silverback} = GX_i$

else

$GX_i = X_i$

$X_{silverback} = X_i$

end

end

t=t+1

end while

return $X_{silverback}, silverback\ score$

5.4.2 MGTO Flow diagram

The flow diagram of the MGTO algorithms is given below.

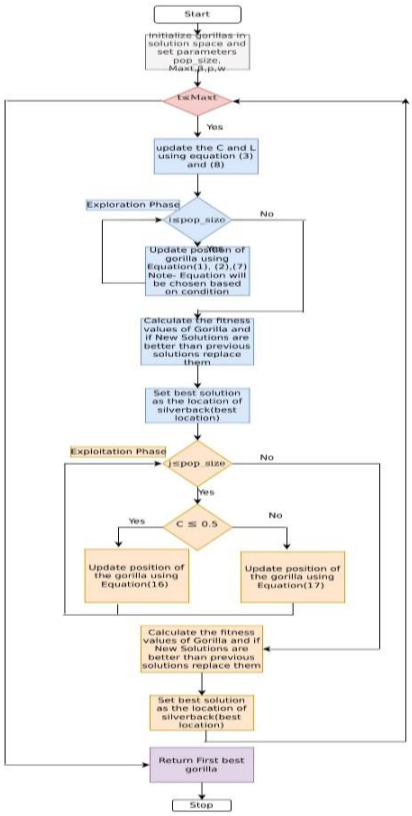


Figure 6- MGTO Flow diagram

6 Experimental Result

The machine is used HP-Pavilion RTL8723DE, having Intel(R) Core (TM) i5-10210U processor with base clock frequency 1.60GHz, 8GB RAM. Programs are written Matlab programming language (version- MATLAB R2021b) and Python programming language (python3 version 3.9.7, Anaconda3) in Windows 11 environment.

6.1 Benchmark Test Functions

In this work the performance of AGTO algorithm is tested on 25 benchmark function that are listed in table 3. Fixed parameter values are used, such as the number of iterations is set to 100, population set as 30, dimension set as 3 and 10 independent runs are taken. The benchmark test functions are listed below.

Function	**Formulation**	**D**	**Range**	f_{opt}
Sphere	$f = \sum_{i=1}^{D} x_i{}^2$	30	[-100,100]	0
Ackley	f(X) $= -20\exp\left(-0.2\sqrt{(1/n)\sum_{i=1}^{n} x_i^2}\right) - \exp\left((1/n)\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	30	[-32,32]	0
Rosenbrock	$f(x) - \sum_{i=1}^{n-1} [100 * (x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	[-30.30]	0
Rastrigin	$f(x) = 10 * \mathrm{D} + \sum_{i=1}^{D} [x_i^2 - 10 * \cos(2\pi * x_i)]$	30	[-5.12,5.12]	0
Schwefel 2.22	$f = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	30	[-10,10]	0
Schwefel 1.22	$f(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j\right)^2$	30	[-100,100]	0
Schwefel 2.21	$\mathrm{f(x)} = \max_i\{\lvert x_i \rvert,\ 1 \le i \le n\}$	30	[-100,100]	0
Step: US	$f = \sum_{i=1}^{D} ([x_i - 0.5])^2$	30	[-100,100]	0
Dixon & Price	$f = (x_1 - 1)^2 + \sum_{i=1}^{D} i(2x_i^2 - x_{i-1})^2$	30	[-10,10]	0
Sum squares	$f = \sum_{i=1}^{D} (ix_i)^2$	30	[-10,10]	0
Griewank	$f = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	[-600,600]	0
Booth	$f(\mathbf{x}) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$	2	[-10,10]	0

Quartic	$f\ (x) = \sum_{i=1}^{D} i x_i^4 + \text{random}\ [0,1]$	30	[-1.28,1.28]	0
Quadratic	$f\ (x) = \sum_{i=1}^{D} (i x_i)^2 + \text{random}\ [0,1]$	30	[-1.28,1.28]	0
Zakharov	$f(\mathbf{x}) = \sum_{i=1}^{n} x_i^2 + (\sum_{i=1}^{n} 0.5 i x_i)^2 + (\sum_{i=1}^{n} 0.5 i x_i)^4$	30	[-5,10]	-1
Periodic	$f(\mathbf{x}) = 1 + \sum_{i=1}^{d} \sin^2 (x_i) - 0.1 \exp\left(-\sum_{i=1}^{d} x_i^2\right)$	30	[-10,10]	0.9
Brown	$f(\mathbf{x}) = \sum_{i=1}^{d-1} (x_i^2)^{(x_{i+1}^2+1)} + (x_{i+1}^2)^{(x_i^2+1)}$	30	[-1,4]	0
Beale	$f(x,y) = (1.5 - x + xy)^2 + (2.25 - x + xy^2)^2 + (2.625 - x + xy^3)^2$	30	[-4.5,4.5]	0
Xin She Yang N.2	$f(\mathbf{x}) = \left(\sum_{i=1}^{d} \lvert x_i \rvert\right) \exp\left(-\sum_{i=1}^{d} \sin (x_i^2)\right)$	30	[-2π,2π]	0
Powell Singular	$f(\mathbf{x}) = \sum_{i=1}^{d/4} [(x_{4i-3} + 10 x_{4i-2})^2 + 5(x_{4i-1} - x_{4i})^2 + (x_{4i-2} - 2x_{4i-1})^4 + 10(x_{4i-3} - x_{4i})^4]$	30	[-100,100]	0
Stepint	$f(X) = 25 + \sum_{i=1}^{n} \lfloor x_i \rfloor$	30	[-100,100]	0
Trid	$f(\mathbf{x}) = \sum_{i=1}^{d} (x_i - 1)^2 - \sum_{i=2}^{d} (x_i - 1 x_{i-1})$	30	[-10,10]	0
Matyas	$f(x,y) = 0.26(x^2 + y^2) - 0.48xy$	30	[-10,10]	0

