Algorithms for Intelligent Systems
Series Editors: Jagdish Chand Bansal · Kusum Deep · Atulya K. Nagar

Seyedali Mirjalili
Hossam Faris
Ibrahim Aljarah *Editors*

Evolutionary Machine Learning Techniques

Algorithms and Applications

Springer

Algorithms for Intelligent Systems

This book series publishes research on the analysis and development of algorithms for intelligent systems with their applications to various real world problems. It covers research related to autonomous agents, multi-agent systems, behavioral modeling, reinforcement learning, game theory, mechanism design, machine learning, meta-heuristic search, optimization, planning and scheduling, artificial neural networks, evolutionary computation, swarm intelligence and other algorithms for intelligent systems.

The book series includes recent advancements, modification and applications of the artificial neural networks, evolutionary computation, swarm intelligence, artificial immune systems, fuzzy system, autonomous and multi agent systems, machine learning and other intelligent systems related areas. The material will be beneficial for the graduate students, post-graduate students as well as the researchers who want a broader view of advances in algorithms for intelligent systems. The contents will also be useful to the researchers from other fields who have no knowledge of the power of intelligent systems, e.g. the researchers in the field of bioinformatics, biochemists, mechanical and chemical engineers, economists, musicians and medical practitioners.

The series publishes monographs, edited volumes, advanced textbooks and selected proceedings.

More information about this series at http://www.springer.com/series/16171

Seyedali Mirjalili · Hossam Faris ·
Ibrahim Aljarah
Editors

Evolutionary Machine Learning Techniques

Algorithms and Applications

Springer

Editors
Seyedali Mirjalili
Torrens University Australia
Brisbane, QLD, Australia

Griffith University
Brisbane, QLD, Australia

Ibrahim Aljarah
King Abdullah II School for Information Technology
The University of Jordan
Amman, Jordan

Hossam Faris
King Abdullah II School for Information Technology
The University of Jordan
Amman, Jordan

ISSN 2524-7565 ISSN 2524-7573 (electronic)
Algorithms for Intelligent Systems
ISBN 978-981-32-9989-4 ISBN 978-981-32-9990-0 (eBook)
https://doi.org/10.1007/978-981-32-9990-0

This Springer imprint is published by the registered company Springer Nature Singapore Pte Ltd.
The registered company address is: 152 Beach Road, #21-01/04 Gateway East, Singapore 189721, Singapore

Preface

This book provides an in-depth analysis of the current evolutionary machine learning techniques. Discussing the most highly regarded methods for classification, clustering, regression, and prediction, it includes techniques such as support vector machines, feature selection, artificial neural networks including feed-forward neural networks, and multi-layer perceptron.

The book includes essential definitions, literature reviews, and the training algorithms for machine learning using classical and modern optimization techniques. It also investigates the pros and cons of classical training algorithms. It features a range of proven and recent nature-inspired optimization algorithms used to train different types of artificial neural networks, including genetic algorithm, particle swarm optimization, grey wolf optimizer, whale optimization algorithm, ant lion optimizer, salp swarm algorithm, moth flame optimization, since cosine algorithm, dragonfly algorithm, grasshopper optimizer, multiverse optimizer, and Harris hawks optimizer.

Brisbane, Australia — Prof. Seyedali Mirjalili
Amman, Jordan — Prof. Hossam Faris
Amman, Jordan — Prof. Ibrahim Aljarah
July 2019

Contents

Feature Selection

About the Editors

Seyedali Mirjalili is an Associate Professor of Artificial Intelligence at Torrens University Australia, and internationally recognised for his advances in nature-inspired artificial intelligence (AI) techniques. He is the author of five books, 100 journal articles, 20 conference papers, and 20 book chapters. With over 12000 citations and H-index of 45, he is one of the most influential AI researchers in the world. From Google Scholar metrics, he is globally the most cited researcher in Robust Optimisation using AI techniques. As an IEEE senior member, he has been the keynote speaker of several international conferences and is serving as an associate editor of top AI journals including Applied Soft Computing, IEEE Access, Advances in Engineering Software, and Applied Intelligence.

Hossam Faris is a Professor in the Information Technology Department at King Abdullah II School for Information Technology at The University of Jordan, Jordan. Hossam Faris received his B.A. and M.Sc. degrees in computer science from the Yarmouk University and Al-Balqa' Applied University in 2004 and 2008, respectively, in Jordan. He was awarded a full-time competition-based scholarship from the Italian Ministry of Education and Research to peruse his Ph.D. degrees in e-Business at the University of Salento, Italy, where he obtained his Ph.D. degree in 2011. In 2016, he worked as a postdoctoral researcher with the GeNeura team at the Information and Communication Technologies Research Center (CITIC), University of Granada, Spain. His research interests include applied computational intelligence, evolutionary computation, knowledge systems, data mining, semantic web, and ontologies.

Ibrahim Aljarah is an Associate Professor of BIG Data Mining and Computational Intelligence at The University of Jordan—Department of Information Technology, Jordan. Currently, he is the Director Assistant to International Affairs Unit at The University of Jordan. He obtained the bachelor degree in computer science from the Yarmouk University, Jordan, 2003. He also obtained his master degree in computer science and information systems from the Jordan University of Science and Technology, Jordan, in 2006. He participated in many conferences in the fields of data

mining, machine learning, and big data such as CEC, GECCO, NTIT, CSIT, IEEE NABIC, CASON, and BigData Congress. Furthermore, he contributed in many projects in USA such as Vehicle Class Detection System (VCDS), Pavement Analysis Via Vehicle Electronic Telemetry (PAVVET), and Farm Cloud Storage System (CSS) projects. He has published more than 35 papers in refereed international conferences and journals. His research focuses on data mining, machine learning, big data, MapReduce, Hadoop, swarm intelligence, evolutionary computation, social network analysis (SNA), and large-scale distributed algorithms.

Introduction to Evolutionary Machine Learning Techniques

Seyedali Mirjalili, Hossam Faris and Ibrahim Aljarah

Abstract This section first provides an overview of the machine learning field in artificial intelligence (AI). The most well-regarded classes of methods in AI are discussed to show where AI optimization algorithms and machine learning techniques fit in. Different types of learning are briefly covered as well including supervised, unsupervised, and reinforcement techniques. The last part of this chapter includes discussions on evolutionary machine learning, which is the focus of this book.

Keywords Machine learning · Artificial Intelligence · Neural network · Support vector machine · Feature selection · Supervised learning · Unsupervised learning · Evolutionary algorithms · Python · Optimization · Reinforcement learning · Classification · Regression · Clustering · Dataset

1 Introduction

The field of artificial intelligence (AI) has become incredibly popular in the last decade. In the past five years, leading information companies largely invested in this area as reliable solutions to solve business problems in a wide range of industries. Governments have increased funding for AI research centres across the globe as well. AI is a broad field and can be divided into several branches:

1. **Search methods**: This branch of AI includes problem-solving methods to solve computationally expensive problems. An example is finding an optimal path in GPS.
2. **Machine learning**: This branch of AI includes methods that make a computer capable of learning from experience. An example is getting a list of recommended videos based on the history of videos that you watched in a video-based platform.

S. Mirjalili (✉)
Torrens University Australia, Brisbane, QLD 4006, Australia
e-mail: ali.mirjalili@gmail.com

Griffith University, Brisbane, QLD 4111, Australia

H. Faris · I. Aljarah
King Abdullah II School for Information Technology, The University of Jordan, Amman, Jordan

S. Mirjalili et al. (eds.), *Evolutionary Machine Learning Techniques*,
Algorithms for Intelligent Systems, https://doi.org/10.1007/978-981-32-9990-0_1

3. **Knowledge representation and reasoning**: This branch of AI allows representing knowledge and inference in a computer. In other words, AI methods in this class are able to make logical decisions based on a given set of knowledge rules. An example is a knowledge-based system that is used as a chatbot in customer service departments.
4. **Machine vision**: This branch of AI includes techniques that allow computer to see and understand the world. In other words, the focus is on the perception, processing and analysing images, videos, or any other high-dimensional computer inputs (e.g. depth images and heat map images). An example is a hand gesture recognition or face detection system in verification systems.
5. **Natural Language Processing**: In this branch of AI, computers are used to understand and/or genera a natural language. For instance, a telephone banking system can assist customer by understanding the sentences and generating proper responses.
6. **Robotics**: In this branch of AI, robots are used to intelligently sense, plan, and act. AI algorithms are mostly used in the planning phase of a robot. An example is a self-driving car that reads inputs from proximity sensors/cameras, plans, and uses wheel, engine, brake, etc., to avoid obstacles.

It is hard to draw a line between these classes since they are highly overlapped. For instance, a search method in the first class might be used for pruning a knowledge tree in the second class or training a neural network in the fourth class.

The following section provides more details of search methods and machine learning branches since they are the main focus of this book.

2 Search Methods in AI

Search methods were of the first AI techniques that became popular in solving challenging real-world problems. Since the 1940s, a large number of search algorithms have been proposed to solve search problems. Such search methods can be categorized as uninformed and informed. In the former category, an algorithm blindly searches a search space without knowing how 'close' or 'far' the final solution is. Some of the uninformed search algorithms are: brute-force search, breadth-first search, depth-first search, and bidirectional search. Such search algorithms are complete, which means that they eventually find the best solution for a given search problem. However, we have to provide them with enough computing and storage resources. For computationally expensive problems, however, such methods are not practical since they are highly time-consuming.

In the latter category, often called heuristics, the search algorithm uses heuristics assistant, which is typically a function, to roughly measure how 'close' the final solution is. This means that the search algorithm does not 'blindly' looks for the final solutions and makes 'educated decisions' to efficiently search promising regions of a search space. Using such heuristic assistances results in higher speed and the

need for significantly low computing and storage devices. However, it comes at the cost of being incomplete. This means that the search algorithm does not guarantee finding the best possible solutions for problems. Since the algorithm makes educated decisions and search promising areas, however, it is able to find reasonably good solution in a reasonably short period of time.

Heuristic search algorithms are suitable for solving problems that we cannot afford using blind search methods. The recent significant advancements in computing and data storage devices have made blind search methods more practical. However, there are still an extremely large number of problems that cannot be solved with them, especially those with exponential or factorial growth. Some of the early heuristic algorithms are: A*, hill climbing [1], and simulated annealing [2].

Early years of heuristic search methods witnessed problem-specific algorithms. This means that they were designed to solve a specific set of problems. For instance, the first version of hill climbing was designed to trees for combinatorial problems. Another property was the need to change the heuristic function for different problems. In other words, they were not general-purpose problem solvers and highly depend on the problem itself.

Such drawbacks led to the proposal of metaheuristics. In such techniques, the algorithm considers a search problem as a black box. The heuristic information can be drawn only using the output of the problem, which is often called objective, cost, or fitness function. The algorithm constantly looks for a better solution by maximizing or minimizing the objective function. Since they are problem independent, they have been widely used in both science and industry.

The field of metaheuristics is one of the fastest growing sub-branches of AI with a large number of conventional and recent algorithms. They can be divided into two main classes: evolutionary and swarm intelligence. In the first class, an algorithm simulates evolutionary phenomena in nature. For instance, genetic algorithm (GA) [3] mimics the Darwinian theory of evolution, in which fittest animals have higher chance of survival.

In the swarm-based methods, an algorithm mimics local interactions between individuals in nature that leads to collaborative problem solving. In an ant colony, for instance, there is no centralized control unit. However, each ant uses pheromone to communicate with other ants to find food sources, defend the colony, or help the queen in babysitting. The ant colony optimization (ACO) [4] algorithm is a metaheuristic that simulates the process of finding the closest path from a nest to a food source in any colony.

This book concentrates on the improvements or use of such algorithms in the area of machine learning. The field of machine learning is full of search and optimization problems that are computationally very expensive. The computational and storage growth of the majority of such problems are exponential. Therefore, blind search methods are not practical to use, so this book focuses on heuristics and metaheuristics.

3 Machine Learning

As discussed in Sect. 1, machine learning is a sub-branch of AI that focuses on the learning aspect of computers. Learning itself is divided into three classes: supervised, unsupervised, and reinforcement. In the following paragraphs, these three methods of learning are discussed with popular methods in each class.

3.1 Supervise Learning

As its name implies, in a supervised learning, there is a supervisor to give the learning algorithm insights on how much an action or decision is good or bad. In supervised learning methods, the data set is completely ladled and the learning method can check whether a particular action is correct or incorrect, and by how much. Popular supervised machine learning algorithms are as follows:

- Support vector machine [5]: This learning method finds an N-dimensional hyper-volume in an N-dimensional space to classify a data set with N features.
- Random forest [6]: In this supervised learning method, multiple decision trees (made or features) are created and merged using a merit factor to provide accurate classification or prediction rates.
- Neural network [7]: This learning method is made of simple neurons arranged in multiple layers and connected using a set of weights. It simulates the way biological neurons work.

3.2 Unsupervised Learning

In this type of learning, the data set is not labelled. This means that the algorithm should find the labels and define them. Such learning algorithms need to learn the structure of data set and the relationship between the features.

- K-means clustering [8]: In this technique, the algorithm clusters data into multiple groups sharing similar or closely related features.
- Self-organizing Neural Networks [9]: In this kind of NN, a learning technique organizes neurons in a way to minimize an error function designed for a problem.

3.3 Reinforcement Learning

In the reinforcement learning, the learning algorithm gets rewarded in case of a correct action and/or gets published in case of a wrong action. This type of learning simulates

the way that creatures learn through rewards and punishments. Some examples of reinforcement learning are as follows:

- Q learning [10]: This learning method is based on Bellman equation and maximizes Q-value.
- Deep Q Network (DQN) [11]: This learning method is similar to Q learning, but it can be generalized.
- Deep Deterministic Policy Gradient (DDPG) [12]: This method is similar to DQN but is able to solve problems with continuous action space.

The focus of this book is on supervised learning methods. In the next section, the use of search methods in machine learning is discussed.

4 Evolutionary Machine Learning

The preceding section introduced the three types of learning methods in the area of machine learning. Regardless of the type of learning, there are a large number of challenges and difficulties that should be handled to efficiently use machine learning techniques. Such problems are often solved by AI search and optimization algorithms. This is exactly the main focus of this book. We try to use the state-of-the-art techniques to tackle some of the most common challenges in machine learning algorithms of which some of them are discussed as follows:

Training classifiers and prediction methods

Training machine learning techniques is to find the optimal learning parameters and even the structure to maximum classification or prediction accuracy. This process is considered as an optimization problem. Gradient-based methods have been popular, but they suffer from local optima stagnation due to the large number of parameters involved in the training process. This makes AI heuristic techniques more reliable alternatives.

Parameter tuning

In addition to the parameters that should be optimized in a trainer, the trainer itself typically has a number of controlling parameters. For instance, the back-propagation algorithm [13] for training NNs has two parameters: learning rate and momentum. Finding an optimal value for such parameters is essential that can be considered as another search/optimization problem. AI heuristic search methods can be used again to optimize this problem.

Multiple objectives

In the majority of machine learning techniques, the objective is to minimize the error. Therefore, the problem can be considered as a single-objective problem. However, we might need to optimize other objectives simultaneously as well. For instance, one might minimize the error and the number of features at the same time. Such problems

are called multi-objective problems. To solve them, the objective can be aggregated using a set of weights. The multi-objective formulation can be maintained as well. In either case, a multi-objective AI search or optimization algorithm can be used to solve the problem.

Over-/under-fitting

Under-fitting and over-fitting are two common issues in the field of machine learning. In over-fitting, a machine learning technique captures noises in the data. This happens when the machine learning technique is too flexible and learns not only the underlying data pattern but also the noises. On the other hand, under-fitting occurs when the machine learning technique is 'rigid' and cannot even learn the underlying pattern in the data.

Both over-fitting and under-fitting are undesirable and result in high error rate in machine learning techniques. Finding a right balance in the structure of machine learning to prevent these two conflicting issues is a challenge that should be considered and addressed when using machine learning techniques.

Feature selection

One of the common issues in the area of machine learning is the existence of a large number of features. To achieve high classification or prediction accuracy, we might not need to include all the features. Some of the features might be correlated, and some might be redundant. In the area of feature selection, an optimal set of features should be found.

5 Structure of the Book

As discussed above, this book presents and proposes a wide range of supervised machine learning methods. The focus is on the use, adaptation, and improvement of recent AI search/optimization algorithms to tackle the problems investigated in Sect. 4. The rest of this book is organized as follows:

Part 1: In the first part of this book, the focus is on classification and prediction. Several types of neural networks are used to solve problems in the areas of Internet of things, link prediction, robot navigation, and quantity prediction. The problem tackled in this part is to improve the training methods and handle multiple objectives using AI optimization algorithms. In addition, support vector machine is chosen as the machine learning technique to classify medical data sets as well. Two recent AI search techniques such as grey wolf optimizer and salp swarm algorithm are used to improve the learning and tune the parameters of the SVM algorithm.

Part 2: In the second part of this book, the problem of feature selection is tackled using a wide range of machine learning and AI search/optimization algorithms. An open-source nature-inspired optimization framework is first proposed to solve feature selection problems using a wide range of optimization algorithms in Python. Several

AI optimization algorithms including particle swarm optimization, Harris hawks optimizer, and evolutionary search are then used to select an optimal set of features.

References

1. Selman B, Gomes CP (2006) Hill-climbing Search, Encyclopedia of cognitive science
2. Van Laarhoven PJ, Aarts EH (1987) Simulated annealing. In: Simulated annealing: theory and applications. Springer, Heidelberg, pp 7–15
3. Holland JH (1992) Genetic algorithms. Sci Am 267:66–73
4. Dorigo M, Di Caro G (1999) Ant colony optimization: a new meta-heuristic. In: Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406), pp 1470–1477
5. Wang L (2005) Support vector machines: theory and applications, vol 177: Springer Science & Business Media
6. Breiman L (2001) Random forests. Mach Learn 45:5–32
7. Zhang GP (2000) Neural networks for classification: a survey. IEEE Trans Syst Man Cybern Part C (Appl Rev) 30:451–462
8. Ding C, He X (2004) K-means clustering via principal component analysis. In: Proceedings of the twenty-first international conference on Machine learning, p 29
9. Ultsch A (1993) Self-organizing neural networks for visualisation and classification. In: Information and classification. Springer, Heidelberg, pp 307–313
10. Watkins CJ, Dayan P (1992) Q-learning. Mach Learn 8:279–292
11. Sorokin I, Seleznev A, Pavlov M, Fedorov A, Ignateva A (2015) Deep attention recurrent Q-network. arXiv preprint arXiv:1512.01693
12. Barth-Maron G, Hoffman MW, Budden D, Dabney W, Horgan D, Muldal A et al (2018) Distributed distributional deterministic policy gradients. arXiv preprint arXiv:1804.08617
13. Hirose Y, Yamashita K, Hijiya S (1991) Back-propagation algorithm which varies the number of hidden units. Neural Netw 4:61–66

Classification and Predication

Salp Chain-Based Optimization of Support Vector Machines and Feature Weighting for Medical Diagnostic Information Systems

Ala' M. Al-Zoubi, Ali Asghar Heidari, Maria Habib, Hossam Faris, Ibrahim Aljarah and Mohammad A. Hassonah

Abstract Nowadays, medical diagnosis based on machine learning is an essential, active, and interdisciplinary research area. Making smart diagnosis and decision support systems have a profound impact on healthcare informatics. Integrating machine learning classifier systems into computer-aided diagnosis systems promotes the early detection of diseases, which results in more effective treatments and prolonged survival. In this chapter, we address popular diagnosis problems using an evolutionary machine learning approach which performs feature weighting and tuning the parameters of support vector machines (SVMs) simultaneously. A new and powerful metaheuristic called salp swarm algorithm is combined with SVM for this task. The designed SSA-SVM approach shows several merits compared to other SVM-based frameworks with well-regarded algorithms such as genetic algorithm (GA) and particle swarm optimization (PSO).