Holder table	$f(X)$ $= -\left\lvert \sin(x_1)\cos(x_2)e^{\left\lvert 1-\frac{\sqrt{x_1^2+x_2^2}}{\pi}\right\rvert}\right\rvert$	30	[-10,10]	0
Sumpow	$f(\mathbf{x}) = \sum_{i=1}^{d} \lvert x_i \rvert^{i+1}$	30	[-100,100]	0
levy13	$f(x,y) = \sin^2(3\pi x) + (x - 1)^2(1 + \sin^2(3\pi y)) + (y - 1)^2(1 + \sin^2(2\pi y))$	2	[-10,10]	0
Sum Squares	$f = \sum_{i=1}^{D} (ix_i)^2$	30	[-10,10]	0

Table 3- Benchmark function

6.2 Manually Results of Sphere function

But when it is manually done, the sphere function that has been chosen by iteration got the best score of 2.7595. Chosen Parameter, p = 0.03, $\beta = 3, w = 0.08$, lb= -10, ub= 10, Maxt=1, pop_size=3.

No of Gorillas	X(t)	F(X(t))	GX'(t+1)	F(GX'(t+1))	GX''(t+1)	F(GX''(t+1))
1	[4.1702, 3.0233, 1.8626]	30.0005	[4.1702, 3.0233, 1.8626]	30.0005	[4.1702, 3.0233, 1.8626]	30.0005
2	[7.2032, 1.4676, 3.4556]	65.9817	[1.4731 0.3001 0.7067]	2.7595	[1.4731 0.3001 0.7067]	2.7595
3	[0.0011, 0.9234, 3.9677]	16.5951	[0.0011, 0.9234, 3.9677]	16.5951	[0.0011, 0.9234, 3.9677]	16.5951

Table 4- Manually result

And $X_{silverback}$ = [1.4731 0.3001 0.7067] (Best solution) with silverback score 2.7595.

6.3 Experimental result of AGTO using MATLAB and PYTHON and MGTO using PYTHON

The execution time and best score are compared on these two platforms by using the Gorilla Troops Optimizer. Also, the best score is compared between AGTO and MGTO. Then ten runs were completed. The detailed comparative analysis is listed below. And JUPYTER NOTEBOOK and SPYDER 3.9 were the first to be compared to MATLAB. an open-source

programming language that has some advantages compared to other languages. It is easy to implement as well as easy to understand. It has some inbuilt libraries also. Python supports OOP (object-oriented programming) as well. Some well-known Python libraries are used in the AGTO Python code, such as Numpy, which is used for mathematical operations, Matplotlib, which is used for various curve plotting; and Random, which is used to randomly initialise the random position of a candidate solution (Gorilla candidate position vector). Aside from that, the optimise library is used along with the time library to determine the execution time of the entire program. By using ten benchmark functions, the best solution has been computed that has a silverback score and its positions. Chosen parameters are listed below.

Maximum no of iteration	Upper bound	Lower bound	p	w	Beta	Population size	Variables no
100	10	-10	0.03	0.8	3	30	3

Table 5- Chosen parameter

6.3.1 Matlab result, taken run =1

Benchmark function / Obtain result	Sphere function	Ackley function	Rosenbrock Function	Rastrigin Function	Schwefel function
Best Score	1.3962e-93	-1.7634	3.2547e-13	0	5.6947e-49
Best position	[-0.1358 0.3107 -0.1569]	[-0.1841 0.0782 -0.0589]	[1.000 1.000 1.0000]	[0.0683 -0.2148 0.2723]	[5.2392 5.2392 5.2392]
Execution time	0.568507 seconds	0.525480 seconds	0.568498 seconds	0.581671 seconds.	0.530179 seconds

Table 6- Matlab result 1

6.3.2 Best, Worst, Median, Mean and Std. value of Benchmark function using MATLAB, run = 10

Function	Population size	Dimension	Iterations No	Best value	Worst value	Median value	Mean value	Std.	Execution time
Sphere	30	3	100	0	4.4082e-86	2.172e-92	4.4359e-87	1.3931e-86	0.67 seconds
Ackley	30	3	100	0	-1.7634	-1.7634	-1.7634	4.6811e-16	0.69 seconds
Rosenbrock	30	3	100	0	1.1768e-10	9.817e-15	1.2481e-11	3.7008e-11	0.67 seconds
Rastrigin	30	3	100	0	0	0	0	0	0.66 seconds
Schwefel 2.22	30	3	100	0	1.3505e-42	2.1508e-45	1.7134e-43	4.2441e-43	0.92 seconds
Step:	30	3	100	0	5.4328e-	1.6983	6.3874	0	0.56

US					22	e-25	e-23		seconds
Dixon & Price	30	3	100	0	2.1243e-18	2.1243 e-18	2.1243 e-18	0	0.57 seconds
Sum squares	30	3	100	0	1.5683e-86	1.9612 e-91	1.6477 e-87	0	0.53 seconds
Griewank	30	3	100	0	0	0	0	0	0.52 seconds
Booth	30	3	100	0	0	0	0	0	0.995839 seconds
Quartic	30	3	100	0.0000	0.0011037	0.0006 1034	0.0005 8192	0.0003 1458	1.09 seconds
Quadratic	30	3	100	0.0003	0.0001907899	0.0001	0.0002	74.3943	1.21seconds
Zakharov	30	3	100	0.0000	9.1486e-86	4.9044 e-88	1.4429 e-86	3.0771 e-86	1.00seconds
Periodic	30	3	100	0.9	0.9	0.9	0.9	1.1703 e-16	2.79 seconds
Brown	30	3	100	0.0000	2.395e-85	9.646e-91	2.3962 e-86	7.5733 e-86	1.24 seconds
Beale	30	3	100	0.0000	3.2359e-27	3.9406 e-29	5.9096 e-28	1.1865 e-27	1.01seconds
Xin She Yang N.2	30	3	100	0.0000	9.6994e-36	2.1532 e-40	9.8261 e-37	3.0629 e-36	1.09 seconds.
Xin She Yang N.4	30	3	100	-1	-1	-1	-1	0	1.27 seconds
Powell Singular	30	3	100	0	0	0	0	0	1.17 seconds
Stepint	30	3	100	-5.0000	-5	-5	-5	0	1.115566 seconds
Trid	30	3	100	-7.0000	-7	-7	-7	-7	1.05 seconds
Matyas	30	3	100	0.0000	5.1226e-81	1.7498 e-93	5.123e-82	1.6199 e-81	1.02 seconds
Holdertable	30	3	100	-19.2085	-19.2085	-19.2085	-19.2085	2.3685 e-15	1.04 seconds
Sumpow	30	3	100	0.0000	7.4437e-103	8.3894 e-107	8.9059 e-104	2.3279 e-103	1.19 seconds
levy13	30	3	100	0.00	1.3498e-	1.3498	1.3498	0	1.14

				00	31	e-31	e-31		seconds
Sum Squares	30	3	100	0.0000	5.4067e-84	1.5467e-89	5.4423e-85	1.7085e-84	1.08 seconds

Table 7- Matlab result 2

6.3.3 Python result, taken run =1

Benchmark function / Obtain result	**Sphere function**	**Ackley function**	**Rosenbrock Function**	**Rastrigin Function**	**Schwefel function**
Best Score	5.643786374245626	7.680405931435092	6.216985969084413	21.12256935259996	8.819334724316416
Best position	[-0.10935595 1.6833396 -1.67278075]	[2.64744421 0.69371122 -0.62286713]	[1.72012837 -2.17059875 2.86536901]	[0.07593651 8.71199648 -8.89904316]	[4.52957593 3.43958932 0.05127718]
Execution time	0.48864197731018066 Seconds	0.4109630584716797 seconds	0.41329026222229004 seconds	0.36345386505126953 seconds	0.45641446113586426 seconds

Table 8- Python result 1

6.3.4 Best, Worst, Median, Mean and Std. value of Benchmark function using PYTHON, run = 10

Function	**Population size**	**Dimension**	**Iterations No**	**Best value**	**Worst value**	**Median value**	**Mean value**	**Std.**	**Execution time**
Sphere	30	3	100	0.0	0.0026	0.0	0.00002	0.0007	29.94 seconds
Ackley	30	3	100	0.0	1.1964	0.0	0.11960	0.3589	101.44 seconds
Rosenbrock	30	3	100	0.0	0.3921	0.0	0.03920	1.1679	11.08 seconds
Rastrigin	30	3	100	0.0	20.00000	0.0	2.00000	6.00000	14.76 seconds
Schwefel 2.22	30	3	100	0.0	0.0008	0.0	8.906721466798 42e-05	0.0002	39.31 seconds

Step: US	30	3	100	0.0	0.0023	0.0	0.0002	0	16.09 seconds
Dixon & Price	30	3	100	0.0	0.0019	0.0	0.0002	0	21.69 seconds
Sum squares	30	3	100	0.0	2.3741735320932275e-05	0.0	7.125205962796 83e-06	0	18.48 seconds
Griewank	30	3	100	0.0	1.6558912764863543e-05	0.0	1.655891276486 3543e-06	4.967673829459063e-06	6.73 seconds
Booth	30	3	100	0.0	0.0908	0.0	0.0090	0.02726	6.90 seconds
Quartic	30	3	100	0.0	0.1981	0.0	0.0198	0.05945	23.16 seconds
Quadratic	30	3	100	0.0	0.0795	0.0	0.00795	0.02385	23.24 seconds
Zakharov	30	3	100	0.0	0.117	0.0	0.01174	0.03524	10.269315481185913 seconds
Periodic	30	3	100	0.9	0.9000	0.9000	0.09000	0.270000	73.44 seconds
Brown	30	3	100	0.0	0.009	0.0	0.00099	0.002977	38.51 seconds
Beale	30	3	100	0.0	5.648	0.0	0.56489	1.69467	9.33 seconds
Xin She Yang N.2	30	3	100	0.0	0.002	0.0	0.00019	0.00058	18.59 seconds
Powell	30	3	100	0.0	0.000	0.0	6.07	0.000	14.25

Singular					6		6052 7772 5874 3e-0 5	18	seconds
Stepint	30	3	100	0.0	4.0	0.0	0.4	1.200 0	41.25 seconds
Trid	30	3	100	-10. 05	0.0	0.0	-1.00 5	3.015 9	9.89 seconds
Matyas	30	3	100	0.0	3.051 3144 0175 5933 e-07	0.0	3.05 1314 4017 5593 3e-0 8	9.153 9432 0526 7799 e-08	12.57 seconds
Holder table	30	3	100	0.0	1.007	0.0	0.10 07	0.302 17	51.85 seconds
Sumpow	30	3	100	0.0	0.0	0.0	0.0	0.0	3.93 seconds
levy13	30	3	100	0.0	0.161	0.0	0.01 61	0.048	57.13 seconds
Sum Squares	30	3	100	0.0	4.756 2506 9527 0320 5e-0 6	0.0	4.75 6250 6952 7032 04e- 07	1.426 8752 0858 1096 3e-06	6.88 seconds

Table 9- Python result 2

So, after comparison with ten benchmarks function every benchmark test function gives the optimal or near optimal value. By taking 10 run and 10 function mean median, best, worst and standard value has been computed also. In nutshell, by increasing runs from 1 to 10 PYTHO N takes least amount of time compare to MATLAB. And ten benchmark functions gave best score that is 0 or near 0 by using 100 iterations and 10 runs only. So, future propose work PYTHON language will be used for critical gene selection purpose from microarray datasets.