Keywords Medical diagnostic · Machine learning · SVM · SSA · Feature weighting

A. M. Al-Zoubi · M. Habib · H. Faris (✉) · I. Aljarah · M. A. Hassonah
King Abdullah II School for Information Technology,
The University of Jordan, Amman, Jordan
e-mail: hossam.faris@ju.edu.jo

A. M. Al-Zoubi
e-mail: alaah14@gmail.com

M. Habib
e-mail: maryahabeeb@yahoo.com

I. Aljarah
e-mail: i.aljarah@ju.edu.jo

M. A. Hassonah
e-mail: mohammad.a.hassonah@gmail.com

A. A. Heidari
School of Surveying and Geospatial Engineering, College of Engineering,
University of Tehran, Tehran, Iran
e-mail: as_heidari@ut.ac.ir; aliasgha@comp.nus.edu.sg; t0917038@u.nus.edu

Department of Computer Science, School of Computing,
National University of Singapore, Singapore, Singapore

S. Mirjalili et al. (eds.), *Evolutionary Machine Learning Techniques*,
Algorithms for Intelligent Systems, https://doi.org/10.1007/978-981-32-9990-0_2

1 Introduction

Artificial intelligence methods have increasing benefits for experts in medical diagnosis, which minimize potential errors of inexperienced physician or even could reduce the cost of patients monitoring using Internet-based remote techniques. Making accurate, reliable diagnosis at early stages of disease could result in significant positive impact on patient's life. Knowledge discovery based on data mining techniques is the process of searching into large amount of data, in order to reveal hidden pattern of data and produce useful information [1]. Essentially, the outcome of knowledge discovery is used to train computer-aided diagnosis (CAD) systems to support efficient decision making. CAD systems are emerging kind of applications for clinical support systems and decision support systems for disease diagnosis. A pioneering example is the IBM's Watson computational system that identifies the presence of lung, prostate, and breast cancer by dissecting large amount of medical data, and it provides recommendations on treatment options [2].

Our contemporary society is witnessing increasing advances in several fields which is correlated under the industrial and technological revolutions that lead to a surging growth of chronic diseases. Nowadays, chronic diseases are serious health-related problem [3]. According to the National Statistics in UK, liver disorder forms the fifth common factor for death [4]; in USA, the heart disease forms a leading reason for death every year. The continuous developments in science and medical technologies are producing high volume of data at diverse levels of biological systems. Significantly, analyzing and interpreting pervasive, multivariate, and complex medical datasets have massive benefits for transnational or personalized medicine. Over the decades, several research studies have been conducted based on machine learning and data mining techniques for the sake of smart interpretation and analysis of data.

Interestingly, machine learning algorithms have achieved successful deployment in various medical-related problems; for instance [5] showed promising deployment of ensemble learning for schizophrenia prediction, while [6] applied a deep learning approach for Alzheimer's disease prediction and classification. In [7], the authors utilized deep reinforcement learning for lung cancer prediction. Whereas [8] attempted to improve the prediction of Parkinson's disease by proposing a multiple feature evaluation approach of a multi-agent system, which implemented on several classification schemes, nonetheless [9] adopted the convolutional neural network for breast cancer prediction, and in [10] the authors used the neural networks for breast and liver classification. Moreover, different smart classifiers have been applied for the diagnosis of diabetes [11–13]; further, in [14, 15] diverse machine learning classifiers have been applied for liver disorder diagnosis.

Machine learning tools have been showing unprecedented opportunities for medical diagnosis [16]; however, among all, support vector machines demonstrated an eminent promising capability for disease diagnosis [17–22].

Support vector machines (SVMs) are one of the most popular and powerful statistical machine learning methods, which proposed essentially for supervised binary

classification by Vapnik [23]. SVMs make use of support vectors and hyperplanes to identify the decision boundaries between two classes, by maximizing the marginal distance between the hyperplane and the corresponding data instances. SVMs use a kernel function that maps data instances into higher-dimensional space, which makes the process of classes' separation easier and accurate. Certainly, SVMs have been adopted into several application areas, such as: bioinformatics [24, 25], pattern recognition [26, 27], feature selection [28–30], handwritten digit recognition [31, 32], text categorization [33, 34], and image classification [35, 36].

SVMs have superior merits over their counterpart algorithms, since they have stellar generalization ability, a capability to produce high-quality decision boundaries, and the ability to deal with complex and nonlinear classification problems [37]. However, SVMs are extremely sensitive to the initial values of their parameters: the cost (C) and the gamma (γ) parameter of the radial basis function (RBF) kernel [38]. In other words, the accuracy of SVM classifier depends highly on its kernel function, kernel parameters, and the dataset properties, which all influence each other conversely. Therefore, optimizing the algorithm's parameters and the dataset features are at crucial importance to maintain the computational efficiency as robust classifier model.

The C parameter defines a penalty for mis-classified instances; an increased value for C increases the penalty and decreases the number of data points in the error margin, while the gamma parameter is particular for Gaussian SVMs, where it affects the linearity of the hyper-line that separates the hyperplanes. Increasing the value of gamma parameter too much might result in overfitting [39]. Optimizing the hyperparameters of SVMs is critically important; hence, one of the early beginning solutions is the grid search, which is a traditional method that searches for the optimal values of the hyperparameters. However, the grid search technique is exhaustive and time-consuming, and might not perform well [40]. Nevertheless, it cannot perform feature selection as well.

In this chapter, we propose the application of recent swarm intelligent algorithm for tuning the parameters of SVM and optimizing the weights of the input features simultaneously. SSA was proposed by Mirjalili [41], and it is inspired by the foraging behavior of salps, where it mainly imitates the creation of salp chains when searching for food. The proposed SSA-SVM approach is applied on three medical-related datasets drawn from UCI repository [42]; the datasets represent the liver disorder disease, heart disease, and Parkinson's disease. The experiments reveal superior performance results for SSA-SVM approach over other well-regarded metaheuristic algorithms: the genetic algorithm and the particle swarm optimization.

The rest of the chapter is organized as follows; Sect. 2 reviews the main approaches on metaheuristics in combination with SVM , Sect. 3 presents a background for SVM and SSA, Sect. 4 describes the proposed SSA-SVM approach, Sect. 5 is the conducted experiments and obtained results, and finally, Sect. 6 presents concluding remarks.

2 Related Works

Mainly, SVM algorithm is facing two major problems in order to produce efficient classifier model: firstly selecting the best proper kernel parameters, and secondly, how to select the most important set of features. Feature selection is the process of removing redundant and noisy features, yet, keeping the informative features. Wrapper-based feature selection is kind of methods that depend on a performance metric, for example the accuracy, from the induction algorithm to measure the quality of the selected features subset.

Having the optimal set of features can be accomplished by feature selection or feature weighting. The former assigns binary weights either 0 or 1, which means the feature either deleted or kept. The latter assigns integer weights based on a feedback from the induction algorithm. Feature weighting has some advantages over feature selection methods, such as requiring less preprocessing time and less data instances to present good settings [43]. Typically, selecting proper kernel parameter settings, alongside, the best subset of features requires SVMs to use more effective search algorithm than the traditional grid search.

Metaheuristic algorithms are stochastic search algorithms that often utilized to solve hard optimization problems during a logical time [44]. There are many swarm-based optimizers such as dragonfly algorithm (DA) [45], gray wolf optimizer (GWO) [46], salp swarm algorithm (SSA) [41], Harris hawks optimization (HHO) [47], moth-flame optimizer (MFO) [48], water cycle algorithm (WCA) [49], multi-verse optimizer (MVO) [50], ant lion optimizer (ALO) [51], and whale optimizer algorithm (WOA) [52]. Primarily, metaheuristic methods incorporate a randomization component that equips them with the ability to explore different regions of the search space, which enhance solution diversity and avoid stuck in local optima [53, 54]. Basically, different nature-inspired metaheuristic algorithms have been implemented for SVM's parameter optimization, such as the particle swarm optimization [55], genetic algorithm [56–58], bacterial algorithm [59], optimal foraging algorithm [60], sine-cosine algorithm [61], and an artificial immune algorithm [62]. Furthermore [63] presented a comparison for performance evaluation among several bio-inspired metaheuristics, for hyperparameter tuning of SVM classifiers.

Further, throughout the literature, metaheuristics have been applied widely for addressing feature selection [37, 64–67]; however, few algorithms have been designed for optimizing SVM's parameters, simultaneously, with feature selection. In particular, different evolutionary or swarm-based algorithms have been applied for both feature and parameter optimization; thus, authors in [29] adopted grasshopper optimization, while [68] hybridized chaotic search and gravitational search algorithm [69, 70] integrated the use of particle swarm optimization; moreover [71–73] utilized the genetic algorithm [37] used a multi-verse optimizer approach [74] applied a self-adaptive cohort intelligence, while [75, 76] presented the use of ant colony optimization.

Interestingly [38] proposed a hybrid model of genetic algorithms and SVM for feature weighting and parameter optimization. To the best of our knowledge, very few

studies have been focused on optimizing SVM's hyperparameters and performing feature weighting. Hence, in this chapter, a new bio-inspired metaheuristic algorithm called salp swarm algorithm (SSA) has been integrated with SVM to create robust, accurate classifier and feature weighting model in the context of medical diagnosis.

3 Preliminaries

In this section, we provide a brief review on the concepts and mathematical foundations of the utilized techniques.

3.1 Support Vector Machines (SVMs)

SVM is considered as one of the most powerful and well-regarded algorithms in the literature.[1] The main idea behind this SVM is to computationally fit a hyperplane to separate the datasets with d-dimensions seamlessly into two classes. As the input data is not always linearly separable, a kernel-induced feature space can be utilized within SVMs. Additionally, it is possible to attain the VC dimension of SVMs, which is a metric of a method's likelihood to carry out healthy on unobserved info. This is not similar to other learning approaches, for instance, neural networks (NNs), for which we have no proper metric. Generally speaking, SVMs are simple to be implemented, intuitive, hypothetically well established, and the literature shows that SVMs have found their own place among other learning models. SVMs are one of the most utilized classifiers in the literature due to their advantages compared to other learning models. This model only considers the data samples nearby the class boundary to discover a hyperplane for maximizing the margin existing among the classes [77].

As an input to SVM, we have a set of N training vectors $\{\mathbf{x_n}\}_{n=1}^{N}$ and their class labels $\{y_n\}_{n=1}^{N}$, where $\mathbf{x_n} \in R^D$ and $y_n \in \{-1, 1\}$. To start with, let us consider that two classes have to be linearly separable. The hyperplane that is able to separate the classes is obtained via [77]:

$$\mathbf{w}^T \mathbf{x}_n + b = 0, \tag{1}$$

where

$$\mathbf{w}^T \mathbf{x}_n + b \geq 1 \quad \text{for} \quad y_n = +1, \tag{2}$$

$$\mathbf{w}^T \mathbf{x}_n + b \leq -1 \quad \text{for} \quad y_n = -1. \tag{3}$$

[1] The content of this section is based on the lecture note on Mathematical Foundations of Artificial Intelligence of Prof. Daniel Gildea at University of Rochester [77]. We acknowledge their efforts for providing such useful materials, publicly. For full access, refer to http://www.cs.rochester.edu/~gildea/.

Consider that H_1 and H_2 are two hyperplanes. The purpose is to maximize the margin M among two classes. Hence, the objective function can be obtained via:

$$\begin{aligned} &\max_{\mathbf{w}} \ M \\ &\text{s.t. } y^n(\mathbf{w}^T\mathbf{x_n} + b) \geq M, \\ &\quad \mathbf{w}^T\mathbf{w} = 1. \end{aligned}$$

The margin M can be $\frac{2}{\|\mathbf{w}\|}$. Therefore, we have:

$$\begin{aligned} &\min_{\mathbf{w}} \ \frac{1}{2}\mathbf{w}^T\mathbf{w} \\ &\text{s.t. } y^n(\mathbf{w}^T\mathbf{x_n} + b) \geq 1 \end{aligned}$$

In the case of nonlinearly separable classes, we consider slack variables $\{\xi_n\}_{n=1}^N$ and permit some points to be located on a wrong side of the hyperplane considering some cost. Therefore, we have this new cost function:

$$\begin{aligned} &\min_{\mathbf{w}} \ \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{n=1}^{N}\xi_n \\ &\text{s.t. } y^n(\mathbf{w}^T\mathbf{x_n} + b) + \xi_n \geq 1, \\ &\quad \xi_n \geq 0, \ \ \forall n. \end{aligned}$$

The parameter C is adjusted based on a development set. This model can be considered and solved as an initial optimization case to handle SVMs.

Let us consider the Lagrangian for the primal case; hence, we have:

$$\begin{aligned} L(\mathbf{w}, b, \xi, \alpha, \mu) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{n=1}^{N}\xi_n - \sum_n \alpha_n \left[y_n(\mathbf{w}^T\mathbf{x}_n + b)\right] \\ - \sum_n \alpha_n\xi_n + \sum_n \alpha_n - \sum_n \mu_n\xi_n, \end{aligned}$$

where α_n and μ_n, $1 \leq n \leq N$ are Lagrange multipliers. Now, we differentiate the Lagrangian with regard to the variables:

$$\frac{\partial}{\partial \mathbf{w}} L(\mathbf{w}, b, \xi, \alpha, \mu) = \mathbf{w} - \sum_n \alpha_n y_n \mathbf{x}_n = 0 \tag{4}$$

$$\frac{\partial}{\partial b} L(\mathbf{w}, b, \xi, \alpha, \mu) = -\sum_n \alpha_n y_n = 0 \tag{5}$$

$$\frac{\partial}{\partial \xi_n} L\left(\mathbf{w}, b, \xi, \alpha, \mu\right) = C - \alpha_n - \mu_n = 0 \tag{6}$$

After solving these rules, we have:

$$\mathbf{w} = \sum_n \alpha_n y_n \mathbf{x}_n \tag{7}$$

$$\sum_n \alpha_n y_n = 0$$

$$\alpha_n = C - \mu_n \tag{8}$$

Now, we obtain the dual function:

$$\begin{aligned} g(\alpha, \mu) &= \frac{1}{2} \sum_n \sum_m \alpha_n \alpha_m y_n y_m \mathbf{x_n}^T \mathbf{x_m} - \sum_n \sum_n \alpha_n \alpha_m y_n y_m \mathbf{x_n}^T \mathbf{x_m} + \sum_n \alpha_n \\ &= \sum_n \alpha_n - \frac{1}{2} \sum_n \sum_m \alpha_n \alpha_m y_n y_m \mathbf{x_n}^T \mathbf{x_m} \end{aligned} \tag{9}$$

Based on the rules in Eqs. (8) and (9), we can obtain the dual optimization problem as follows:

$$\max_{\alpha} \sum_n \alpha_n - \frac{1}{2} \sum_n \sum_m \alpha_n \alpha_m y_n y_m \mathbf{x_n}^T \mathbf{x_m} \tag{10}$$

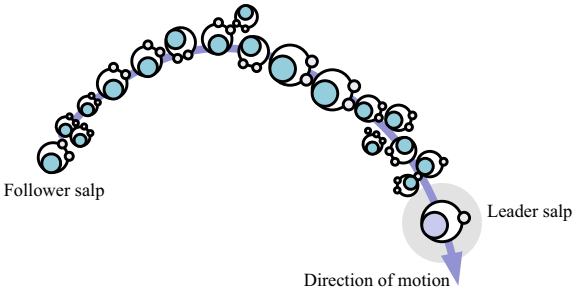
$$\text{s.t. } 0 \leq \alpha_n \leq C. \tag{11}$$

This dual problem is concave and simple to tackle. The dual variables (α_n) are located inside a box having side C. We can often adjust two values α_i and α_j simultaneously and numerically handle this formulation. At the end, we plug in the values of the α_n^*'s to the rules in Eq. (7) to attain the solution $\mathbf{w}^*$.

3.2 Salp Swarm Algorithm

Imitating the daily behaviors of salps in nature inspired the researchers to develop a new optimizer in 2017 [78]. This algorithm has found its place in dealing with many problems including electrical engineering, feature selection [79, 80], and other disciplines [81]. These creatures have a well-organized foraging mechanism that helps them to find the required food source per day and increases their chance of survival. However, these activities cannot be modeled exactly based on nature because a complex model will not help the optimization community to solve a problem in an efficient way. The SSA shows an efficient performance and a good sense of balance

Fig. 1 A salp chain



between its core searching engines. The main process of this method depends on a chain-shaped team-dependent structure, in which each salp has a leader and all salps cooperate to find a food source. This mechanism leads to a condition that a leader salp (food source) guides all members to assist them in finding a favorable region. The mathematical model and operators of SSA are presented as follows.

Like any other population-based method, SSA consists of a series of search agents (salps) to be evolved and increases their quality based on a measured criterion (fitness function). To evolve the quality of solutions and guarantee the convergence behavior of SSA, we first select the best global solution as the leader salp, and then all members of the swarm will follow this agent using a chain-based path to find a high-quality solution (optimum point or optimal solutions) at the termination point of the process. The concept of a salp chain is demonstrated in Fig. 1.