6.3.5 Best, Worst, Median, Mean and Std. value of Benchmark function by run 50 and D=10

Benchmark Function Name	Upper and lower bound	Runs	Population size	Dimension	Iterations No (Maxt)	Best value	Worst value	Median value	Mean value	Std. value	Execution time
Sphere	[-100,100]	50	30	10	100	0	6.8219e-74	1.2791e-83	2.2144e-75	1.0406e-74	4.190691 seconds.
					300	0	5.4158e-239	6.0636e-249	1.8257e-240	0	7.178457 seconds
					500	0	0	0	0	0	13.137892 seconds
					1000	0	0	0	0	0	21.303799 seconds
Ackley	[-100,100]	50	30	10	100	-125.9057	-125.6949	-125.6952	-128.9515	5.9729	3.753510 seconds
					300	-133.8541	-125.6949	-143.7182	-137.9773	8.4211	7.8214

											59 seconds
					500	-145.6949	-125.6949	-145.6949	-140.8443	7.8197	11.353966 seconds.
					1000	-145.6949	-137.1749	-145.6949	-145.5245	1.2049	29.612055 seconds
Rosenbrock	[-100,100]	50	30	10	100	4.5678e-07	7.3806	6.3543	5.2067	2.6325	3.045754 seconds
					300	1.3265e-08	4.7859	3.2029	2.4348	1.7735	6.700962 seconds
					500	0	2.7694	0.85026	0.90405	0.9256	10.023338 seconds
					1000	1.2091e-10	0.42628	0.00030743	0.065387	0.11986	30.176280 seconds
Rastrigin	[-100,100]	50	30	10	100	0	0	0	0	0	2.985380 seconds

					300	0	0	0	0	0	7.109875 seconds
					500	0	0	0	0	0	11.442250 seconds
					1000	0	0	0	0	0	22.989868 seconds
Schwefel 2.22	[-100,100]	50	30	10	100	0	9.066e-37	1.8618e-41	3.7439e-38	1.6459e-37	2.867743 seconds
					300	0	1.2193e-117	7.4154e-125	5.3001e-202	1.723e-118	11.919561 seconds
					500	0	2.2401e-200	2.5342e-209		0	11.919561 seconds
					1000	0	0	0	0	0	22.120704 seconds
Step : US	[-100,100]	50	30	10	100	0	5.4328e-22	1.6983e-25	6.3874e-23	0	6.3874e
					300	0	9.2276e-13	1.3285e-15	3.632e-14	1.3637e-13	6.911996 sec

											onds
					500	0	1.3262e-18	1.6381e-23	2.7935e-20	1.874e-19	12.513371 seconds
					1000	0	1.2289e-24	5.2123e-30	4.8793e-26	2.3492e-25	21.232232 seconds
Dixon & Price	[-100,100]	50	30	10	100	1.7744e-05	0.66669	0.45686	0.35642	0.31701	3.113238 seconds
					300	0	0.66667	3.266e-07	0.053815	0.18258	7.040205 seconds.
					500	0	0.66667	6.8556e-13	0.013333	0.094281	11.204537 seconds
					1000	0	1.1072e-16	3.6278e-21	3.2332e-18	1.6766e-17	21.017895 seconds
Sum squares	[-100,100]	50	30	10	100	7.2639e-90	1.1181e-75	5.7499e-82-	4.402e-77	2.1452e-76	2.982068 seconds.
					300	1.1137e-269	1.6058e-232	3.9699e-248	3.2957e-234	0	7.7291

											44 seconds
					500	0	0	0	0	0	11.267667 seconds.
					1000	0	0	0	0	0	11.267667 seconds.
Griewank	[-100,100]	50	30	10	100	0	0	0	0	0	2.984343 seconds
					300	0	0	0	0	0	6.461281 seconds
					500	0	0	0	0	0	11.888540 seconds
					1000	0	0	0	0	0	21.985548 seconds
Booth	[-100,100]	50	30	10	100	0	7.0997e-30	0	1.7355e-31	1.0116e-30	3.342349 seconds
					300	0	0	0	0	0	6.711347

											seconds.
					500	0	0	0	0	0	11.578858 seconds
					1000	0	0	0	0	0	22.047211 seconds
Quartic	[-100,100]	50	30	10	100	1.8776e-06	0.0019811	0.00040563	0.0005343	0.0004197	3.785807 seconds
					300	3.5038e-07	0.00079649	0.00013282	0.0001843 2	0.0001833	9.366965 seconds
					500	7.0205e-07	0.00042073	9.7941e-05	0.0001098 2	8.6641e-05	14.846080 seconds
					1000	0	0	0	0	0	31.079959 seconds
Quadratic	[-100,100]	50	30	10	100	0.0001450926+152.7588i	2.935853934e-06+100000i	7.94636758e-05+66979.7432i	0.0001895 22546+63288.3772i	33648.7738	4.333412 seconds

					300	8.6445e-05+0.41068i	2.687091415e-06+100000i	5.99702155e-05+61480.8408i	5.28371731e-05+56665.421i	37083.635	8.320311 seconds
					500	2.4818821e-05+3649.4987i	1.49976801e-07+100000i	7.75267577e-06+7316 1.5447i	2.08010499e-05+66489.6126i	31062.1634	12.634103 seconds
					1000	2.2623558e-06+1017.494i	6.036452153e-07+100000i	1.32788064e-05+75336.6382i	1.75506643e-05+64820.8886i	34824.7183	34.866997 seconds.
Zakharov	[-100,100]	50	30	10	100	2601e-82	3.82e-64	5.5109e-228	8.1333e-219	0	9.588292 seconds
					300	6.9236e-240	3.0724e-217	5.5109e-228			9.588292 seconds
					500	0	0	0	0	0	17.626733 seconds
					1000	0	0	0	0	0	20.740368 seconds
Periodic	[-100,100]	50	30	10	100	0.9	1	0.9	0.90256	0.014437	3.539750 seconds
					300	0.9	1	0.9	0.902	0.01	8.4

										4142	35167 seconds.
					500	0.9	0.9	0.9	0.9	0.014142	12.692839 seconds.
					1000	0.9	0.9	0.9	0.9	2.243e-16	24.964191 seconds
Brown	[-100,100]	50	30	10	100	0	2.395e-85	9.646e-10	2.3962e-86	7.5733	1.2359 4
					300	0	1.395e-95	9.646e-20	2.3962e-90	6.5733	11.235 94
					500	0	2.395e-104	10.646e-25	1.3962e-98	7.5733	15.235 94
					1000	0	0	0	0	0	32.964191 seconds
Beale	[-100,100]	50	30	10	100	0	3.9221e-21	4.1307e-30	7.8468e-23	5.5467e-22	3.2252 85 seconds
					300	0	0	0	0	0	7.4405 69 seconds.
					500	0	0	0	0	0	10.388

											632 seconds
					1000	0	0	0	0	0	19.203320 seconds
Xin She Yang N.2	[-100,100]	50	30	10	100	0.00056607	0.00063697	0.00056608	0.0005675 4	1.002e-05	3.534237 seconds
					300	0.00056607	0.00078585	0.00056607	0.0005727 5	3.4754e-05	7.330879 seconds
					500	2.7564e-08	0.00056607	0.00056607	0.0005267	0.00013722	11.769731 seconds
					1000	0	00	0	0	0	28.943888 seconds
Xin She Yang N.4	[-100,100]	50	30	10	100	-1	4.2057e-11	-1	-0.97943	0.14135	3.666171 seconds
					300	-1	-1	-1	-1	0	7.945370 seconds.

					500	-1	-1	-1	-1	0	16.217620 seconds
					1000	-1	-1	-1	-1	0	24.637388 seconds.
Powell Singular	[-100,100]	50	30	10	100	1.0243e-84	2.0916e-60	1.0545e-76	5.3455e-62	3.0534e-61	3.182120 seconds
					300	3.3071e-242	6.3958e-23	1.1543e-67	1.2792e-24	9.045e-24	7.079915 seconds
					500	0	3.0622e-29	2.4145e-66	6.2763e-31	4.3293e-30	11.108785 seconds
					1000	1.3901e-237	6.3299e-31	2.7019e-62	1.867e-32	9.1982e-32	20.907967 seconds
Stepint	[-100,100]	50	30	10	100	-975	-975	-975	-975	0	2.982725 seconds
					300	-975	-975	-975	-975	0	7.003971 sec

											onds
					500	-975	-975	-975	-975	0	10.137557 seconds
					1000	-975	-975	-975	-975	0	19.718070 seconds
Trid	[-100,100]	50	30	10	100	0.01251	0.41251	0.51251	0.61251	0.331251	2.960592 seconds
					300	2.9164e-10	1.9164e-05	1.9164e-05	1.9164e-05	1.9164e-05	6.538663 seconds
					500	3.7741e-09	8.2741e-09	3.2741e-10	3.2751e-08	3.2741e-09	10.436283 seconds
					1000	0	0	0	0	0	20.604756 seconds
Matyas	[-100,100]	50	30	10	100	4.5533e-98	1.1867e-83	1.0223e-90	3.3726e-85	1.7068e-84	2.983224 seconds
					300	5.0995e-280	6.6345e-255	8.9873e-268	1.7382e-256	0	6.5219

											19 seconds
					500	0	0	0	0	0	10.572711 seconds
					1000	0	0	0	0	0	22.572711 seconds
Holdertable	[-100,100]	50	30	10	100	-9.13953218640773 1e+18	-9.13953218640769 1e+18	-9.13953218640772 1e+18	-9.139532186407721e+18	8417.4686	2.968614 seconds
					300	-9.13953218640773 4e+18	-9.13953218640771 6e+18	-9.13953218640772 8e+18	-9.139532186407728e+18	4119.4431	6.685103 seconds.
					500	-9.13953218640773 6e+18	-9.13953218640772 1e+18	-9.13953218640772 9e+18	-9.139532186407731e+18	3424.4608	10.315720 seconds
					1000	-9.13953218640773 5e+18	-9.13953218640772 8e+18	-9.13953218640773 1e+18	-9.139532186407731e+18	1699.6864	19.207292 seconds
Sumpow	[-100,100]	50	30	10	100	6.4432e-114	8.8104e-97	3.724e-105	2.2611e-98	1.27e-97	3.642011 seconds

					300	0	1.7002e-304	2.4782e-320	3.4451e-306	0	8.031961 seconds
					500	0	0	0	0	0	12.328210 seconds
					1000	0	0	0	0	0	20.328210 seconds
levy 13	[-100,100]	50	30	10	100	1.3498e-31	1.3498e-31	1.3498e-31	1.3498e-31	2.2118e-47	2.763957 seconds
					300	1.3498e-31	1.3498e-31	1.3498e-31	1.3498e-31	2.2118e-47	6.845222 seconds
					500	1.3498e-31	1.3498e-31	1.3498e-31	1.3498e-31	2.2118e-47	9.919841 seconds
					1000	1.3498e-31	1.3498e-31	1.3498e-31	1.3498e-31	2.2118e-47	20.064479 seconds
Sum Squares	[-100,1	50	30	10	100	1.7397e-90	7.5849e-74	4.9276e-82	1.5182e-75	1.0727e-74	2.952513

	00]										seconds
					300	9.8506e-263	4.9633e-236	4.0917e-248	1.3486e-237	0	6.201639 seconds
					500	0	0	0	0	0	12.962314 seconds
					1000	0	0	0	0	0	23.962714 seconds

Table 10- Python result 3

First, experimental result was done 100 iterations only. Then 100 iteration, 10 runs and 3 dimensional were taken. In this case I got the optimal or near optimal solutions. After that AGTO has been tested on 100, 300, 500 1nd 1000 iterations with 50 runs and 10 D problems. But in this some benchmark function did not give well result like Holder table and Quadratic benchmark function. But it is proved that the AGTO is suitable for the large dimension also. For some benchmark function it's not work well. Hence an improve and modified version of GTO will be used. Only 5 benchmark function has been tested on MGTO that is given bellow. Rest of function will be tested in future. And we will see how it is superior than AGTO.