In SSA, we have an initial swarm X, which includes N salps with d-dimensions, as shown in Eq. (12):

$$X_i = \begin{bmatrix} x_1^1 & x_2^1 & \dots & x_d^1 \\ x_1^2 & x_2^2 & \dots & x_d^2 \\ \vdots & \vdots & \dots & \vdots \\ x_1^N & x_2^N & \dots & x_d^N \end{bmatrix} \tag{12}$$

To guide the population toward a food source, we need to update the position of leader in a gradual manner. Hence, we have to update the position of leader based on Eq. (13):

$$x_j^1 = \begin{cases} F_j + c_1 \left(\left(ub_j - lb_j \right) c_2 + lb_j \right) & c_3 \geq 0.5 \\ F_j - c_1 \left(\left(ub_j - lb_j \right) c_2 + lb_j \right) & c_3 < 0.5 \end{cases} \tag{13}$$

where x_j^1 shows the leader and F_j shows the position of food source in the j^{th} dimension, ub_j and lb_j are the boundaries of j^{th} dimension, c_2 and c_3 denote random numbers inside [0, 1], and c_1 is a time-varying parameter formulated as in Eq. (14):

$$c_1 = 2e^{-\left(\frac{4t}{L}\right)^2} \tag{14}$$

where t is iteration and L is the maximum limit of t. The parameter c_1 is designed initially to control the mechanisms of diversification and intensification trends in SSA. Using this adaptive parameter, a more smooth transition can happen. In the previous step, we explained how to update the leader salp. Here, we see the rule in Eq. (15) to adjust the position of follower salps.

$$x_j^i = \frac{x_j^i + x_j^{i-1}}{2} \tag{15}$$

where $i \geq 2$ and x_j^i is the position of i^{th} solution at the j^{th} dimension.

The pseudo-code of SSA is represented in Algorithm 1.

Algorithm 1 Pseudo-code of SSA

```
Create the initial swarm x_i (i = 1, 2, ..., n)
while (Termination condition is not reached) do
   Evaluate the fitness values
   Set the food source F
   Update c_1 using Eq. (14)
   for (each salp (x_i)) do
      if (i <= n/2) then
         Update the leader by Eq. (13)
      else (i > n/2 and i < n + 1)
         Update the followers by Eq. (15)
      end if
   end for
   Update each solution with regard to the legal limits
   Remove illegal salps (out of limits).
end while
Return F
```

4 Proposed SSA-SVM Model

In this section, the proposed classification model based on SSA and SVM is described. In this model, SSA is utilized to simultaneously perform two main tasks: The first is to automatically weight the input features, while the second is to optimize the hyperparameters of the SVM model. For this, design details, evaluation criteria, and the architecture of the model are discussed in the following subsections.

4.1 Individual Representation

As mentioned earlier, the search algorithm is designed and prepared to solve complex problems. In our case, SSA is applied for two parts; the first part consists of searching for the optimal parameters of the SVM classifier, which are C and γ, while the second part is responsible for weighting the features of the dataset as shown in Fig. 2. In other words, the number of values that the SSA covers is the two parameters alongside the number of features for each dataset D. These values are combined in one vector denoted as $D + 2$, where all the values fall in the interval [0, 1].

The first two values of the vector are the C and γ parameters; their search space is different from the initial scale, where C is scaled to be in the interval [0, 35,000], while γ is in the interval [0, 32]. The scaled transformation is based on the min-max normalization equation (see Eq. 16).

$$B = \frac{A - \min_A}{\max_A - \min_A}(\max_B - \min_B) + \min_B \tag{16}$$

The remaining values in the vector match the features of the dataset to be utilized for weighting, however, without modifying the original scale. Additionally, each value is multiplied by its matching feature value for every instance. Therefore, in case we have a dataset of 50 instances, the value that corresponds to that feature is multiplied by all the values of the 50 instances for that specific feature (see the solution structure in Fig. 2).

4.2 Fitness Evaluation

In every iteration, the evaluation is performed for each individual using the selected fitness function. Thus, the SSA can have the feedback to execute. The fitness function in our case is the classification accuracy and calculated as shown in the following equation:

$$fitness(I_i^t) = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{N}\sum_{j=1}^{N}\delta(c(x_j), y_j) \tag{17}$$

where $c(x_j)$ denotes the accuracy of the jth instance of the testing set, while y_j is the jth actual label, and δ is the relation between y_j and $c(x_j)$; if, for example, $y_j = c(x_j)$, then $\delta = 1$, and if not, then $\delta = 0$. N and K are the number of instances and number of folds in the testing set, respectively.

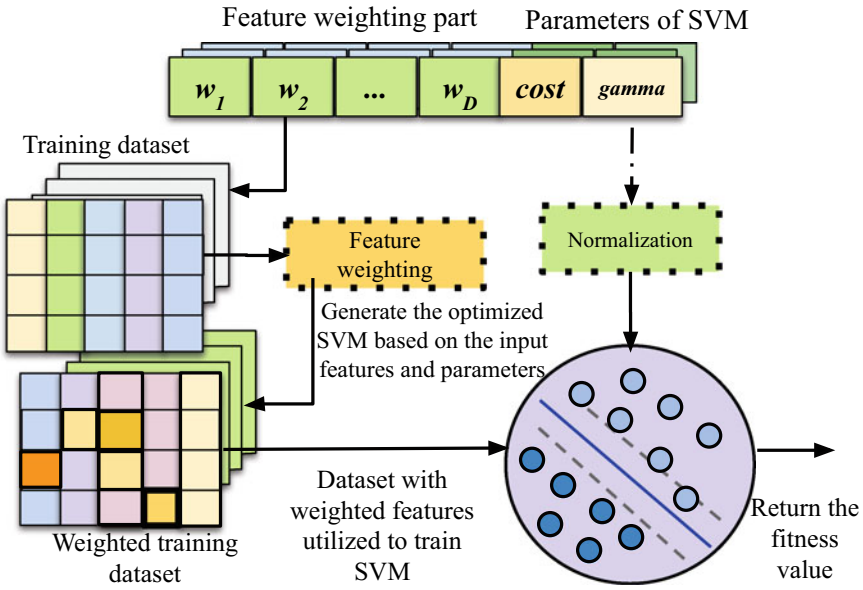


Fig. 2 Solution structure

4.3 System Architecture

The proposed approach commences by splitting the dataset into training and testing sets, and the number of splits is specified by using the same number of experiments. For example, the dataset is divided into k folds, and then the number of experiments will be the same as k, where $k - (1/k)$ folds are for training, and rest $1/k$ is for the testing set. This will guarantee the highest possible diversity of training and testing sets, in order to achieve the most optimal model.

The SSA starts executing by generating a random vector of real numbers, which corresponds to the setting of C and γ, alongside the weights of every feature. Afterward, the SVM classifier begins its training process by running the weighted training split. An inner cross-validation is performed during the training phase, in order to produce a more robust model and to avoid overfitting.

After the training process is finished, SSA will receive the fitness value from the SVM classifier represented by the classification accuracy. All the previous processes will go through a repetition process until the termination criterion for the SSA is met. The termination criterion in our work is the maximum number of iterations. When the number is reached, the optimal individual will be returned by the SSA. The selected individuals are finally used for the testing phase. In the end, for k times, all aforementioned steps are repeated, and the average accuracy out of all k times is considered (see Fig. 3).

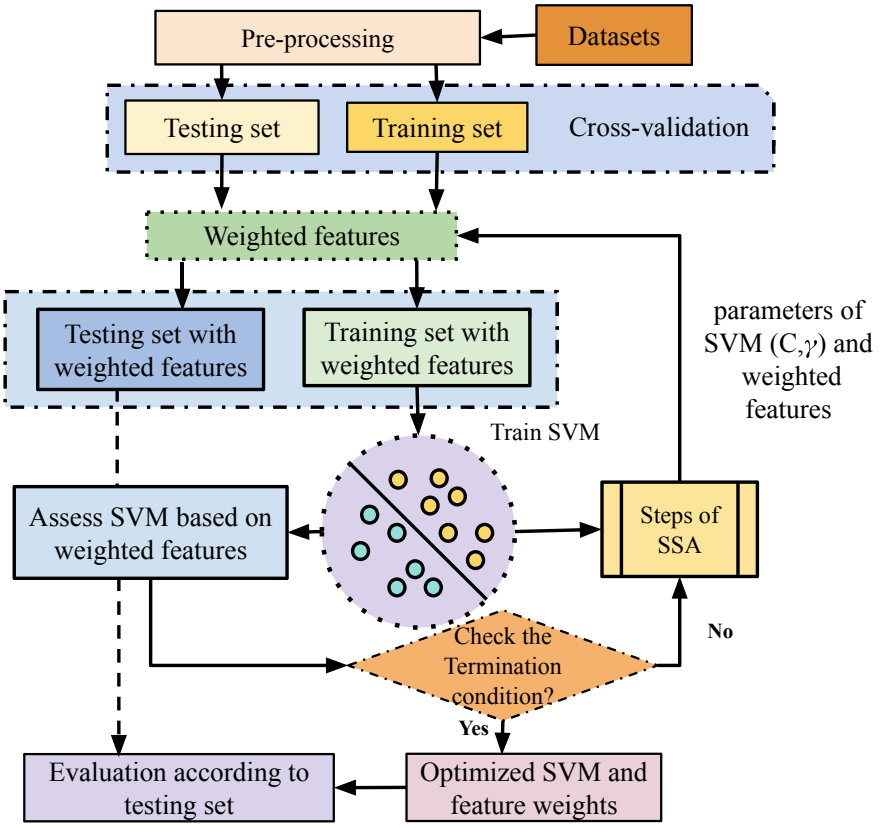


Fig. 3 Proposed SSA-based process

5 Experiment and Results

This section elaborates on the performance of the proposed SSA-based SVM classifier to detect the best approach that deals with the classification of medical datasets. The superiority of SSA should be validated using a comparative study to detect the real hidden exploration and exploitation potentials of the SSA and its ranking position compared to well-established methods. For this purpose, in this study several experiments are performed using different metrics for more robust classification.

5.1 Experimental Setup

Fairness of comparisons is guaranteed by performing the experiments in the same conditions through different aspects. All methods are performed using the same programming language and computer with the details reported in Table 2. Note that the details of system may only affect the speed of computations, but it has no impact on either the accuracy of results or the quality of solutions. Only if we need to record the time results and compare different methods based on elapsed time, details of hardware can help us to have a better interpretation on the performance of different methods.

Three well-known datasets are used for the experiments which were drawn from the UCI repository [82]. The details of the datasets are reported in Table 1. These problems can substantially validate the efficacy of the proposed SSA-based learning model because they cover a varied spectrum of details and characteristics (e.g., number of features, instances, and classes).

The details of our experiments are reported in Table 3. As shown in Table 3, the same population size for all algorithms is used. It is important to note that the population size of swarm-based algorithms plays an essential role in the resulted performance. As we increase the population size, the required computational efforts for initializing and updating the candidate solutions also increase without any clear internal exploratory and exploitative advantages. If we reduce the population size to small values, we should be careful about the negative impact on the efficacy as well because we need enough agents to broadly scan and explore different regions of

Table 1 List of used datasets

#	Dataset	No. of features	No. of instances	No. of classes
1	Liver disorders	6	345	2
3	Parkinsons	22	195	2
3	SAheart	9	462	2

Table 2 Detailed settings of the utilized system

Name	Settings
Hardware	
CPU	Intel Core(TM) i5-6400 processor
Frequency	2.70GHz
RAM	8 GB
Hard drive	500 GB
Software	
Operating system	Windows 7 (64 bits)
Language	MATLAB R2016a

Table 3 Detail of runs

Item	Settings
Splitting criteria	Tenfold
Population size	30
Iterations	50

the feature space. Hence, we need to set a proper number of initial set of solutions. After several trial and errors, we set the population size to 30. This number is not too large to increase the computational time of algorithms without any constructive impact on the performance metrics and not too small to cause negative impacts of the exploratory performance of algorithms. A population-based optimizer like GA [83], PSO, and SSA needs to be repeated enough to find all possible high-quality solutions within the search space. So, we repeated the experiments 50 times in each run. This means the internal iteration number of these methods is set to 50. We also used tenfold cross-validation to record and compare the efficacy of all algorithms.

We also reported the detailed settings of GA and PSO methods in Table 4. It is important to set the initial parameters of PSO and GA, carefully, as the initial parameters of the population-based optimizers such as GA and PSO have significant impact on the resulted performance of these methods. Hence, they should be set based on domain knowledge, sensitivity analysis, properties of the target problem, and dimensions of the feature space. We set these parameters after a sensitivity analysis and several initial experiments are performed.

5.2 Results and Observations

The classification results of the proposed SSA-based approach are compared to those obtained by the GA-based and PSO-based methods in Table 5. Note that these results are recorded based on experimenting the liver disorder dataset. In addition, we

Table 4 Parameter settings

Algorithm	Parameter	Value
GA	Single-point crossover	1
	Mutation	0.01
	Roulette wheel selection	
PSO	Topology fully connected	
	Inertia factor	0.3
	c_1	1
	c_2	1

statistically compare different metrics including accuracy, recall(S), precision(F), precision(S), F-measure, and G-mean measures. The average (Avg) and standard deviation (Std) of results are recorded and reported based on all performed simulations.

As per results in Table 5, we see that the results of SSA-based technique are improved in terms of average and STD of accuracy results.

If we see the results in terms of other metrics, we observe the same pattern. Additionally, the PSO-based approach is observed to be superior to the well-known GA-based method in terms of accuracy results. Statistical test also affirms the significant gaps between the observations. The obtained p-values in Table 8 are less than 0.05. This indicates that the null hypothesis has failed and the difference in results is significant. Other statistical tests can be used; however, this depends on the evaluated results and their characteristics.

The classification results of the proposed SSA-based method are compared to those recorded by the GA-based and PSO-based algorithms in Table 6. The results are attained based on experimenting the Parkinson dataset.

As the results are shown in Table 6, SSA-based technique has achieved the best performance in terms of average and Std of accuracy results. If we have a look at the p-values in Table 8, it is also seen that they are less than 0.05, which as well indicate that the null hypothesis has failed and the difference in results is significant. In terms of different metrics, the superiority of the proposed SSA-based technique is also observed. Based on results, we see the GA can show a slightly better accuracy rate compared to the PSO.

The accuracy results for SAheart dataset are shown in Table 7. We can see that the results evolved using salp chain-based movements are better than those obtained by GA and PSO algorithms. According to G-mean metric, we also see better results of the proposed SSA-based method. In addition, F-measure results support the superiority of the results. Checking the p-values in Table 8, it is also seen that the values are less than 0.05, which reveal that the null hypothesis has failed and the difference in results is significant.

The reasons that the SSA can show better results are not limited to a few remarks. First, the main difference between these algorithms is that they utilize different strategies to update and evolve the population. Generally speaking, evolutionary backgrounds of these methods are not similar. The proposed SSA-based optimizer uses a harmonized chain of salps (search agents) to update the swarm, while there is a leader salp or food source to guide all members of the population in each iteration. The other competitor, PSO, has a different way to evolve the swarm. PSO employs the global best and current best solutions of the population to lead the evolving solutions toward the better regions of the feature space. Hence, it has a memory, which in the case of local optima may decrease the quality of agents in several iterations.

Table 5 Classification results for liver disorder dataset

Algorithm	Accuracy		Recall(F)		Recall(S)		Precision(F)		Precision(S)		F-measure		G-mean	
	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std
GA	68.084	8.340	61.583	15.000	71.521	12.103	58.134	16.944	74.804	9.872	58.571	14.391	65.497	9.313
PSO	68.429	6.776	65.750	15.337	70.054	6.312	53.383	9.846	78.974	9.569	58.341	9.754	67.478	8.715
SSA	**73.672**	6.929	71.432	7.957	75.133	11.279	62.511	15.525	82.503	4.457	65.566	10.339	72.880	5.873

Table 6 Classification results for Parkinson dataset

Algorithm	Accuracy		Recall(F)		Recall(S)		Precision(F)		Precision(S)		F-measure		G-mean	
	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std
GA	92.763	7.488	93.766	6.919	91.571	11.938	96.255	5.201	84.119	16.403	94.897	5.412	92.520	8.510
PSO	92.342	8.600	93.703	7.243	89.702	15.804	95.513	7.422	81.631	22.025	94.521	6.718	91.399	10.386
SSA	**96.395**	4.233	97.333	5.622	95.000	8.051	97.905	3.376	93.810	13.099	97.486	2.947	96.005	4.262

Table 7 Classification results for SAheart dataset

Algorithm	Accuracy		Recall(F)		Recall(S)		Precision(F)		Precision(S)		F-measure		G-mean	
	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std
GA	70.120	7.340	73.923	8.228	61.479	14.486	84.308	7.767	45.076	10.349	78.388	5.755	66.797	8.697
PSO	69.676	6.320	74.582	6.859	57.440	14.113	81.507	6.964	47.697	12.300	77.689	5.313	65.007	8.513
SSA	**72.303**	7.246	75.470	8.036	65.513	13.631	84.856	8.771	48.454	10.518	79.596	6.871	69.942	8.202

Table 8 P-value of the comparisons

Data	PSO	GA
Liver disorders	3.12E−06	2.58E−07
Parkinsons	2.54E−04	2.00E−03
SAheart	3.68E−02	1.42E−02

The GA has another philosophy, in which genes evolve based on crossover and mutation schemes. Again, in the case of immature convergence, GA cannot jump out of the local optima in all iterations. Hence, in a gradual manner, the quality of the solution shows unstable behavior without significant improvement. We observed, in terms of exploration, the SSA can make a better balance between exploratory and exploitative leanings to avoid stagnation downsides, while the basic PSO cannot gain a fine balance. Same statement happens for the comparison of the GA and SSA. Another reason is that none of the PSO-based and GA-based techniques has a dynamic nature similar to that utilized in SSA. SSA uses a parameter c_1 to control the interchange between the searching propensities. This helps SSA to make a more stable balance in the case of immature convergence and stagnation.

However, note that, according to the no free lunch (NFL) theorem [84], no optimizer can always be a winner algorithm. Accurately, SSA may not outperform the PSO and GA using some other datasets. The reason is that an optimizer can show its best performance for a limited set of problems, and if we increase the number of problems to infinite, all optimizers will perform similarly. That is, they win some games, but they will lose the rest of the competitions.

The real scenario is that we never face all problems in the world; hence, we do not need to provide a universally best optimizer. In most of the real-world problems, companies or third parties only request a well-designed competitive optimizer to find the optimal decisions for their problems. As per the NFL, we do not need to always utilize a previously winner optimizer because it may fail this time. This means that we can develop an optimization approach for a specific problem and then it can be better than previous methods. However, if we change the problem, there will be no guarantee to repeat the observations. That new approach may not generate superior results anymore. Nonetheless, this is why we need to understand the nature of the problem before solving it: its characteristics, reasons of selecting a specific optimizer, the role of initial parameters and settings, and diversification and intensification behaviors of the selected optimizer in dealing with similar problems.

6 Conclusion and Future Directions

In essence, analyzing and interpreting large amount of medical data can enhance the quality of disease diagnosis and improve the functionality of the health information systems. This chapter proposed a new approach for feature weighting and

model selection, which implemented the SSA in combination with support vector machines. The SSA-SVM approach was applied to three widespread medical cases. The purpose was to predict the presence of liver disorder disease, the heart disease, and Parkinson's disease. Experiments show that the SSA-SVM has superior performance and enhanced results in terms of accuracy, recall, precision, F-measure, and G-mean. Compared to the GA and PSO, SSA-SVM is able to prove attractable merits that make it a potential approach be applied for computer-aided diagnosis systems.

SSA is still a new method; hence, there are many chances to enhance its structure to reach to higher levels of exploration and exploitation. Such a direction will also lead to enriched performance of the developed SVM-based framework.

References

1. Cios KJ, Pedrycz W, Swiniarski RW (2012) Data mining methods for knowledge discovery, vol 458. Springer Science & Business Media
2. Friedman LF (2014) Ibm's watson supercomputer may soon be the best doctor in the world. Bus Insid, Sci
3. Ambrosio L, Portillo C, Rodríguez-Blázquez C, Rodriguez-Violante M, Castrillo JCM, Arillo VC, Garretto NS, Arakaki T, Dueñas MS, Álvarez M et al (2016) Living with chronic illness scale: international validation of a new self-report measure in parkinson's disease. npj Parkinson's Dis 2:16022
4. Statistics: release calendar, Mar 2019
5. Kalmady SV, Greiner R, Agrawal R, Shivakumar V, Narayanaswamy JC, Brown MRG, Greenshaw AJ, Dursun SM, Venkatasubramanian G (2019) Towards artificial intelligence in mental health by improving schizophrenia prediction with multiple brain parcellation ensemble-learning. Npj Schizophr 5(1):2
6. Spasov S, Passamonti L, Duggento A, Lio P, Toschi N, Neuroimaging Initiative Alzheimer's Disease et al (2019) A parameter-efficient deep learning approach to predict conversion from mild cognitive impairment to alzheimer's disease. NeuroImage 189:276–287
7. Liu Z, Yao C, Hang Y, Taihua W (2019) Deep reinforcement learning with its application for lung cancer detection in medical internet of things. Futur Gener Comput Syst
8. Mostafa SA, Mustapha A, Mohammed MA, Hamed RI, Arunkumar N, Ghani MKA, Jaber MM, Khaleefah SH (2019) Examining multiple feature evaluation and classification methods for improving the diagnosis of parkinson's disease. Cogn Syst Res 54:90–99
9. Ting FF, Tan YJ, Sim KS (2019) Convolutional neural network improvement for breast cancer classification. Expert Syst Appl 120:103–115
10. Brunetti A, Carnimeo L, Trotta GF, Bevilacqua V (2019) Computer-assisted frameworks for classification of liver, breast and blood neoplasias via neural networks: a survey based on medical images. Neurocomputing 335:274–298
11. Choudhury A, Gupta D (2019) A survey on medical diagnosis of diabetes using machine learning techniques. In: Recent developments in machine learning and data analytics. Springer, pp 67–78
12. Das H, Naik B, Behera HS (2018) Classification of diabetes mellitus disease (dmd): a data mining (dm) approach. In: Progress in computing, analytics and networking, pp 539–549. Springer
13. Ndaba M, Pillay AW, Ezugwu AE (2018) An improved generalized regression neural network for type ii diabetes classification. In: International conference on computational Science and its applications. Springer, pp 659–671

14. Haque MR, Islam MM, Iqbal H, Reza MS, Hasan MK (2018) Performance evaluation of random forests and artificial neural networks for the classification of liver disorder. In: 2018 international conference on computer, communication, chemical, material and electronic engineering (IC4ME2), pp 1–5. IEEE
15. Kumar S, Katyal S (2018) Effective analysis and diagnosis of liver disorder by data mining. In: 2018 international conference on inventive research in computing applications (ICIRCA), pp 1047–1051. IEEE
16. AlAgha AS, Faris H, Hammo BH, A-Zoubi AM (2018) Identifying β-thalassemia carriers using a data mining approach: The case of the gaza strip, palestine. Artif Intell Med 88:70–83
17. Das V, Dandapat S, Bora PK (2019) A novel diagnostic information based framework for super-resolution of retinal fundus images. Comput Med Imaging Graph
18. Goyal H, Khandelwal D, Aggarwal A, Bhardwaj P (2018) Medical diagnosis using machine learning. Bhagwan Parshuram Inst Technol 7
19. Samant P, Agarwal R (2018) Machine learning techniques for medical diagnosis of diabetes using iris images. Comput Methods Programs Biomed 157:121–128
20. Saqlain SM, Sher M, Shah FA, Khan I, Ashraf MU, Awais M, Ghani A (2018) Fisher score and matthews correlation coefficient-based feature subset selection for heart disease diagnosis using support vector machines. Knowl Inf Syst 1–29
21. Sidey-Gibbons JAM, Sidey-Gibbons JAM (2019) Machine learning in medicine: a practical introduction. BMC Med Res Methodol 19(1):64
22. Zheng X, Lv G, Zhang Y, Lv X, Gao Z, Tang J, Mo J (2019) Rapid and non-invasive screening of high renin hypertension using raman spectroscopy and different classification algorithms. Spectrochim Acta Part A: Mol Biomol Spectrosc 215:244–248
23. Vapnik VN (1999) An overview of statistical learning theory. IEEE Trans Neural Netw 10(5):988–999
24. Byvatov E, Schneider G (2003) Support vector machine applications in bioinformatics. Appl Bioinform 2(2):67–77
25. Staples M, Chan L, Si D, Johnson K, Whyte C, Cao R (2019) Artificial intelligence for bioinformatics: Applications in protein folding prediction. bioRxiv, pp 561027
26. Burges Christopher JC (1998) A tutorial on support vector machines for pattern recognition. Data Min Knowl Discov 2(2):121–167
27. Wu H, Qing H, Daqing W, Lifu G (2018) A cnn-svm combined model for pattern recognition of knee motion using mechanomyography signals. J Electromyogr Kinesiol 42:136–142
28. Al-Zoubi Ala M, Faris Hossam, Hassonah Mohammad A et al (2018) Evolving support vector machines using whale optimization algorithm for spam profiles detection on online social networks in different lingual contexts. Knowl-Based Syst 153:91–104
29. Aljarah I, Al-Zoubi AM, Faris H, Hassonah MA, Mirjalili S, Saadeh H (2018) Simultaneous feature selection and support vector machine optimization using the grasshopper optimization algorithm. Cogn Comput 1–18
30. Sadiq AS, Faris H, Al-Zoubi AM, Mirjalili S, Ghafoor KZ (2019) Fraud detection model based on multi-verse features extraction approach for smart city applications. In: Smart cities cybersecurity and privacy. Elsevier, pp 241–251
31. Naik VA, Desai AA (2018) Online handwritten gujarati numeral recognition using support vector machine
32. Niu X-X, Suen CY (2012) A novel hybrid cnn-svm classifier for recognizing handwritten digits. Pattern Recognit 45(4):1318–1325
33. Xuelian D, Yuqing L, Jian W, Jilian Z (2019) Feature selection for text classification: a review. Multimed Tools Appl 78(3):3797–3816
34. Mohammad AH, Alwada'n T, Al-Momani O (2018) Arabic text categorization using support vector machine, naïve bayes and neural network. GSTF J Comput (JoC) 5(1):
35. Chandra MA, Bedi SS (2018) Survey on svm and their application in image classification. Int J Inf Technol 1–11
36. Sudharshan PJ, Petitjean C, Spanhol F, Oliveira LE, Heutte L, Honeine P (2019) Multiple instance learning for histopathological breast cancer image classification. Expert Syst Appl 117:103–111

37. Faris H, Hassonah MA, Al-Zoubi AM, Mirjalili S, Aljarah I (2018) A multi-verse optimizer approach for feature selection and optimizing svm parameters based on a robust system architecture. Neural Comput Appl 30(8):2355–2369
38. Phan AV, Nguyen ML, Bui LT (2017) Feature weighting and svm parameters optimization based on genetic algorithms for classification problems. Appl Intell 46(2):455–469
39. Lameski P, Zdravevski E, Mingov R, Kulakov A (2015) Svm parameter tuning with grid search and its impact on reduction of model over-fitting. In: Rough sets, fuzzy sets, data mining, and granular computing. Springer, pp 464–474
40. Staelin C (2003) Parameter selection for support vector machines. Hewlett-Packard Company, Tech. Rep. HPL-2002-354R1
41. Mirjalili S, Gandomi AH, Mirjalili SZ, Saremi S, Faris H, Mirjalili SM (2017) Salp swarm algorithm: a bio-inspired optimizer for engineering design problems. Adv Eng Softw 114:163–191
42. Lichman M (2013) UCI machine learning repository
43. Wettschereck D, Aha DW, Mohri T (1997) A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. Artif Intell Rev 11(1–5):273–314
44. Heidari AA, Pahlavani P (2017) An efficient modified grey wolf optimizer with lévy flight for optimization tasks. Appl Soft Comput 60:115–134
45. Mirjalili S (2016) Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. Neural Comput Appl 27(4):1053–1073
46. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61
47. Heidari AA, Mirjalili S, Faris H, Aljarah I, Mafarja M, Chen H (2019) Harris hawks optimization: algorithm and applications. Futur Gener Comput Syst 97:849–872
48. Mirjalili S (2015) Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm. Knowl-Based Syst 89:228–249
49. Hadi E, Ali S, Ardeshir B, Mohd H (2012) Water cycle algorithm-a novel metaheuristic optimization method for solving constrained engineering optimization problems. Comput & Struct 110:151–166
50. Mirjalili S, Mirjalili SM, Hatamlou A (2016) Multi-verse optimizer: a nature-inspired algorithm for global optimization. Neural Comput Appl 27(2):495–513
51. Mirjalili S (2015) The ant lion optimizer. Adv Eng Softw 83:80–98
52. Mirjalili S, Lewis A (2016) The whale optimization algorithm. Adv Eng Softw 95:51–67
53. Heidari AA, Abbaspour RA, Jordehi AR (2017) An efficient chaotic water cycle algorithm for optimization tasks. Neural Comput Appl 28(1):57–85
54. Yang X-S (2010) Nature-inspired metaheuristic algorithms. Luniver press,
55. Guo XC, Yang JH, Wu CG, Wang CY, Liang YC (2008) A novel ls-svms hyper-parameter selection based on particle swarm optimization. Neurocomputing 71(16–18):3211–3215
56. Chunhong Z, Licheng J (2004) Automatic parameters selection for svm based on ga. In: Fifth world congress on intelligent control and automation (IEEE Cat. No. 04EX788), vol 2, pp 1869–1872. IEEE
57. Nanda MA, Seminar KB, Solahudin M, Maddu A, Nandika D (2018) Implementation of genetic algorithm (ga) for hyperparameter optimization in a termite detection system. In: Proceedings of the 2nd international conference on graphics and signal processing. ACM, pp 100–104
58. Ren Y, Bai G (2010) Determination of optimal svm parameters by using ga/pso. JCP 5(8):1160–1168
59. Jin Q, Chi M, Zhang Y, Wang H, Zhang H, Cai W (2018) A novel bacterial algorithm for parameter optimization of support vector machine. In: 2018 37th Chinese control conference (CCC), pp 3252–3257. IEEE
60. Sayed GI, Soliman M, Hassanien AE (2019) Parameters optimization of support vector machine based on the optimal foraging theory. In: Machine learning paradigms: theory and application. Springer, pp 309–326

61. Mirjalili S (2016) Sca: a sine cosine algorithm for solving optimization problems. Knowl-Based Syst 96:120–133
62. Ilhan A, Mehmet K, Erhan A (2011) A multi-objective artificial immune algorithm for parameter optimization in support vector machine. Appl Soft Comput 11(1):120–129
63. Godínez-Bautista A, Padierna LC, Rojas-Domínguez A, Puga H, Carpio M (2018) Bio-inspired metaheuristics for hyper-parameter tuning of support vector machine classifiers. In: Fuzzy logic augmentation of neural and optimization algorithms: theoretical aspects and real applications. Springer, pp 115–130
64. Mafarja M, Aljarah I, Faris H, Hammouri AI, Al-Zoubi AM, Mirjalili S (2019) Binary grasshopper optimisation algorithm approaches for feature selection problems. Expert Syst Appl 117:267–286
65. Mafarja M, Aljarah I, Heidari AA, Faris H, Fournier-Viger P, Li X, Mirjalili S (2018) Binary dragonfly optimization for feature selection using time-varying transfer functions. Knowl-Based Syst 161:185–204
66. Mafarja M, Aljarah I, Heidari AA, Hammouri AI, Faris H, Al-Zoubi AM, Mirjalili S (2018) Evolutionary population dynamics and grasshopper optimization approaches for feature selection problems. Knowl-Based Syst 145:25–45
67. Mafarja M, Heidari AA, Faris H, Mirjalili S, Aljarah I (2020) Dragonfly algorithm: theory, literature review, and application in feature selection. In: Nature-Inspired Optimizers. Springer, pp 47–67
68. Chaoshun Li, Xueli An, Ruhai Li (2015) A chaos embedded gsa-svm hybrid system for classification. Neural Computing and Applications 26(3):713–721
69. Huang C-L, Dun J-F (2008) A distributed pso-svm hybrid system with feature selection and parameter optimization. Appl Soft Comput 8(4):1381–1391
70. Liu Y, Wang G, Chen H, Dong H, Zhu X, Wang S (2011) An improved particle swarm optimization for feature selection. J Bionic Eng 8(2):191–200
71. Bouraoui A, Jamoussi S, BenAyed Y (2017) A multi-objective genetic algorithm for simultaneous model and feature selection for support vector machines. Artif Intell Rev 1–21
72. Huang C-L, Wang C-J (2006) A ga-based feature selection and parameters optimizationfor support vector machines. Expert Syst Appl 31(2):231–240
73. Sarafrazi S, Nezamabadi-pour H (2013) Facing the classification of binary problems with a gsa-svm hybrid system. Math Comput Model 57(1–2):270–278
74. Aladeemy M, Tutun S, Khasawneh MT (2017) A new hybrid approach for feature selection and support vector machine model selection based on self-adaptive cohort intelligence. Expert Syst Appl 88:118–131
75. Costa VO, Rodrigues CR (2018) Hierarchical ant colony for simultaneous classifier selection and hyperparameter optimization. In: 2018 IEEE congress on evolutionary computation (CEC), pp 1–8. IEEE
76. Huang C-L (2009) Aco-based hybrid classification system with feature subset selection and model parameters optimization. Neurocomputing 73(1–3):438–448
77. Gildea D, Naim I (2013) CSC 446 notes: Lecture 7
78. Faris H, Mirjalili S, Aljarah I, Mafarja M, Heidari AA (2020) Salp swarm algorithm: theory, literature review, and application in extreme learning machines. Springer International Publishing, Cham, pp 185–199
79. Aljarah I, Mafarja M, Heidari AA, Faris H, Zhang Y, Mirjalili S (2018) Asynchronous accelerating multi-leader salp chains for feature selection. Appl Soft Comput 71:964–979
80. Faris H, Mafarja MM, Heidari AA, Aljarah I, Al-Zoubi AM, Mirjalili S, Fujita H (2018) An efficient binary salp swarm algorithm with crossover scheme for feature selection problems. Knowl-Based Syst 154:43–67
81. Zhang Q, Chen H, Heidari AA, Zhao X, Xu Y, Wang P, Li Y, Li C (2019) Chaos-induced and mutation-driven schemes boosting salp chains-inspired optimizers. IEEE Access 7:31243–31261
82. Lichman M et al (2013) Uci machine learning repository

83. Faris H, Al-Zoubi AM, Heidari AA, Aljarah I, Mafarja M, Hassonah MA, Fujita H (2019) An intelligent system for spam detection and identification of the most relevant features based on evolutionary random weight networks. Inf Fusion 48:67–83
84. Wolpert DH, Macready WG et al (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82

Support Vector Machine: Applications and Improvements Using Evolutionary Algorithms

Seyed Hamed Hashemi Mehne and Seyedali Mirjalili

Abstract A description of the theory and the mathematical base of support vector machines with a survey on its applications is first presented in this chapter. Then, a method for obtaining nonlinear kernel of support vector machines is proposed. The proposed method uses the gray wolf optimizer for solving the corresponding nonlinear optimization problem. A sensitivity analysis is also performed on the parameter of the model to tune the resulting classifier. The method has been applied to a set of experimental data for diabetes mellitus diagnosis. Results show that the method leads to a classifier which distinguished healthy and patient cases with 87.5% of accuracy.

Keywords Machine learning · Support vector machine · Optimization · Meta-heuristics · Evolutionary algorithms · Benchmark · Artificial intelligence

1 Introduction

Supervised learning is a branch in the field of machine learning dealing with a given set of labeled data. This data that consist of input and outputs is used to train the desired model. Supervised learning has itself two types of classification and regression algorithms. A classification algorithm attempts to organize the data or observations into distinct categories in order to locate new data in future easily. Therefore, it consists of two main parts: model training and prediction. Support vector machine (SVM) is a well-known class of algorithms for data classification. Based on the literature, SVMs have been used in many different practical problems effectively.

S. H. H. Mehne
Aerospace Research Institute, 1465774111 Tehran, Iran
e-mail: hmehne@ari.ac.ir

S. Mirjalili (✉)
Torrens University Australia, Brisbane, QLD 4006, Australia
e-mail: ali.mirjalili@gmail.com

Griffith University, Brisbane, QLD 4111, Australia

S. Mirjalili et al. (eds.), *Evolutionary Machine Learning Techniques*,
Algorithms for Intelligent Systems, https://doi.org/10.1007/978-981-32-9990-0_3

In this chapter, a literature review of support vector machines, their theory, and applications are given at the first for motivations. The main focus is on the optimization problem that arises in determining a typical SVM. As this problem is usually nonlinear and with a large number dimensions, a wide range of optimization algorithms has been reported to solve it in literature. Meta-heuristic evolutionary methods such as ant colony [33], particle swarm optimization [32], and whale optimization method [2, 36] have been applied to solve the optimization problem or in model parameter tuning.

In this work, the gray wolf optimizer (GWO) has been adopted to solve the problem of SVM accompanied with a radial basis kernel. The GWO is a nature-inspired evolutionary algorithm for optimization and has been developed based on the social life of gray wolfs. As the method is originally for unconstrained problems, in the case of SVM, the related constraint is added to the objective function as a penalty. In order to make an evaluation, the method has been applied to a set of experimental data for diabetes. Results show that the method solves the SVM problem and finds the classifier efficiently. Moreover, a parameter analysis has been also performed to find the optimum value of the parameter. The method with an optimum parameter was able to predict and distinguish the healthy and patient cases based on the features with 87.5% correctness.