6.3.6 MGTO result, taken run=1

Benchmark function / **Obtain result**	**Sphere function**	**Ackley function**	**Rosenbrock Function**	**Rastrigin Function**	**Schwefel function**
Best Score	17.326798396503644	11.247158601644335	10.688392735562644	22.116257138319348	5.6947e-49
Best position	[1.212611813.263797372.28122724]	[4.21104304 1.34402006 1.19716897]	[-0.47899742 1.65591665 1.01389467]	[0.91884999 -3.11292959 -8.90420554]	[5.2392 5.2392 5.2392]

Execution time	0.4081132411956787 Seconds	0.3608527183532715 Seconds	0.3656277656555176 Seconds	0.4256248474121094 Seconds	0.530179 seconds

Table 10- Python result 4

So, I got best fitness value of Schwefel function from Modified Gorilla troops optimizer compare to Artificial Gorilla Troops optimizer by taking run 1 only. Rest of function gave better value in AGTO.

6.4 Curve representation of benchmark function

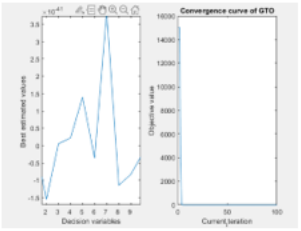


Figure 5- Sphere Function

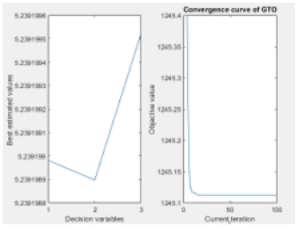


Figure 9 Schwefel function 2.22

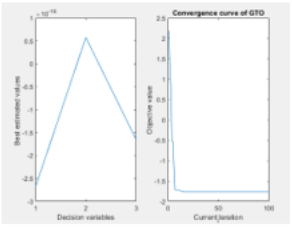


Figure 6- Ackley function

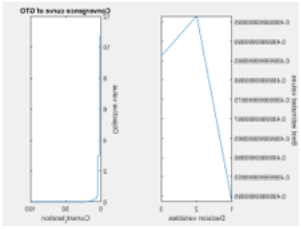


Figure 10- Step: US

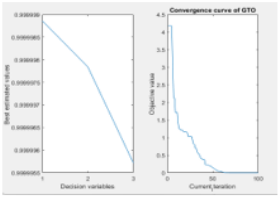


Figure 7-Rosenbrock Function

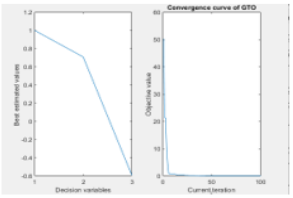


Figure 11- Dixon and Price

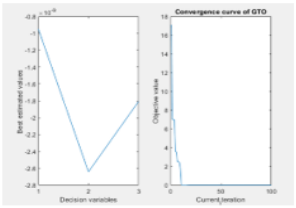


Figure- 8 Rastrigin Function

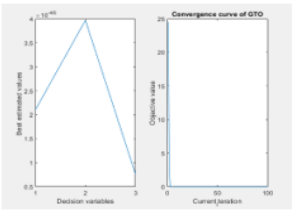


Figure 12- Sum squares

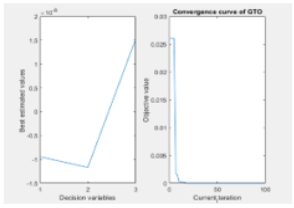


Figure 13- Griewank

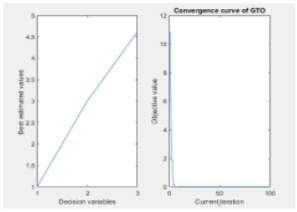


Figure 14- Booth

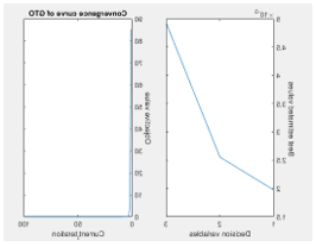


Figure 15- Quartic

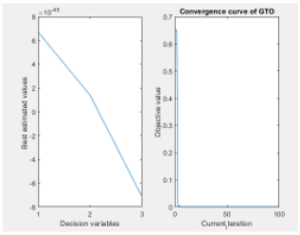


Figure 19- Brown

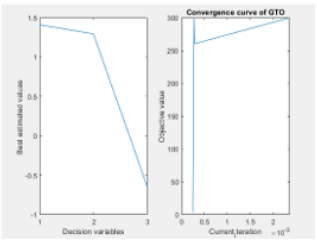


Figure 16- Quadratic

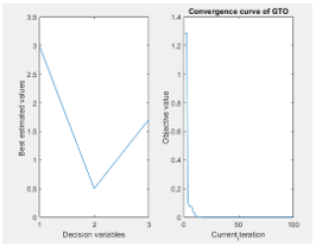


Figure 20- Beale

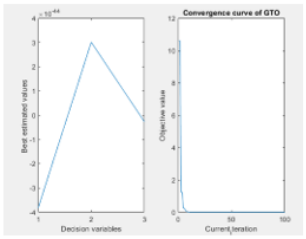


Figure 17- Zakharov

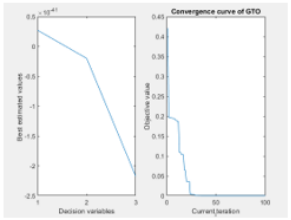


Figure 21- Xin She Yang N.2

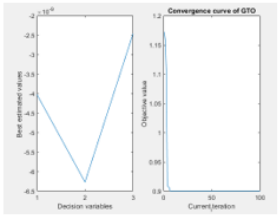


Figure 18- Periodic

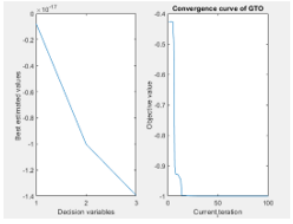


Figure 22- Xin She Yang N.4

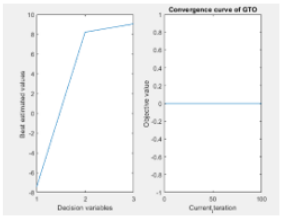


Figure 23- Powell Singular

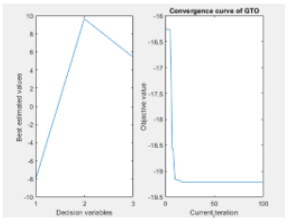


Figure 27- Holder table

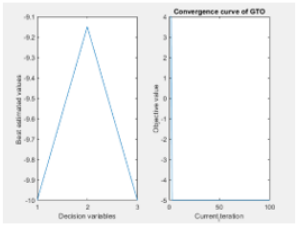


Figure 24- Stepint

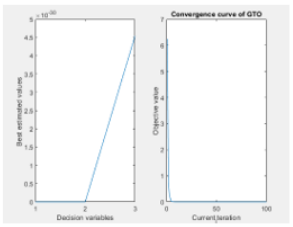


Figure 28- Sumpow

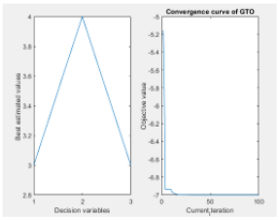


Figure 25- Trid

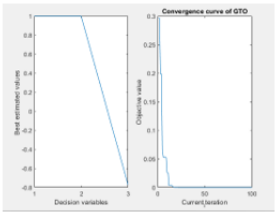


Figure 29- levy13

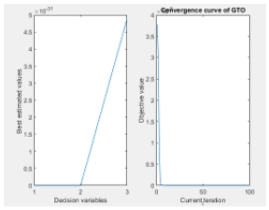


Figure 26- Matyas

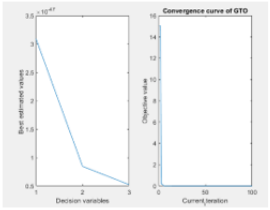


Figure 30- Sum Squares

Figure 7- Benchmark curve

7 Conclusion and future work

A new metaheuristic algorithm has been introduced that is the Gorilla Troops Optimizer (GTO), which is stimulated by the Gorilla group and their social behaviour with their intelligence in nature. The GTO algorithm has a unique mechanism for switching between the exploration (Diversification) and exploitation (Intensification) phases. Additionally, because this method employs a variety of processes, it has demonstrated good performance and can be utilised as a robust metaheuristic algorithm to address a variety of issues. Due to the fact that the experimental results of the numerous standard functions utilised in the GTO test suggest outstanding performance, it is necessary to conduct adequate exploration and exploitation operations in order to achieve excellent results for several of the standard benchmark test functions under experiment. The suggested approach is evaluated against 25 benchmark function standards that are both standard and diverse. The results of the analysis are given above, which are interpreted through the above table. The results are compared based on best fitness value as well as run time analysis. The GTO algorithm produces a more optimal solution with a higher