There are other implementations of GWO in SVM classifications in the literature. As an instance, GWO has been used in [8] for optimal setting of SVM parameters in order to increase the accuracy of the model. In [22], an improved GWO was introduced for feature selection by separating helpful data from the database. Optimizing the SVM parameters was also the goal of GWO utilization in [27]. The method is examined for image classification in the field of water quality management.

The main difference between the above-mentioned applications of GWO and the present method of this chapter is that they employed the GWO for parameter optimization and feature selection, while we use it for solving the main problem of margin maximization which is of higher dimension. For parameter optimization, we use a simple parameter estimation method due to the low number of decision variables. Therefore, the proposed method has lower computational complexity in the phase of parameter tuning.

The rest of this chapter is organized as follows: In Sect. 2, the theory and application of SVMs are reviewed and explained. The method of predictive model construction is explained in Sect. 3. The implementation of the method on experimental data and model analysis is discussed in Sect. 4. Finally, some concluding remarks and future work suggestions are given in Sect. 5.

2 Support Vector Machine

In this section, applications of the support vector machines in different fields of research will be reviewed. Then, the mathematical base and theoretical notes will be explained.

2.1 Applications of SVM

- Genetics: Generating gene classifiers that assign genes to special labels based on a set of given data is a principal task in genetics. Statistical methods, fuzzy logic, and artificial neural networks are traditional tools in gene classification. Support vector machines are also interesting for this type of classification and demonstrate successful results especially for large data sets. For example, in [7], SVM approach to microarray gene classification problem has been presented. The SVM model has been developed using a set of gene expression samples. Simulations show that the resulting classifier has the ability to assign new samples to certain cancer diseases. A mixed kernel SVM has been also proposed in [36] for prediction of colorectal cancer. Results show that the classifier discriminates healthy and patient samples with improved accuracy in the presence of redundant genes and imbalanced data. Another recent research in this field was reported in [4], where an improved penalized SVD for identification of informative genes and pathways among cancer microarray data has been proposed.
- Environmental hazards: Natural hazards such as earthquakes, volcanic eruptions, wildfire, floods, and droughts cause the death of livings, infrastructures and property damage, and environmental degradation. Therefore, prediction and warning of such hazards and identifying the potential of disasters will reduce corresponding undesirable effects.
  Technological enhancements such as remote sensing, geographic information systems, and high-performance computing alongside with statistical methods, fractals, and artificial intelligence increase the capability of prediction.
  In recent years, machine learning tools such as vector support machine have been employed to drive predictive models for natural disasters. For example, SVM was utilized to assign the probability of wildfire to a certain region in [9]. This results in a map with four levels of fire ignition potentials helping more effective surveillance of human life and natural resources. Development of a drought forecasting model has been also reported in [6]. This model is based on five meteorological sites in eastern Australia using SVM.
- Pathological brain detection: Magnetic resonance imaging (MRI) is the common source of data in pathological brain detection. The traditional human-based method of interpretation of brain images is time-consuming, costly, and unreliable. Therefore, computer-based methods have been proposed to automate the process of classification these images as healthy or pathological. SVM has been also implemented for this type of classification. For example, in [30], special versions of SVM were proposed as a classifier for brain images with more than 99% accuracy. In [34], the generalized eigenvalue proximal SVM has been utilized as a binary classifier to develop an automatic system to determine abnormal and normal brains. Diagnosis of the depressive disorder based on MRI and brain networks reported in [21] is another example of using SVM classifier. SVM has been also used in [12] for early diagnosis of Alzheimer's disease. The method is able to categorize samples as Alzheimer's disease, mild cognitive impairment, and elderly normal control.

- Solar radiation prediction: Study of solar radiation is a major prerequisite task in the design and installation of photovoltaic and thermal electrical power stations. In order to reduce the cost of direct measurement, predictive models have been developed to estimate solar radiation from meteorological data. SVM has been used recently as a reliable tool in generating such accurate predictive models. For example, in [20], SVM has been utilized to predict the monthly mean horizontal global solar radiation based on sunshine duration and maximum–minimum ambient temperature as inputs. Reported in [19], the combination of support vector machine and wavelet transform algorithm provided a model to predict daily and monthly horizontal global solar radiation. The results indicate more than 88% accuracy and functional excellence with respect to other models. Another notable work in this field is [3].
- Text categorization: Assigning predefined labels or categories to text according to its content is text categorization or text classification. Among widespread applications of text categorization, one can mention web news organization, academic paper classification, spam filtering, and language detection.
  Rule-based methods, artificial intelligence, and machine learning systems are the most commonly used approaches to text categorization. Due to the linearly separable nature of text categorization, linear SVM has been utilized. It usually offers accurate results with a small amount of training data. Some examples of employing SVM for text classification are [2, 10, 29].
- Miscellaneous applications: In addition to the above-mentioned application of support vector machines, there are many real-world practical applications in literature such as underwater acoustic target classification [24], business and economy [16, 31, 32], health monitoring of mechanical systems [1, 5, 14, 26], automatic cloud classification [28], facial emotion recognition [35], electrical power and electricity [11, 15, 23], pan evaporation modeling [13], and crop classification [37].

2.2 How Does SVM Work?

As mentioned before, SVM algorithms generate learning models for automatic data classification based on a set of given data. To start, let us assume the problem of separating a set of data into two classes. As shown in Fig. 1 in a two-dimensional example, to separate circles from squares, different straight lines may be drawn. Here, the inputs are x_1 and x_2, the coordinates of a point while its shape (circle or square) is the output or class label. H_1 and H_2 are straight lines or more general hyperplanes that refer to linear classification. If one on these hyperplanes is selected for the classifier model, then when a new data received, the decision on its shape depends on its position relative to the hyperplane.

Instead of linear classification, one may use nonlinear separator when data cannot be separated linearly. An example of a two-dimensional nonlinear separator has been shown in Fig. 2. Decision between linear or nonlinear classification depends on distribution of samples in the input data.

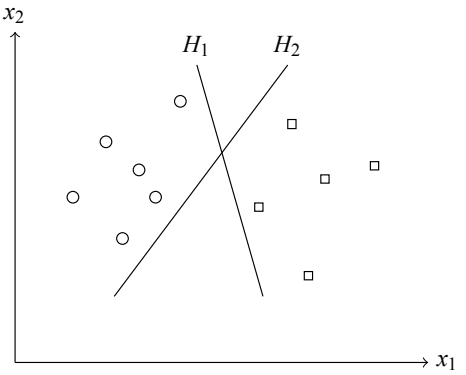


Fig. 1 An example of 2d linear data classification

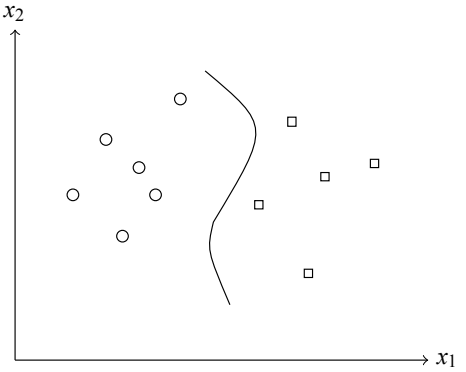


Fig. 2 An example of 2d data nonlinear classification

2.2.1 Linear SVM

As it was discussed and visualized in Fig. 1, there are many different separator lines for an individual data set of training. In support vector machine, the aim is to find the best of them. For further description, let us consider the following set of data with N samples.

$$D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots (\mathbf{x}_N, y_N)\} \tag{1}$$

where in a sample $(\mathbf{x}_i, y_i)$, $\mathbf{x}_i$ is the input vector in $\mathbb{R}^q$ and y_i is the corresponding output. In this general formulation of q-dimensional classification problem, the hyperplanes are subspaces of dimension $q - 1$ like straight lines in two-dimensional space.

To make a geometrical sense, let us consider the problem with inputs in $\mathbb{R}^2$ as given in Fig. 3. Here, there are two distinguished categories and then $y_i \in \{-1, 1\}$. Linear SVM is seeking for hyperplanes with a maximum possible margin like dash lines in Fig. 3. Vectors are allowed to appear on hyperplanes and they are called support vectors.

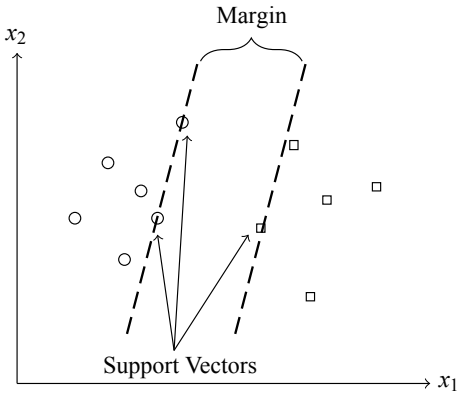


Fig. 3 An example of 2d data nonlinear classification

If $\mathbf{w}$ is the normal vector to these hyperplanes, the equations of them are as follows:

$$\mathbf{w}.\mathbf{x} + b = 1 \tag{2}$$
$$\mathbf{w}.\mathbf{x} + b = -1 \tag{3}$$

Therefore, any point above or on the first hyperplane belongs to the class with label 1 and any point below or on the second hyperplane belongs to the class label -1. In general, the margin between such two parallel separating hyperplanes is $\frac{2}{\|\mathbf{w}\|}$. Now, maximizing the margin is equivalent to minimizing $\|w\|$ with the following constraints guarantee the separation of samples:

$$\mathbf{w}.\mathbf{x}_i + b \geq 1, \quad \text{if} \quad y_i = 1 \tag{4}$$
$$\mathbf{w}.\mathbf{x}_i + b \leq -1, \text{ if} \quad y_i = -1 \tag{5}$$

The above inequalities may be also combined to the following

$$y_i(\mathbf{w}.\mathbf{x}_i + b) \geq 1, \quad \text{for all} \quad i = 1, 2, \ldots, N \tag{6}$$

In summary, the problem of finding maximum margin hyperplanes will be written as:

$$\text{Min} \quad \|\mathbf{w}\| \tag{7}$$
$$\text{subject to:}$$
$$y_i(\mathbf{w}.\mathbf{x}_i + b) \geq 1, \quad i = 1, 2, \ldots, N. \tag{8}$$

Here, $\mathbf{w}$ is the normal vector of hyperplanes and is the decision variable of the above optimization problem with b. Therefore, the problem has $q + 1$ decision variables and N constraints. As soon as the above problem is solved, the classifier will be determined as $x \mapsto \text{sign}(\mathbf{w}.\mathbf{x} - b)$ which maps a new data to a class.

In order to convert the problem to one with easier constraints, the Lagrangian form of (7)–(8) is usually considered. If λ_i is non-negative Lagrange multiplier corresponding to ith constraints of (8), then the reformulated form of (7)–(8) is as follows:

$$\text{Min } L_P = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} \lambda_i y_i (\mathbf{w}.\mathbf{x}_i + b) + \sum_{i=1}^{N} \lambda_i \tag{9}$$

subject to:

$$\lambda_i \geq 0, \quad i = 1, 2, \ldots, N. \tag{10}$$

Because the necessary condition of optimum for (9) is to vanish its gradient, the following conditions hold:

$$\frac{\partial L_P}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{N} \lambda_i y_i x_i = 0 \tag{11}$$

$$\frac{\partial L_P}{\partial b} = \sum_{i=1}^{N} \lambda_i y_i = 0 \tag{12}$$

Therefore, the solution will satisfy in

$$\mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i x_i \tag{13}$$

$$\sum_{i=1}^{N} \lambda_i y_i = 0 \tag{14}$$

Substituting (13)–(14) in (9) leads to the dual form of the problem as follows:

$$\text{Max } L_D = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_i y_i y_j (\mathbf{x}_i.\mathbf{x}_j) \tag{15}$$

subject to:

$$\sum_{i=1}^{N} \lambda_i y_i = 0, \quad i = 1, 2, \ldots, N, \tag{16}$$

$$\lambda_i \geq 0, \qquad i = 1, 2, \ldots, N. \tag{17}$$

The above dual problem is a convex optimization problem and then has a unique solution namely $\{\lambda_1^*, \lambda_2^*, \ldots, \lambda_N^*\}$. Based on the nature of duality, nonzero λ_i^*s correspond to equality constraints, i.e., the support vectors. Therefore, since a number of support vectors are usually less than the inner data points, most of λ_i^*s in the optimal solution are zero.

As soon as the solution is found, the original optimal values of $\mathbf{w}$ will be calculated as the following linear combination of input data $\mathbf{x_i}$ with the product of outputs y_i and λ_i^*s as weight coefficients:

$$\mathbf{w}^* = \sum_{i=1}^{N} \lambda_i^* y_i \mathbf{x}_i \tag{18}$$

The bias term b^* will be also computed from the following relation, where i_0 is the index of a support vector:

$$y_{i_0}(\mathbf{w}^* \mathbf{x}_{i_0} + b^*) = 1 \tag{19}$$

The classifier function that determines the related category of new data such as $\mathbf{x}$ has also the following form:

$$f(\mathbf{x}) = \text{sign}(\sum_{i=1}^{N} \lambda_i^* y_i \mathbf{x}_i \mathbf{x} + b^*) \tag{20}$$

2.3 *Interpretation of Inner Product*

One of the benefits in dealing with the dual problem is the appearance of the inner products of training data, i.e., $x_i.x_j$ in the formulation of (15) which plays a key rule in nonlinear SVMs. This dot product has also a description of similarity: if x_i and x_j are similar vectors then they are parallel and in the case of unit length, their inner product equals to 1. On the other hand, if they are completely dissimilar, they are perpendicular and $x_i.x_j = 0$. Therefore, $x_i.x_j$ will measure the similarity of x_i and x_j.

Now, let us assume that x_i and x_j are completely dissimilar, then they have not contributed to L_D, based on (15). When x_i and x_j are similar, then there are two cases: if $y_i = y_j$, then $\lambda_i \lambda_i y_i y_j (\mathbf{x}_i.\mathbf{x}_j)$ is positive when $\lambda_i, \lambda_j > 0$ and this has negative effect on the objective function. Therefore, in this case, it is better that at least one of the λ_i and λ_j be zero. This forces to form $\mathbf{w}$ such that they are in the same class. If $y_i \neq y_j$, then the positive effect on objective is achieved when $\lambda_i, \lambda_j > 0$ which means that the optimizing process forces to put them in different classes.

2.3.1 Nonlinear SVM

In the last section, the linear SVM has been discussed. In linear SVM, the classes are separated with hyperplanes or in the special case of two-dimensional, straight lines. However, many real-world examples may be addressed where experimental data cannot be divided by straight lines. As it is shown in Fig. 4, separating squares

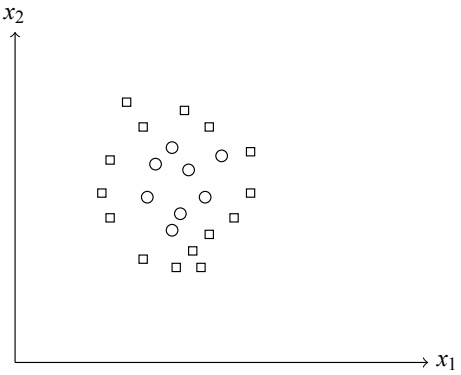


Fig. 4 A data sample which is not separated linearly

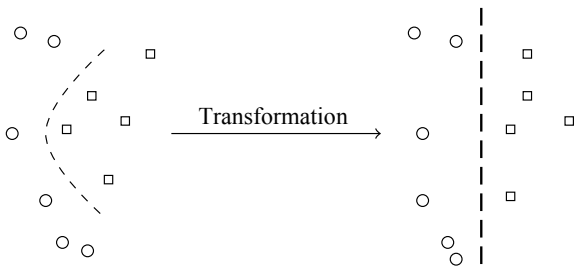


Fig. 5 Transformation of data into a linearly separable space

and circles requires a closed curve rather than straight lines. For such cases that the relation between classes is complex, the nonlinear SVM has been introduced.

In the case of nonlinear data distribution, a mapping is used to the input data to transform it to a linearly separable data as illustrated in Fig. 5. This transformation is usually called the features mapping which maps $\mathbf{x}$ to $\phi(\mathbf{x})$. In this case, the inner product $\mathbf{x_i x_j}$ in the objective function (15) is substituted by $K(\mathbf{x_i}, \mathbf{x_j}) = \phi(\mathbf{x_i})\phi(\mathbf{x_j})$. In more general, kernel function is defined based on the input data patterns. Some of the important kernels are given in Table 1.

Table 1 Examples of kernel functions

Pattern type	Kernel function	Parameters
Linear	$\mathbf{xz}$	
Polynomial (homogeneous)	$(\mathbf{xz})^p$	p is the tuning parameter
Polynomial (inhomogeneous)	$(\mathbf{xz} + c)^p$	p and c are tuning parameters
Radial basis function	$e^{(-\frac{\|\mathbf{x}-\mathbf{z}\|^2}{2\sigma^2})}$	σ is the tuning parameter
Sigmoid	$\tanh(\kappa\mathbf{xz} + c)$	$\kappa > 0$ and $c < 0$ are restricted parameters

In the case of using kernel function, the corresponding classifier function which has been given for linear case in (20) will be expressed as follows:

$$f(\mathbf{x}) = \text{sign}(\sum_{i=1}^{N} \lambda_i^* y_i K(\mathbf{x}_i \mathbf{x}) + b^*) \tag{21}$$

3 Method's Description

In this section, the proposed method for SVM based on radial basis kernel is presented. In this type of kernel function, there is a parameter σ which should be determined in an optimal way. Therefore, in this case, we faced two optimization problems, one for choosing the optimal value of parameters and second for finding the corresponding (w, b). We call them outer (for parameters) and inner optimization problem. As the outer one has only one decision variable, it is simpler than the outer with N variables. Therefore, for the outer optimization problem, a simple method of parameter analysis is used and for the inner one which is more complex, a nature-inspired meta-heuristic method called gray wolf optimizer (GFO) is implemented.

Fig. 6 indicates the flowchart of the proposed method. In the first step, the input data is divided into training and testing data that contain 80% and 20% of the input data, respectively. Then, an initial value for σ is generated to construct the abstract form of the kernel. In the next step, the gray wolf optimizer solves the corresponding dual problem presented by (15)–(17). From the optimal solution, the classifier is formed. Then, by applying the test data into the resulting classifier, the model is evaluated. The evaluation phase is to compare the output of classifier with the output of the test data. The procedure is repeated until the desired accuracy occurs.

3.1 Gray Wolf Optimizer

Gray wolf optimizer is a nature-inspired evolutionary method for optimization algorithm proposed by Mirjalili et al. in 2014 [17]. This algorithm simulates the leadership hierarchy and hunting mechanism of gray wolves in nature for optimization problems. The method has been used to solve different challenging problems in various application areas successfully. As a relatively complete reference on this method, its applications, and literature review, one may refer the reader to [18].

The method starts with a population of gray wolfs as search agents. Then, in each iteration, the population is updated based on fitness values in a random way. The algorithm of the method for minimizing a function $g(x)$ is as follows:

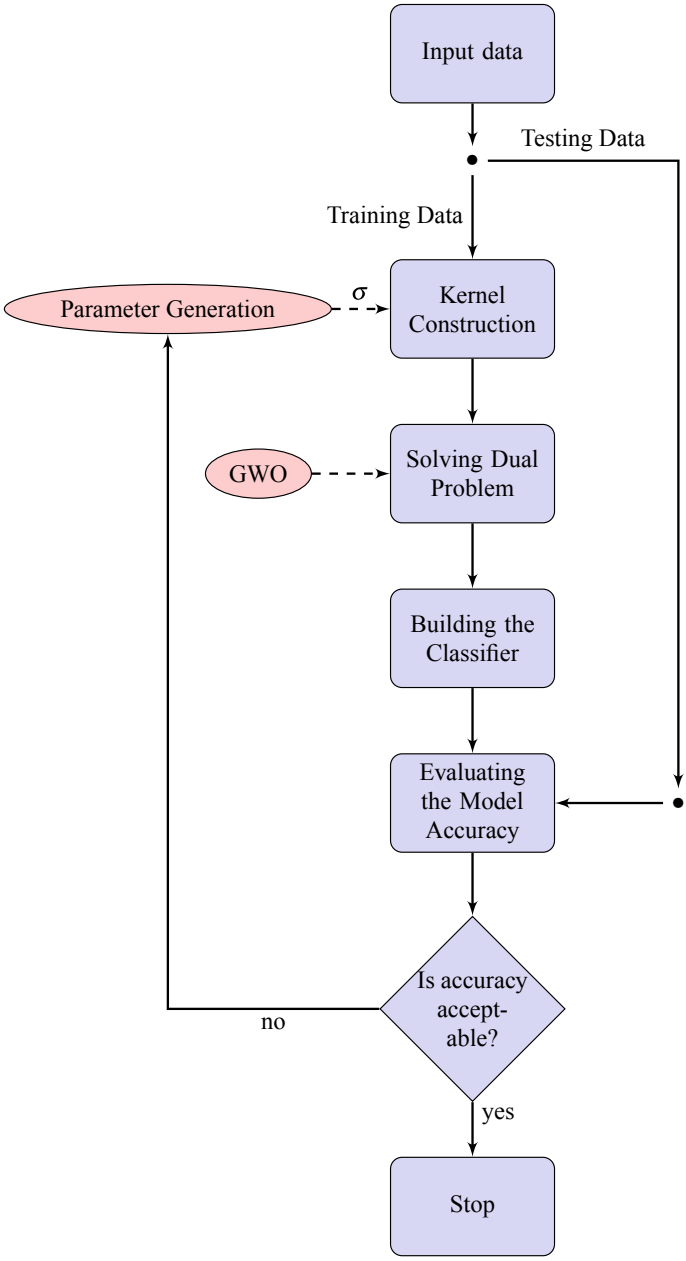


Fig. 6 Flowchart of construction and evaluating of the proposed SVM

Gray Wolf Optimizer algorithm

Start: Choose an initial population of search agents or gray wolfs $\{X_i : i = 1, 2, \cdots, n\}$, choose maximum number of iteration K. Let $t = 1$.
Step 1: Calculate the fitness value of each search agent e.i. $\{g(X_i) : i = 1, 2, \cdots, n\}$.
Step 2: Determine the best, the second best and the third best search agent as follows:
$X_\alpha = \text{argmin}\{g(X_i) : i = 1, 2, \cdots, n\}$
$X_\beta = \text{argmin}\{g(X_i) : i = 1, 2, \cdots, n, i \neq \alpha\}$
$X_\delta = \text{argmin}\{g(X_i) : i = 1, 2, \cdots, n, i \neq \alpha, \beta\}$
Step 3: Let $i = 1$ and $a = 2(1 - \frac{t}{K})$.
Step 4: Choose six normal random vector r_{1j}, r_{2j}, $j = 1, 2, 3$. Calculate the followings:
$A_j = a(2r_{1j} - \mathbf{1})$, $C_j = 2r_{2j}$, $j = 1, 2, 3$.
$D_\alpha = |C_1.X_\alpha - X_i|$, $D_\beta = |C_2.X_\beta - X_i|$, $D_\delta = |C_3.X_\delta - X_i|$,
$X_1 = X_\alpha - A_1.D_\alpha$, $X_2 = X_\beta - A_2.D_\beta$, $X_3 = X_\delta - A_3.D_\delta$
Step 5: Update search agents by $X_i = \frac{X_1+X_2+X_3}{3}$.
Step 6: If $i < n$, let $i = i + 1$ and go to Step 4.
Step 7: If $t < K$, let $t = t + 1$ and go to Step 5.
Stop: Stop the algorithm with X_α as the solution.

4 Experimental Results

In this section, the proposed method is evaluated on a set of experimental data. The input data set was obtained from [25]. It is related to the diagnosis of diabetes mellitus and taken from a population in India. We used a portion of this data set with 120 samples including 96 samples for training and 24 samples for testing. There are eight attributes in this data set which have been taken as features for training in our study listed in Table 2.

We consider two classes of patients or tested positive for diabetes and healthy or tested negative for diabetes labeled respectively by 1 and -1. In the training data, there are 36 member in class 1 while the class -1 has 33 member. In the test data, there are, respectively, 8 and 16 cases in these classes. This data is used as input data for the classification of cases as patient or healthy.

The GWO algorithm has been implemented of the dual problem with parameters as given in Table 3. It is evident that the GWO-based method show accurate results.

Table 2 Data features of the experiment

Number of times pregnant
Plasma glucose concentration
Diastolic blood pressure (mm Hg)
Triceps skin fold thickness (mm)
2-Hour serum insulin (mu U/ml)
Body mass index (kg/m^2)
Diabetes pedigree function
Age (years)

Table 3 Parameters of GWO and model

Parameter	Value
Number of search agents	30
Maximum number of iterations	100
Number of runs for a single case	3
Allowed variables range	[0, 5]
σ	[0.1, 15]

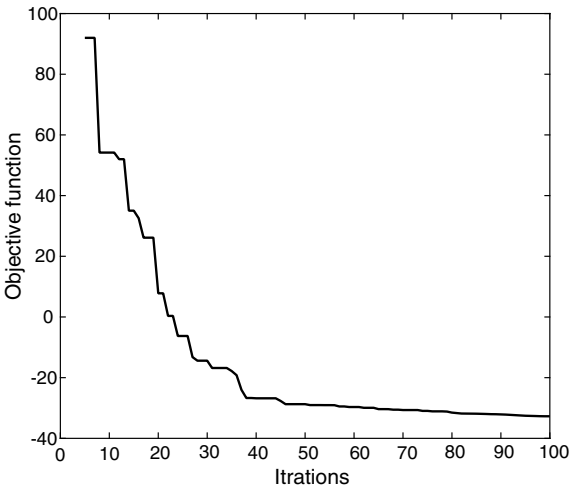


Fig. 7 The negative of the objective function in terms of iterations

Because of the stochastic nature of the method, the program executed three times for each single case. Then the best solution has been chosen as the optimum.

Since the GWO is developed for minimization, we had to minimize the negative of the objective function instead of maximization the original objective function. For example, Fig. 7 shows the curve of the negative value of objective function in terms of iterations that indicates the high rate of convergence.

A parameter sensitivity analysis of σ has been also performed. In this analysis, the values of σ changes from 0.1 to 15 and it was observed that the low accuracy occurs with $\sigma < 1$ and this range is not suggested to use. The best solution has been obtained for $\sigma = 10.0$ with the objective function equals to 33.04 and error in satisfying the constraint is of order 10^{-4}. In this optimum case, the accuracy of the method in the correct prediction of the remaining 24 cases was 87.5%. Other suitable values of σ leads to 79.2% and 83.3% correct prediction.

The curve in Fig. 8 demonstrates the percentage of correct prediction of the models in terms of the related value of σ. It is dedicated that the accuracy of the models rise with the value of sigma up to $\sigma = 10$ and then it falls again. Therefore, the optimum value is in this interval and may be estimated more precisely by a line search method.

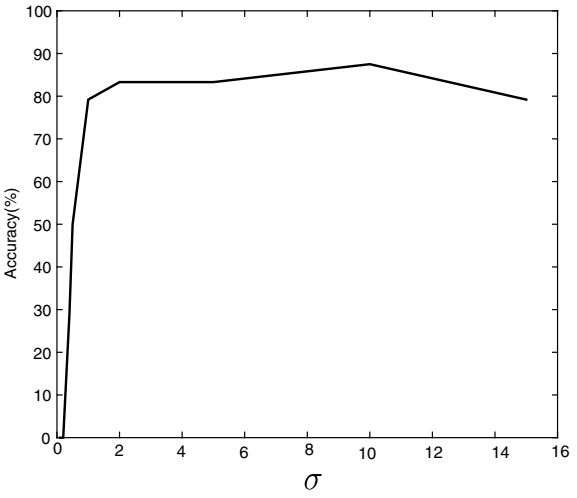


Fig. 8 The accuracy of the model in prediction in terms of the parameter values

5 Conclusion and Remarks

A method for determining the nonlinear support vector machine was presented. The method is based on using gray wolf optimization and parameter analysis. Results in a set of experimental data show that the method is accurate and reliable. For further investigation, using a line search algorithm such as the golden section method to find the optimal value of the model parameter is advised. Moreover, the study of the effect of other method parameters such as number of search agents, the maximum number of iterations, and variable interval may be performed.

References

1. Abdelghafar S, Darwish A, Hassanien AE (2019) Cube satellite failure detection and recovery using optimized support vector machine. In: Hassanien A, Tolba M, Shaalan K, Azar A (eds) Proceedings of the international conference on Advanced Intelligent Systems and Informatics 2018. AISI (2018) Advances in Intelligent Systems and Computing, vol 845. Springer, Cham
2. Al-Zoubi AM, Faris H, Alqatawna J, Hassonah MA (2018) Evolving support vector machines using Whale optimization algorithm for spam profiles detection on online social networks in different lingual contexts. Knowl-Based Syst 153:91–104
3. Belaid S, Mellit A (2016) Prediction of daily and mean monthly global solar radiation using support vector machine in an arid climate. Energ Convers Manage 118:105–118
4. Chan WH, Mohamad MS, Deris S, Zaki N, Kasim S, Omatu S, Corchado JM, Al Ashwal H (2016) Identification of informative genes and pathways using an improved penalized support vector machine with a weighting scheme. Comput Biol Med 77:102–115
5. Dalian Y, Yilun L, Songbai L, Xuejunc L, Liyong M (2015) Gear fault diagnosis based on support vector machine optimized by artificial bee colony algorithm. Mech Mach Theory 90:219–229

6. Deo RC, Kisi O, Singh VP (2017) Drought forecasting in eastern Australia using multivariate adaptive regression spline, least square support vector machine and M5Tree model. Atmos Res 184:149–175
7. Devi Arockia Vanitha C, Devaraj D, Venkatesulu M (2015) Gene expression data classification using support vector machine and mutual information-based gene selection. Proc Comput Sci 47:13–21
8. Elhariri E, El-Bendary N, Hassanien AE, Abraham A (2015) Grey wolf optimization for one-against-one multi-class support vector vachines. In: 7th international conference of Soft Computing and Pattern Recognition (SoCPaR), pp 7–12
9. Jaafari A, Pourghasemi HR (2019) Factors influencing regional-scale wildfire probability in Iran: an application of random forest and support vector machine. In: Spatial modeling in GIS and R for Earth and environmental sciences, pp 607–619
10. Joachims T (1998) Text categorization with support vector machines: learning with many relevant features. In: Ndellec C, Rouveirol C (eds) Machine learning: ECML-98. ECML 1998. Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence), vol 1398, pp 137–142
11. Kaytez F, Cengiz Taplamacioglu M, Camb E, Hardalac F (2015) Forecasting electricity consumption: a comparison of regression analysis, neural networks and least squares support vector machines. Int J Elec Power 67:431–438
12. Khedher L, Ramrez J, Grriz JM, Brahim A, Segovia F (2015) Early diagnosis of Alzheimers disease based on partial least squares, principal component analysis and support vector machine using segmented MRI images. Neurocomputing 151:139–150
13. Kisi O (2015) Pan evaporation modeling using least square support vector machine, multivariate adaptive regression splines and M5 model tree. J Hydrol 528:312–320
14. Li Y, Xu M, Wei Y, Huang W (2016) A new rolling bearing fault diagnosis method based on multiscale permutation entropy and improved support vector machine based binary tree. Energy 67:80–94
15. Meng J, Luo G, Gao F (2016) Lithium polymer battery state-of-charge estimation based on adaptive unscented Kalman filter and support vector machine. IEEE T Power Electr 31:2226–2238
16. Min SH, Lee J, Han I (2006) Hybrid genetic algorithms and support vector machines for bankruptcy prediction. Expert Syst Appl 31:652–660
17. Mirjalili SA, Mirjalili SM, Lewis A (2014) Grey Wolf optimizer. Adv Eng Softw 69:46–61
18. Mirjalili S, Aljarah I, Mafarja M, Heidari AA, Faris H (2020) Grey Wolf optimizer: theory, literature review, and application in computational fluid dynamics problems. In: Mirjalili S et al (eds) Nature-inspired optimizers, studies in computational intelligence, vol 811, pp 87–105
19. Mohammadi K, Shamshirband S, Tong CW, Arif M, Petkovic D, Ch S (2015) A new hybrid support vector machinewavelet transform approach for estimation of horizontal global solar radiation. Energ Convers Manage 92:162–171
20. Olatomiwa L, Mekhilef S, Shamshirband S, Mohammadi K, Petkovic K, Ch S (2015) A support vector machinefirefly algorithm-based model for global solar radiation prediction. Sol Energy 115:632–644
21. Sacchet MD, Prasad G, Foland-Ross LC, Thompson PM, Gotlib IH (2015) Support vector machine classification of major depressive disorder using diffusion-weighted neuroimaging and graph theory. Front Psychiatry 6:1–10
22. Shankar K, Lakshmanaprabu SK, Gupta D, Maseleno A, de Albuquerque VHC (2018) Optimal feature-basedmulti-kernel SVM approach for thyroid disease classification. J Supercomput 1573–0484:1–16
23. Sheng H, Xiao J (2015) Electric vehicle state of charge estimation: nonlinear correlation and fuzzy support vector machine. J Power Sources 281:131–137
24. Sherin BM, Supriya MH (2015) Selection and parameter optimization of SVM kernel function for underwater target classification. In: 2015 IEEE Underwater Technology (UT) Chennai, India, pp 1–5

25. Smith JW, Everhart JE, Dickson WC, Knowler WC, Johannes RS (1988) Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In: Proceedings of symposium on computer applications and medical care, pp 261–265
26. Soualhi A, Medjaher K, Zerhouni N (2015) Bearing health monitoring based on Hilbert-Huang transform, support vector machine and regression. IEEE T Instrum Meas 64:52–62
27. Sweidan AH, El-Bendary N, Hassanien AE, Hegazy OM, Mohamed AE (2015) Water quality classification approach based on bio-inspired gray wolf optimization. In: 7th international conference of Soft Computing and Pattern Recognition (SoCPaR), pp 1–6
28. Taravat A, Del Frate F, Cornaro C, Vergari S (2015) Neural networks and support vector machine algorithms for automatic cloud classification of whole-sky ground-based images. IEEE Geosci Remote S 12:666–670
29. Tong S, Koller D (2001) Support vector machine active learning with applications to text classification. J Mach Learn Res 2:45–66
30. Wang S, Lu S, Dong Z, Yang J, Yang M, Zhang Y (2016) Dual-tree complex wavelet transform and twin support vector machine for pathological brain detection. Appl Sci 6:1–18
31. Wu CH, Tzeng GH, Goo YJ, Fang WC (2007) A real-valued genetic algorithm to optimize the parameters of support vector machine for predicting bankruptcy. Expert Syst Appl 32:397–408
32. Yang XS, Deb S, Fong S (2011) Accelerated particle swarm optimization and support vector machine for business optimization and applications. In: Fong S (eds) Networked Digital Technologies (NDT), Communications in computer and information science, vol 136. Springer, Heidelberg
33. Zhang X, Chen X, He Z (2010) An ACO-based algorithm for parameter optimization of support vector machines Expert Sys Appl 37:6618–6628
34. Zhang Y, Dong Z, Wang S, Ji G, Yang J (2015) Preclinical diagnosis of Magnetic Resonance (MR) brain images via discrete wavelet packet transform with Tsallis entropy and Generalized Eigenvalue Proximal Support Vector Machine (GEPSVM). Entropy 17:1795–1813
35. Zhang YD, Yang ZJ, Lu HM, Zhou XX, Phillips P, Li QM, Wang SH (2016) Facial emotion recognition based on biorthogonal wavelet entropy, fuzzy support vector machine, and stratified cross validation. IEEE Access 4:8375–8385
36. Zhao D, Liu H, Zheng Y, He Y, Lu D, Lyu C (2019) Whale optimized mixed kernel function of support vector machine for colorectal cancer diagnosis. J Biomed Inform 92:103124
37. Zheng B, Myint SW, Thenkabail PS, Aggarwal RM (2015) A support vector machine to identify irrigated crop types usingtime-series Landsat NDVI data. Int J Appl Earth Obs 34:103–112

Efficient Moth-Flame-Based Neuroevolution Models

Ali Asghar Heidari, Yingyu Yin, Majdi Mafarja, Seyed Mohammad Jafar Jalali, Jin Song Dong and Seyedali Mirjalili

Abstract This chapter proposes a new efficient moth-flame-embedded multilayer perceptrons (MLP) neuroevolution model to deal with classification problems. Moth-flame optimizer (MFO) is one of the effective swarm-based metaheuristic methods inspired by the natural direction-finding behaviours of moth insects and their well-known entrapment phenomena when they circulate the non-natural lights and flames. MFO is capable of demonstrating a very promising performance in terms of exploration and exploitation inclinations. The proposed MFO-MLP model is exten-

A. A. Heidari
School of Surveying and Geospatial Engineering, College of Engineering, University of Tehran, Tehran, Iran
e-mail: as_heidari@ut.ac.ir; aliasgha@comp.nus.edu.sg; t0917038@u.nus.edu

A. A. Heidari · Y. Yin · J. S. Dong
Department of Computer Science, School of Computing, National University of Singapore, Singapore, Singapore
e-mail: yinyingyu@gpnu.edu.cn; yinyingyu@comp.nus.edu.sg; t0917011@u.nus.edu

Y. Yin
Department of Electronics and Information Engineering, Guangdong Polytechnic Normal University, Guangzhou 51665, China

M. Mafarja
Department of Computer Science, Faculty of Engineering and Technology, Birzeit University, PoBox 14, Birzeit, Palestine
e-mail: mmafarja@birzeit.edu

S. M. J. Jalali
Intelligent Systems Research and Innovations (IISRI), Deakin University, Waurn Ponds, VIC 3216, Australia
e-mail: sjalali@deakin.edu.au

J. S. Dong
Institute for Integrated and Intelligent Systems, Griffith University, Nathan Brisbane, Australia
e-mail: dongjs@comp.nus.edu.sg; j.dong@griffith.edu.au

S. Mirjalili (✉)
Torrens University Australia, Brisbane, QLD 4006, Australia
e-mail: ali.mirjalili@gmail.com

Griffith University, Brisbane, QLD 4111, Australia

S. Mirjalili et al. (eds.), *Evolutionary Machine Learning Techniques*, Algorithms for Intelligent Systems, https://doi.org/10.1007/978-981-32-9990-0_4

sively substantiated on 16 benchmark datasets, and the results are compared to well-known methods such as particle swarm optimizer (PSO), population-based incremental learning (PBIL), differential evolution (DE), and genetic algorithm (GA). The obtained results indicate the efficacy of the MFO-embedded neuroevolution model as a potential method in dealing with classification cases.