degree of convergence than its competitors. The experimental results indicate that the GTO approach is applicable to the real world for solving any real-time problem. So, it is concluded that this approach can also be used to handle situations involving multi-objective optimization problems. Meanwhile, because metaheuristic algorithms are used to handle an extensive variety of problems, GTO can be evaluated in the future and used to address recombination optimization problems and diversified problems with different anchors. It has adequate capacity for moving the entire optimization space without any delay. GTO is smart enough to improve gorillas position vector in half the iterations, which indicates an excellent ability that GTO has compared to other meta-heuristic optimization algorithms. The first gorilla has momentous movements in the early stages, but it only goes by sudden movements in some stages. These features suggest GTO has an excellent ability to explore in various search spaces. But in next stages, the range of fluctuations changed and become decreased. This signifies that GTO has an exploitative nature in promising areas. And also, GTO is very stable for the first guerrilla movement. But GTO also has a problem. In GTO, the exploitation phase is still suffering from slow convergence speed. And GTO can fall into premature convergence. As in GTO, a random number is used significantly, which can increase the step size of the optimization process. And that's why the new solution might be far from the best solution. It might involve the desired near-optimal solution but not be involved in the optimum solution. The equality between intensification and diversification might reduce the performance of the entire GTO algorithm, and hence, in some cases, higher exploration and/or exploitation operators are needed. But this gap can also be handled by MGTO, or modified Gorilla Troops Optimizer, which can give the desired optimum solution. In MGTO, the exploitation operator is replaced by another one. MGTO aids in exploring more promising search regions around the best solution. And one amazing thing is that other places in the promising search regions might be able to give the desired optimal solution as well as an optimal solution. In the current experiment, GTO is suitable for other benchmark functions that are not only unimodal but also multimodal and mixed benchmark functions. So, this algorithm will also be compared with other meta-heuristic algorithms like GWO, PO, GA, PSO, WO, etc., and GTO and MGTO will be applied in multi-objective functions as well as in bioinformatics problems for predicting biomarkers or critical genes from any high-dimensional genomic data.

8 References

[1] Bayat, A. (2002). Science, medicine, and the future: Bioinformatics. *BMJ: British Medical Journal*, *324*(7344), 1018.

[2] Markowetz, F. (2017). All biology is computational biology. *PLoS biology*, *15*(3), e2002050.

[3] Liao, Y., Xiao, H., Cheng, M., & Fan, X. (2020). Bioinformatics analysis reveals biomarkers with cancer stem cell characteristics in lung squamous cell carcinoma. *Frontiers in genetics*, *11*, 427.

[4] Masoudi-Sobhanzadeh, Y., Motieghader, H., Omidi, Y., & Masoudi-Nejad, A. (2021). A machine learning method based on the genetic and world competitive contests algorithms for selecting genes or features in biological applications. *Scientific Reports*, *11*(1), 1-19.