Keywords Moth-flame optimization · Optimization · Multi-layer perceptron · Neural networks · Artificial intelligence · Machine learning · Data science

1 Introduction

Artificial neural networks (ANNs) [1] are widely utilized for approximating functions and learning from available patterns in datasets, especially in dealing with classification tasks. Multilayer perceptron (MLPs) are the most well-regarded class of ANNs in previous studies [2]. Referring to Kolmogorov's theorem published in 1989 [3], it is interesting that a simple MLP with a single hidden layer has this ability to estimate any kind of continuous problems. However, the results of MLPs depend on the learning technique that we use to train the model [4]. Therefore, evolutionary algorithms (EA) can be embedded in training process of MLP-based models to enhance the chance of LO avoidance and mitigate the stagnation problems.

In the last decade, swarm-based metaheuristic algorithms have fascinated many researchers in different fields such as pattern recognition, computer vision, machine learning, robotics, and operational research owing to their efficiency and flexibility in tackling larger-scale, dynamic problems, time-consuming, and complex clustering classification and learning tasks during a reasonable time [5]. In this regard and during the contemporary digital era, the bio-inspired optimizer has progressively got prominence in the aforementioned fields. Furthermore, the human is witnessing a rapid information explosion, including spatial, temporal, uncertain, mobile, and spatio-temporal data in every second and in every location. Often, the new problems are NP-hard and high-dimensional. Hence, detecting the optimum in an n-dimensional hyperspace gradually becomes a very difficult task if it is practical and possible to detect, due to increasing complexity, and the tremendous and active scope of feasible solutions. In this context, both enhanced and conventional optimizers are capable of learning and disclosing a feasible solution in dealing with many computationally complex tasks during a reasonable time [6].

Stochastic optimizers have gained increasing attention in these years [7]. The reason is that these methods have shown promising results for many real-world scenarios [8]. In addition, they are simple to understand, and any user can develop a straightforward code for the targeted problem [9]. Some of the well-known and recently modified methods are differential evolution (DE) [9] and particle swarm optimization (PSO) [8].

Each of these algorithms has its own advantages and weaknesses. But, majority of them may fall into local optima (LO) in dealing with some problems [10]. However,

this is expected, and it is convenient to modify an optimizer with regard to the nature of different problems. Therefore, there are many efforts to develop more evolutionary training methods. Several works focused on the benefits of methods such as DE [11] and biogeography-based optimizer (BBO) [12].

Among the recent methods, MFO has attracted a lot of researchers due to its superior efficacy in realizing a lot of optimization problems [13]. The basic MFO algorithm [14] is inspired by the navigation tactics of the insects called moths in nature. They perform transverse orientation every time, which can be used to develop a good model for solving optimization problems. This method has shown a satisfactory performance on many unconstrained and constrained cases. Because of the good performance of MFO, it has been applied to many cases. In this chapter, we review some of the main contributions of MFO. Zhao et al. [15] used MFO within a hybrid electricity consumption forecasting approach. Sayed et al. [16] developed an intelligent mitosis detection technique by using neutrosophic concepts and the MFO. Allam et al. [17] proposed using the MFO to detect the unknown parameters of the three diode model. Li et al. [18] applied the MFO to optimize the parameters of the least squares support vector machine (SVM). Ng Shin Mei et al. [19] investigated the performance of the MFO in tackling the optimal reactive power dispatch problem (ORPD). Li et al. [20] built a diagnostic technique to predict the diagnosis of tuberculous pleural effusion. They used MFO to find the core parameters of SVM. Aziz et al. [21] applied both the WOA and MFO to detect the optimal threshold values of images. The second group of works tried to enhance the operators of the basic MFO. Li et al. [22] used Levy flight to enhance the MFO (LMFO). Trivedi et al. [23] used the MFO together with the Levy flight. Hassanien et al. [24] improved the basic MFO to spontaneously sense the diseases of tomato. Wang et al. [25] developed a new model based on the kernel extreme learning machine (KELM) and simultaneous application of the chaos-induced MFO. Zhang et al. [26] proposed an evolutionary FA-based method in which the spiral operator of MFO and the Levy flight are used, and finally, they used their method for feature selection cases. Apinantanakon et al. [27] evaluated the opposition-based learning technique to be embedded into the MFO for speeding up the convergence rate. Li et al. [28] used multi-objective MFO to improve the performance of the water resource cases. They also used opposition-based learning within the optimization core. Khalilpourazari et al. [29] used both MFO and the water cycle algorithm (WCA) to realize the optimal solutions of the constrained problems. Elsakaan et al. [30] proposed an improved MFO (EMFO) for dealing with the non-convex economic dispatch cases. Although some studies also utilized MFO to evolve their neural model [31, 32], we evaluate it in a more extensive way to provide new insights and findings.

2 Structure of MFO

As a recent method, MFO mimics the glamorized celestial transverse bearings and navigation manoeuvres of fancy moth creatures in sky [14]. The moths can consider

a star or moonshine to fix their direction and attain a straight trajectory for a long flight in the night-time. In the case of observing non-natural lights, the moths will be trapped in a tragic spiral-form trajectory around them. The central reason for these illogical spiral motions is the intrinsic flight-to-light behaviours of moths. To model these behaviours, the MFO considers an initial random swarm of moths as the possible solutions for the problem. Similar to other SMA, a group of moths, which have the role of search agents, can check and scan the surroundings (search space) to discern a sequence of flames (the best locations).

In the basic MFO, the location of moths should be generated using

$$M_i = S(M_i, F_j) \tag{1}$$

where M_i is the position vector of i-th moth, F_j denotes the j-th flame, and S is a function to generate logarithmic spiral trajectories, which can be formulated as

$$S(M_i, F_j) = D_i \mathrm{e}^{bz} \cos(2\pi z) + F_j \tag{2}$$

where b is set to 1 in the MFO, z is a random value in $[-1, 1]$, and D_i is the distance among moths and flames determined by

$$D_i = \left| F_j - M_i \right| \tag{3}$$

In the MFO, the number of flames (N_F) should be adaptively reduced using

$$N_F = \text{round}\left\{N - t \times (N - 1/T)\right\} \tag{4}$$

where t is the iteration, N is the total number of flames, and T is the maximum iterations.

The succeeding clarifications summarize some of the exploration and exploitation advantages of the MFO:

- The utilized spiral motions assist the MFO in exploiting the vicinity of superior flames.
- Each moth is allocated to a flame, and then all flames are updated in each step. This tactic can intensify the exploration tendency of the MFO, and it supports it in the situation of LO entrapment.
- The MFO uses the fresh best agents attained so far as the flames. The manner keeps the superior agents as team leaders for the rest of moths.
- The MFO implements an adaptive function to decrease the leaders (flames). This technique is very constructive to recover a fine balance between exploitation and exploration.

The pseudo-code of MFO is shown in the algorithm 1.

Algorithm 1 Pseudo-code of MFO technique

Input: Total number of moths and iterations (T_{max}).
Output: The best solution and the its fitness value.
Initialize positions of moths $x_i (i = 1, 2, \ldots, n)$
Obtain the fitness of moths.
while (Looping condition is not met) **do**
  Update flame no. based on Eq. (4)
  Define: OM = Fitness Function(M)
  if $(i == 1)$ **then**
    F = sort(M), OF=sort(OM)
  else
    F = sort(M_{t-1}, M_t), OF = sort(M_{t-1}, M_t);
  end if
  for $i = 1 : n$ **do**
    for $j = 1 : d$ **do**
      Update r and t
      Obtain D by Eq. (3) with regard to the related moth
      Update $M(i, j)$ via Eqs. (1) and (2) with regard to the related moth
    end for
  end for
end while
Return the best solution

3 Foundation of MLP

FNNs are a family member of ANN models that are widely used for classification tasks [4]. The neurons are the sets of processing elements that are considered as the backbone block of each FNN and settled into different layers. The most popular model for training of FNNs is known as multilayer perceptron (MLP) algorithm. A sample structure of an MLP is illustrated in Fig. 1.

The output of each neuron in MLPs is determined by two phases: calculation of summation and activation functions. The first phase is the calculation of weighted summation of the input layer and bias terms using Eq. (5) :

$$S_j = \sum_{i=1}^{n} \omega_{ij} I_i + \beta_j, \tag{5}$$

where I_i is the input variable, n denotes the number of input variables, w_{ij} represents the weights linking neurons of input layer to the hidden layer, and β_j is the bias of the network.

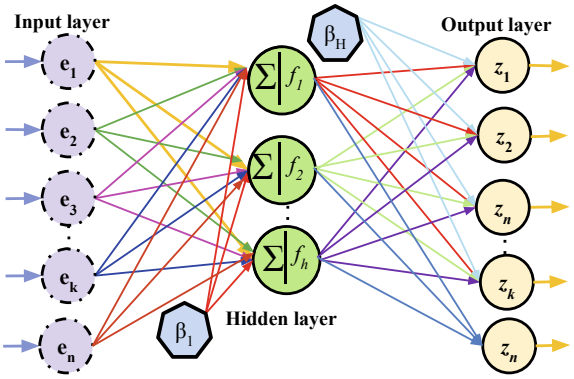


Fig. 1 Example of MLP network based on design of Heidari et al. [33]

The activation function is the second step should be addressed by using the output of neurons as shown in Eq. (5). Several different forms of activation functions are proposed for training MLPs. The most common method used in the literature is the S-shaped curved sigmoid function as shown in Eq. (6):

$$f_j(x) = \frac{1}{1 + e^{-S_j}} \tag{6}$$

As a result, the concluding output of the network can be calculated as given in Eq. (7):

$$y_i = f_j\left(\sum_{i=1}^{n} \omega_{ij} I_i + \beta_j\right) \tag{7}$$

After designing the structure of MLP, the process of tuning and updating network weights is performed by utilizing a training technique to justify the outcomes and minimization of the errors of the network.

4 Structure of MFO-embedded MLPs

This section represents a detailed description for training process of MLP network using MFO-based optimizer algorithm. This approach is named MFO-MLP and is applied on a single hidden layer of MLP network in this study. To achieve this goal, two key aspects are taken into consideration for constructing MFO-MLP algorithm: the encoding representation of individuals (interaction between moths and flames) in the MFO algorithm and selecting the formulation of the fitness function. In MFO-MLP, all individuals are encoded as one-dimensional vectors of random real numbers inside the interval $[-1, 1]$. Each generated solution by the encoding schema repre-

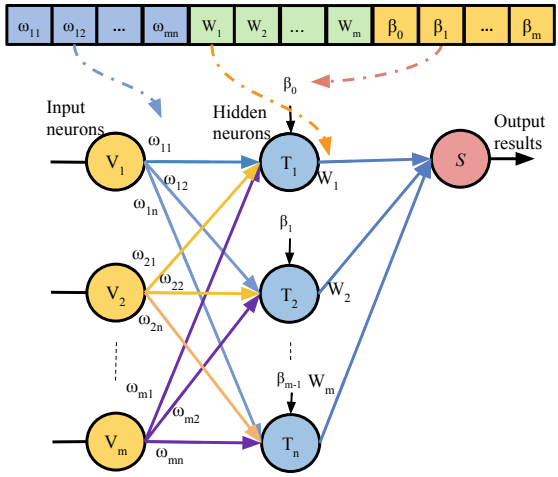


Fig. 2 Solutions's structure in MFO-MLP based on design of Heidari et al. [33]

sents an MLP candidate. The designed vectors include three key parts: a set of weights connecting the input layer to hidden layer, the connection weights between the hidden layer and the output layer, and a set of bias weights. The structure of agents in the proposed MFO-MLP is shown in Fig. 2.

To evaluate the fitness value of MFO-MLP approach, the vector of biases and weights is passed to the MLP network. In this work, the mean squared error (MSE) is utilized as the fitness function. This evaluation metric calculates the difference between the actual and predicted values obtained by the generated individuals (MLPs) using training samples. MSE metric is attained by Eq. (8):

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(z_i - \hat{z}_i)^2, \tag{8}$$

where z and $\hat{z}$ are the actual and predicted values, and n indicates the number of instances in the dataset.

The workflow of the MFO-based MLP algorithm is shown in Fig. 3.

5 Results and Discussions

In this section, the results of MFO-MLP model are evaluated using 16 well-regarded datasets (see Table 2). These problems have been subject to many studies and can show the real potential of MLP-based models. They have several features with various instances and characteristics that make them more attracting to benchmark classifiers and learning models. Note that these problems are also publicly available at UCI machine learning repository [34]. This open access also is beneficial because other

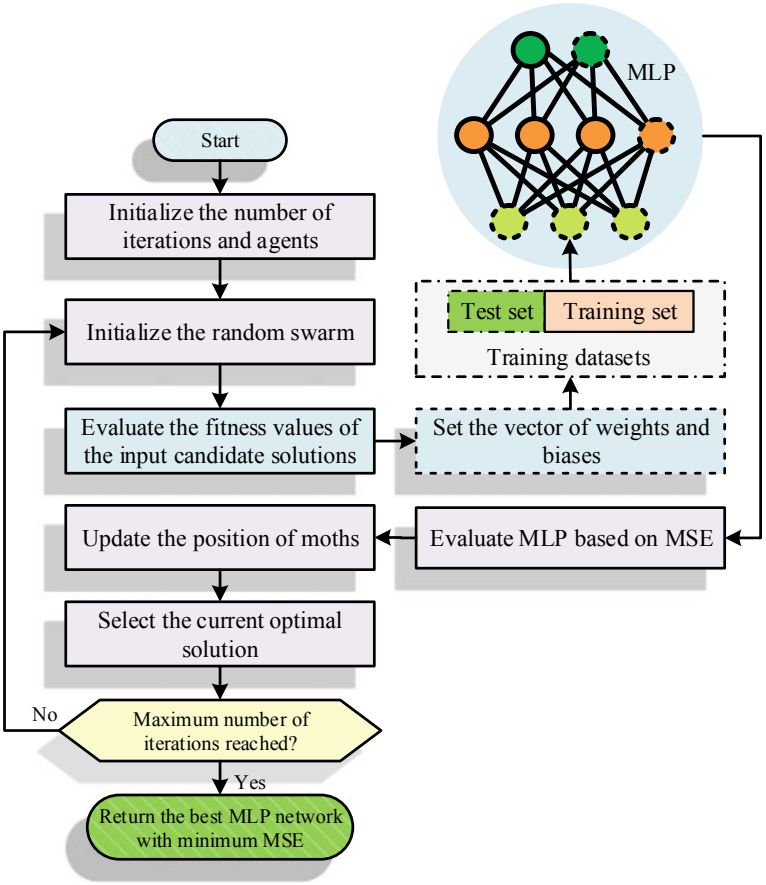


Fig. 3 Flowchart of the proposed MFO-based model

MLP-based models can also substantiate their performances and then compare the results with previously available models. However, in this study, we performed all methods simultaneously to ensure fair comparison and similar initial population. Details of all used datasets are further described in detail on Table 2.

One of the important stages for validating the evolutionary methods is the choice of other compared methods. First, they should be comparable, and the comparison should be fair in different aspects. Second, they should be able to provide competitive results. Hence, in this section, we used GA [35], PSO [36], DE [37], and population-based incremental learning (PBIL) [12] optimizers to substantiate the efficacy of the proposed MFO-based neuroevolution model. There are several reasons that we selected these methods from a big set of available optimizers: first, these methods are almost implemented for any kind of problems (constrained, unconstrained, single-objective, multi-objective, etc.) and stable performance and results reported. Hence,

there is no doubt about their core abilities in solving optimization problems. Second, we selected these methods because almost any audience in the field of optimization and machine learning is familiar with at least one of them. Third, the parameter's settings of these algorithms are clear and accurate, and there is no origin bias or lack of clarity in the structure of these well-established methods.

We performed all experiments in the same computing server, in which the details are reported in Table 1. Note that all experiments in this study follow the settings and runs in previous works by Heidari et al. [33, 38]. The experiments in this chapter reach the same results reported in [33] because of the same parameters and computing conditions.

Table 1 Detail of the computing system

Name	Settings
Hardware	
CPU	1.70 Ghz
Frequency	2.70 GHz
RAM	64 GB RAM
Hard drive	500 GB
Software	
Operating system	Windows 7 (64-bit) Windows server 2012
Language	Matlab 7.10 (R2010a)

Table 2 Summary of datasets [33]

Dataset	Features	Training samples	Testing samples
Bank	48	2261	2260
Ionosphere	33	231	120
Chess	36	2109	1087
Australian	14	455	235
Credit	61	670	330
Spam base	57	2300	2301
Breast	8	461	238
Mammographic-mass	5	633	328
Monk	6	285	147
Sonar	60	137	71
Ring	20	4884	2516
Phoneme	5	3566	1838
Tic-Tac-Toe	9	632	326
Titanic	3	1452	749
Twonorm	20	4884	2516
Wdbc	30	375	194

Table 3 Used parameters of optimizers

Algorithm	Parameter	Value
DE	Crossover rate	0.9
	Differential weight	0.5
	Number of agents	50
GA	Crossover rate	0.9
	Mutation rate	0.1
	Selection method	Random sampling
	Number of agents	50
	Number of iterations	200
PBIL	Learning ratio	0.05
	Fit swarm member	1
	Not-fit swarm member	0
	Elitism rate	1
	Mutational rate	0.1
PSO	Inertia weights	[0.9, 0.6]
	Acceleration factors	[2.1, 2.1]
	Number of agents	50

Another important aspect in evaluating the evolutionary approaches is how we set the initial parameters. Note that we set the parameters of optimizers based on two observations: primary tests and original papers. Details of settings are presented in Table 3.