[5] Koenig, I. R., Fuchs, O., Hansen, G., von Mutius, E., & Kopp, M. V. (2017). What is precision medicine?. *European respiratory journal*, *50*(4).

[6] Abdollahzadeh, B., Soleimanian Gharehchopogh, F., & Mirjalili, S. (2021). Artificial gorilla troops optimizer: A new nature-inspired metaheuristic algorithm for global optimization problems. *International Journal of Intelligent Systems*, *36*(10), 5887-5958.

[7] Yang, X. S. (2011). Metaheuristic optimization. *Scholarpedia*, *6*(8), 11472.

[8] Abdel-Basset, M., Mohamed, R., & Chang, V. (2021). An Efficient Parameter Estimation Algorithm for Proton Exchange Membrane Fuel Cells. *Energies*, *14*(21), 7115.

[9] Beheshti, Z., & Shamsuddin, S. M. H. (2013). A review of population-based meta-heuristic algorithms. *Int. J. Adv. Soft Comput. Appl*, *5*(1), 1-35.

[10] Dorigo, M., Maniezzo, V., & Colorni, A. (1996). Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *26*(1), 29-41.

[11] Askari, Q., Younas, I., & Saeed, M. (2020). Political Optimizer: A novel socio-inspired meta-heuristic for global optimization. *Knowledge-Based Systems*, *195*, 105709.

[12] Kumar, A., Pant, S., & Ram, M. (2017). System reliability optimization using gray wolf optimizer algorithm. *Quality and Reliability Engineering International*, *33*(7), 1327-1335.

[13] Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *science*, *220*(4598), 671-680.

[14] Rana, N., Abd Latiff, M. S., & Chiroma, H. (2020). Whale optimization algorithm: a systematic review of contemporary applications, modifications and developments. *Neural Computing and Applications*, 1-33.

[15] Abdollahzadeh, B., Soleimanian Gharehchopogh, F., & Mirjalili, S. (2021). Artificial gorilla troops optimizer: A new nature-inspired metaheuristic algorithm for global optimization problems. *International Journal of Intelligent Systems*, *36*(10), 5887-5958.

[16] Ginidi, A., Ghoneim, S. M., Elsayed, A., El-Sehiemy, R., Shaheen, A., & El-Fergany, A. (2021). Gorilla troops optimizer for electrically based single and double-diode models of solar photovoltaic systems. *Sustainability*, *13*(16), 9459.

[17] Krishna, R. K., & Ramanjaneyulu, B. S. Gorilla Optimization Based Clustering and Fittest Node Routing Technique for Improving the Lifetime of Wireless Sensor Network.

[18] Abdel-Basset, M., Mohamed, R., & Chang, V. (2021). An Efficient Parameter Estimation Algorithm for Proton Exchange Membrane Fuel Cells. *Energies*, *14*(21), 7115.

[19] Amjad, M. K., Butt, S. I., Kousar, R., Ahmad, R., Agha, M. H., Faping, Z., ... & Asgher, U. (2018). Recent research trends in genetic algorithm based flexible job shop scheduling problems. *Mathematical Problems in Engineering*, *2018*.

[20] Garg, J., Khatri, P., & Ahmad, A. (2011). Economic load dispatch using genetic algorithm. *Int J Adv Res Comput Sci*, *2*(3), 451-454.

[21] Sharma, S., & Singh, D. S. (2014). Heart Disease Diagnosis using Genetic and Particle Swarm Optimization. *International Journal of Engineering Research & Technology*, *3*(8), 1499-1503.

[22] Sarkar, J. P., Saha, I., Sarkar, A., & Maulik, U. (2021). Machine learning integrated ensemble of feature selection methods followed by survival analysis for predicting breast cancer subtype specific miRNA biomarkers. *Computers in Biology and Medicine*, *131*, 104244.

[23] Liao, Y., Xiao, H., Cheng, M., & Fan, X. (2020). Bioinformatics analysis reveals biomarkers with cancer stem cell characteristics in lung squamous cell carcinoma. *Frontiers in genetics*, *11*, 427.

[24] Gomaa, W. E. (2011, December). Modelling gene regulatory networks: A survey. In *2011 9th IEEE/ACS International Conference on Computer Systems and Applications (AICCSA)* (pp. 204-208). IEEE.

[25] Patowary, P., & Bhattacharyya, D. K. (2021). PD_BiBIM: Biclustering-based biomarker identification in ESCC microarray data. *Journal of Biosciences*, *46*(3), 1-18.

[26] Bandaru, S., & Deb, K. (2016). Metaheuristic techniques. *Decision sciences*, 693-750.

[27] Sarkar, J. P., Saha, I., Sarkar, A., & Maulik, U. (2021). Machine learning integrated ensemble of feature selection methods followed by survival analysis for predicting breast cancer subtype specific miRNA biomarkers. *Computers in Biology and Medicine*, *131*, 104244.

[28] Patowary, P., & Bhattacharyya, D. K. (2021). PD_BiBIM: Biclustering-based biomarker identification in ESCC microarray data. *Journal of Biosciences*, *46*(3), 1-18.

[29] Giannos, P., Kechagias, K. S., & Gal, A. (2021). Identification of Prognostic Gene Biomarkers in Non-Small Cell Lung Cancer Progression by Integrated Bioinformatics Analysis. *Biology*, *10*(11), 1200.

[30] Debata, P. P., & Mohapatra, P. (2021). Identification of significant bio-markers from high-dimensional cancerous data employing a modified multi-objective meta-heuristic algorithm. *Journal of King Saud University-Computer and Information Sciences*.

[31] Popovic, D., Sifrim, A., Pavlopoulos, G. A., Moreau, Y., & De Moor, B. (2012, November). A simple genetic algorithm for biomarker mining. In *IAPR International Conference on Pattern Recognition in Bioinformatics* (pp. 222-232). Springer, Berlin, Heidelberg.

[32] Saraç, E., & Özel, S. A. (2014). An ant colony optimization based feature selection for web page classification. *The Scientific World Journal, 2014*.