Table 4 exposes the average accuracy rates of MFO-MLP and standard deviations (STD) of this method compared to the results attained by other competitors. Table 5 also reveals the ranking of all optimizers based on average accuracy results presented in Table 4.

From Table 4, we see that MFO outperforms other well-regarded peers such as DE, GA, PSO, and PBIL on Australian, bank, breast, chess, credit, ionosphere, mammographic-mass, monk, phoneme, ring, sonar, spambase, and twonorm datasets; whereas, MFO can show better results on 81.25% of cases, and it can be detected that PBIL has achieved the second best stage and can obtain satisfactory and very competitive rates only on two datasets. We also see that PSO and DE cannot show the best rates for any of the datasets.

As per ranks in Table 5, we see the MFO is the best optimizer, followed by PBIL, GA, PSO, and DE, respectively. We see that DE cannot show superior performance in dealing with any of the datasets, which again remind us the conclusion of no free lunch (NFL) theorem [39] that indicates several conclusions: first, a good optimizer is not always the best for other problems, and there will be better competitors on new cases. Second, new optimizers such as MFO can obtain better results on the majority of cases, while it is expected to see also some worst results on other classes of problems. The main fact is that the average performance on all classes of problems

Table 4 Average accuracy results of MFO-MLP versus DE-MLP, PSO-MLP, PBIL-MLP, and GA-MLP methods

Datasets	Metric	MFO-MLP	DE-MLP	PSO-MLP	PBIL-MLP	GA-MLP
Australian	avg.	**8.538E−01**	7.588E−01	8.178E−01	8.254E−01	8.205E−01
	std.	4.514E−02	5.102E−02	2.182E−02	1.538E−02	2.303E−02
Bank	avg.	**9.025E−01**	8.850E−01	8.854E−01	8.879E−01	8.937E−01
	std.	8.223E−02	4.733E−03	4.730E−03	3.688E−03	2.724E−03
Breast	avg.	**9.821E+01**	9.391E−01	9.646E−01	9.661E−01	9.723E−01
	std.	2.540E−03	1.964E−02	9.788E−03	7.797E−03	6.203E−03
Chess	avg.	**7.356E−01**	6.263E−01	6.885E−01	7.051E−01	6.149E−01
	std.	8.112E−02	2.832E−02	2.567E−02	2.270E−02	7.709E−02
Credit	avg.	**7.141E−01**	6.917E−01	6.989E−01	7.069E−01	7.011E−01
	std.	1.580E−02	2.128E−02	2.136E−02	2.128E−02	1.685E−02
Ionosphere	avg.	**7.852E−01**	7.331E−01	7.614E−01	7.797E−01	7.556E−01
	std.	3.745E−02	5.616E−02	4.390E−02	3.430E−02	4.240E−02
Mammographic-mass	avg.	**7.995E−01**	7.939E−01	7.917E−01	7.882E−01	7.934E−01
	std.	2.553E−03	1.711E−02	1.169E−02	1.325E−02	5.412E−03
Monk	avg.	**8.141E−01**	7.120E−01	7.732E−01	7.732E−01	8.079E−01
	std.	8.223E−02	5.562E−02	3.433E−02	3.410E−02	2.111E−02
Phoneme	avg.	**7.692E−01**	7.395E−01	7.564E−01	7.584E−01	7.585E−01
	std.	6.855E−03	2.451E−02	9.628E−03	1.597E−02	1.135E−02
Ring	avg.	**7.254E−01**	6.499E−01	6.905E−01	7.124E−01	6.990E−01
	std.	5.223E−02	2.227E−02	2.596E−02	1.588E−02	3.015E−02
Sonar	avg.	**6.895E−01**	6.150E−01	6.291E−01	6.723E−01	5.869E−01
	std.	2.550E−02	7.213E−02	6.723E−02	5.940E−02	7.538E−02
Spambase	avg.	**7.683E−01**	6.566E−01	7.333E−01	7.322E−01	7.656E−01
	std.	3.652E−02	4.878E−02	2.748E−02	2.831E−02	2.499E−02
Tictac	avg.	6.322E−01	6.070E−01	6.304E−01	**6.442E−01**	6.323E−01
	std.	1.234E−02	2.671E−02	3.372E−02	2.744E−02	1.994E−02
Titanic	avg.	7.631E−01	7.623E−01	7.635E−01	**7.646E−01**	7.621E−01
	std.	5.290E−03	7.649E−03	6.260E−03	6.691E−03	5.033E−03
Twonorm	avg.	**9.810E−01**	8.062E−01	9.039E−01	9.257E−01	9.735E−01
	std.	3.220E−02	4.021E−02	1.963E−02	1.319E−02	2.271E−03
Wdbc	avg.	9.427E−01	8.617E−01	9.270E−01	9.246E−01	**9.445E−01**
	std.	5.523E−02	3.786E−02	1.852E−02	1.343E−02	1.779E−02

will be similar. However, the third fact is that we can develop a special optimizer for special cases of problems like MLP-based classification cases. However, we should make it clear that the MFO is not an exception, and it also follows the general NFL conditions.

Details of all experiments are shown in Table 6.

In Table 6, we reported best, worst, and median of all optimizers in dealing with all problems. If we observe these results, we again see similar patterns that show the

Table 5 Ranks of algorithms based on accuracy rates

Dataset	MFO-MLP	DE-MLP	PSO-MLP	PBIL-MLP	GA-MLP
Australian	1	5	4	2	3
Bank	1	5	4	3	2
Breast	1	5	4	3	2
Chess	1	4	3	2	5
Credit	1	5	4	2	3
Ionosphere	1	5	3	2	4
Mammographic-mass	1	2	4	5	3
Monk	1	5	3	3	2
Phoneme	1	5	4	3	2
Ring	1	5	4	2	3
Sonar	1	4	3	2	5
Spambase	1	5	3	4	2
Tictac	3	5	4	1	2
Titanic	3	4	2	1	5
Twonorm	1	5	4	3	2
Wdbc	2	5	3	4	1
Average rank	1.3125	4.625	3.5	2.625	2.875
Sum of the ranks	21	74	56	42	46
Overall rank	1	5	4	2	3

MFO-MLP can outperform other competitors or provide very competitive results in almost all cases. As we see, the MFO-MLP model also has enhanced the median of solutions for most of the datasets.

There are several core reasons why MFO-MLP can show enhanced classification rates. The first reason is that the MFO-MLP has a more stable capability in balancing the exploratory and exploitative inclinations. When the MFO engine faces LO, it can successfully jump out of them and continue the gradual enhancements in quality. Second, MFO assigns a flame to each moth, which enhances the diversity of the population during the optimization. Any enhancement in diversity will result in a higher chance of avoiding LO and immature convergence problems. Third, in the MFO part of the MFO-MLP, a number of flames progressively decrease, which lead to a healthier equilibrium between exploration and exploitation tendencies. Any better balance between these trends will lead to improved results in terms of average and median of solutions as we observed in the results of MFO-MLP. Fourth, the proposed MFO-MLP approach utilizes an adaptive convergence parameter that can assist the base method in showing accelerated convergence propensities around the flames during more iteration. Last but not the least is that the best agents in MFO core are saved in the F matrix, and this means that they never get lost during the next steps of the exploration and exploitation. Actually, best agents will guide the

Table 6 Statistical measures of accuracy rates for all techniques

Algorithms	MFO-MLP			DE-MLP			PBIL-MLP			PSO-MLP			GA-MLP		
Dataset	MAX	MIN	MED.	MAX	MIN	MED.	MAX	MIN	MED.	MAX	MIN	MED.	MAX	MIN	MED.
Australian	8.73E−01	8.12E−01	8.33E−01	8.60E−01	6.58E−01	7.63E−01	8.51E−01	7.89E−01	8.29E−01	8.55E−01	7.81E−01	8.16E−01	8.64E−01	7.54E−01	8.22E−01
Ionosphere	8.46E−01	7.35E−01	7.82E−01	8.25E−01	5.92E−01	7.42E−01	8.50E−01	7.25E−01	7.79E−01	8.58E−01	6.67E−01	7.58E−01	8.58E−01	6.67E−01	7.58E−01
Mammo-graphic-mass	8.09E−01	7.90E−01	7.90E−01	8.17E−01	7.50E−01	7.99E−01	8.05E−01	7.50E−01	7.93E−01	8.17E−01	7.68E−01	7.90E−01	8.08E−01	7.84E−01	7.93E−01
Bank	9.22E−01	8.92E−01	9.13E−01	8.90E−01	8.67E−01	8.86E−01	8.94E−01	8.76E−01	8.88E−01	8.94E−01	8.73E−01	8.87E−01	8.99E−01	8.86E−01	8.94E−01
Breast	9.92E−01	9.75E−01	9.81E−01	9.71E−01	8.95E−01	9.41E−01	9.79E−01	9.54E−01	9.66E−01	9.79E−01	9.45E−01	9.66E−01	9.83E−01	9.58E−01	9.75E−01
Chess	8.16E−01	7.10E−01	7.35E−01	6.94E−01	5.81E−01	6.21E−01	7.53E−01	6.67E−01	6.99E−01	7.53E−01	6.41E−01	6.90E−01	7.75E−01	5.13E−01	6.00E−01
Phoneme	7.81E−01	7.41E−01	7.59E−01	7.87E−01	6.85E−01	7.41E−01	7.85E−01	7.13E−01	7.60E−01	7.86E−01	7.37E−01	7.57E−01	7.79E−01	7.36E−01	7.59E−01
Sonar	7.18E−01	6.10E−01	6.79E−01	7.61E−01	4.51E−01	6.20E−01	7.75E−01	5.21E−01	6.76E−01	7.75E−01	4.65E−01	6.48E−01	7.32E−01	4.65E−01	5.77E−01
Credit	7.35E−01	6.51E−01	7.00E−01	7.39E−01	6.45E−01	6.92E−01	7.58E−01	6.67E−01	7.06E−01	7.36E−01	6.36E−01	7.00E−01	7.18E−01	6.42E−01	7.09E−01
Ring	7.72E−01	7.11E−01	7.21E−01	7.02E−01	6.04E−01	6.52E−01	7.42E−01	6.76E−01	7.15E−01	7.41E−01	6.22E−01	6.94E−01	7.65E−01	6.52E−01	6.92E−01
Monk	8.41E−01	7.68E−01	8.01E−01	8.37E−01	5.92E−01	7.21E−01	8.50E−01	7.07E−01	7.82E−01	8.50E−01	7.01E−01	7.69E−01	8.44E−01	7.69E−01	8.03E−01
Spambase	8.13E−01	7.14E−01	7.59E−01	7.35E−01	5.61E−01	6.63E−01	8.01E−01	6.74E−01	7.41E−01	7.85E−01	6.81E−01	7.31E−01	7.92E−01	6.97E−01	7.75E−01
Twonorm	9.61E−01	9.15E−01	9.51E−01	8.76E−01	6.94E−01	8.03E−01	9.52E−01	9.01E−01	9.25E−01	9.45E−01	8.62E−01	9.08E−01	9.78E−01	9.69E−01	9.73E−01
Wdbc	9.72E−01	8.84E−01	9.41E−01	9.33E−01	7.68E−01	8.58E−01	9.54E−01	9.02E−01	9.25E−01	9.69E−01	8.81E−01	9.28E−01	9.85E−01	8.97E−01	9.43E−01
Tictac	7.12E−01	6.23E−01	6.63E−01	6.60E−01	5.49E−01	6.04E−01	6.99E−01	5.95E−01	6.40E−01	6.81E−01	5.49E−01	6.33E−01	6.66E−01	5.98E−01	6.37E−01
Titanic	7.69E−01	7.51E−01	7.62E−01	7.70E−01	7.38E−01	7.61E−01	7.78E−01	7.40E−01	7.66E−01	7.70E−01	7.46E−01	7.68E−01	7.70E−01	7.52E−01	7.61E−01

rest of agents towards more favourable zones of the feature space. Taking together, all these merits can be expected from the proposed MFO-MLP framework. From the other hand, methods such as PSO, DE, PBIL, and GA cannot show such behaviours, mathematically. One point that should not be overlooked is that any optimizer is a good alternative compared to some other candidates, but this will not remain always valid. It is possible to deal with some other classes of datasets with different nature and properties and see different results. It is obvious that all these methods are stochastic in nature and almost all of them cannot guarantee convergence to global best. Although the results may be different, the MFO will still show similar merits and searching patterns that can be beneficial in solving any kind of problems, especially in machine learning.

6 Conclusions and Future Directions

This chapter focused on the mathematical model of a new optimizer called MFO. As one of the initial studies, we utilized the exploration and exploitation phases of MFO to develop an efficient neuroevolution model for classification problems. This chapter provides the theoretical backgrounds required to develop an effective and simple neuroevolution model based on MLP. We presented an encoding approach and defined an objective function to be minimized. Then, we utilized the proposed MFO-based model to deal with 16 widely used datasets. The results show improved performance of the proposed model on mediocre and large datasets compared to models-based GA, PSO, DE, and PBIL techniques.

For future works, other metrics can be investigated. In addition, there are many high-dimensional datasets in real-world applications that can be used to further verify the efficacy of the developed model.

References

1. McCulloch WS, Pitts W (1943) A logical calculus of the ideas immanent in nervous activity. Bull Math Biophys 5:115–133
2. Chen J-F, Do QH, Hsieh H-N (2015) Training artificial neural networks by a hybrid pso-cs algorithm. Algorithms 8:292–308
3. Cybenko G (1989) Approximation by superpositions of a sigmoidal function. Math Control Signals Syst (MCSS) 2:303–314
4. Ojha VK, Abraham A, Snášel V (2017) Metaheuristic design of feedforward neural networks: a review of two decades of research. Eng Appl Artifi Intell 60:97–116
5. Valdez F, Melin P, Castillo O (2014) A survey on nature-inspired optimization algorithms with fuzzy logic for dynamic parameter adaptation. Exp Syst Appl 41:6459–6466
6. Heidari AA, Mirjalili S, Faris H, Aljarah I, Mafarja M, Chen H (2019) Harris hawks optimization: algorithm and applications. Future Gener Comput Syst 97:849–872
7. Luo J, Chen H, Heidari AA, Xu Y, Zhang Q, Li C (2019) Multi-strategy boosted mutative whale-inspired optimization approaches. Appl Math Modell 73:109–123

8. Deng W, Yao R, Zhao H, Yang X, Li G (2017) A novel intelligent diagnosis method using optimal ls-svm with improved pso algorithm. Soft Comput. https://doi.org/10.1007/s00500-017-2940-9
9. Storn R, Price K (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 11:341–359
10. Heidari AA, Aljarah I, Faris H, Chen H, Luo J, Mirjalili S (2019) An enhanced associative learning-based exploratory whale optimizer for global optimization. Neural Comput Appl
11. Mallipeddi R, Suganthan PN, Pan Q-K, Tasgetiren MF (2011) Differential evolution algorithm with ensemble of parameters and mutation strategies. Appl Soft Comput 11:1679–1696
12. Simon D (2008) Biogeography-based optimization. IEEE Trans Evol Comput 12:702–713
13. Xu Y, Chen H, Heidari AA, Luo J, Zhang Q, Zhao X, Li C (2019) An efficient chaotic mutative moth-flame-inspired optimizer for global optimization tasks. Exp Syst Appl 129:135–155
14. Mirjalili S (2015) Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm. Knowl Based Syst 89:228–249
15. Zhao H, Zhao H, Guo S (2016) Using GM (1, 1) optimized by MFO with rolling mechanism to forecast the electricity consumption of inner mongolia. Appl Sci (Switz) 6:1–18
16. Sayed GI, Hassanien AE (2017) Moth-flame swarm optimization with neutrosophic sets for automatic mitosis detection in breast cancer histology images. Appl Intell 47:397–408
17. Allam D, Yousri DA, Eteiba MB (2016) Parameters extraction of the three diode model for the multi-crystalline solar cell/module using moth-flame optimization algorithm. Energy Convers Manage 123:535–548
18. Li C, Li S, Liu Y (2016) A least squares support vector machine model optimized by moth-flame optimization algorithm for annual power load forecasting. Appl Intell 45:1166–1178
19. Ng Shin Mei R, Sulaiman MH, Mustaffa Z, Daniyal H (2017) Optimal reactive power dispatch solution by loss minimization using moth-flame optimization technique. Appl Soft Comput J 59:210–222
20. Li C, Hou L, Sharma BY, Li H, Chen C, Li Y, Zhao X, Huang H, Cai Z, Chen H (2018) Developing a new intelligent system for the diagnosis of tuberculous pleural effusion. Comput Methods Programs Biomed 153:211–225
21. Aziz MAE, Ewees AA, Hassanien AE (2017) Whale optimization algorithm and moth-flame optimization for multilevel thresholding image segmentation. Exp Syst Appl 83:242–256
22. Li Z, Zhou Y, Zhang S, Song J (2016) Levy-flight moth-flame algorithm for function optimization and engineering design problems. Math Probl Eng 2016:22
23. Trivedi IN, Kumar A, Ranpariya AH, Jangir P (2016) Economic load dispatch problem with ramp rate limits and prohibited operating zones solve using levy flight moth-flame optimizer. In: 2016 international conference on energy efficient technologies for sustainability, ICEETS 2016, pp 442–447
24. Hassanien AE, Gaber T, Mokhtar U, Hefny H (2017) An improved moth flame optimization algorithm based on rough sets for tomato diseases detection. Comput Electron Agric 136:86–96
25. Wang M, Chen H, Yang B, Zhao X, Hu L, Cai Z, Huang H, Tong C (2017) Toward an optimal kernel extreme learning machine using a chaotic moth-flame optimization strategy with applications in medical diagnoses. Neurocomput 267:69–84
26. Zhang L, Mistry K, Neoh SC, Lim CP (2016) Intelligent facial emotion recognition using moth-firefly optimization. Knowl Based Syst 111:248–267
27. Apinantanakon W, Sunat K (2018) Omfo: a new opposition-based moth-flame optimization algorithm for solving unconstrained optimization problems. Adv Intell Syst Comput 566:22–31
28. Li WK, Wang WL, Li L (2018) Optimization of water resources utilization by multi-objective moth-flame algorithm. Water Resour Manage 32:3303–3316
29. Khalilpourazari S, Khalilpourazary S (2019) An efficient hybrid algorithm based on water cycle and moth-flame optimization algorithms for solving numerical and constrained engineering optimization problems. Soft Computing 23(5):1699–1722
30. Elsakaan AA, El-Sehiemy RA, Kaddah SS, Elsaid MI (2018) An enhanced moth-flame optimizer for solving non-smooth economic dispatch problems with emissions. Energy 157:1063–1078

31. Yamany W, Fawzy M, Tharwat A, Hassanien AE (2015) Moth-flame optimization for training multi-layer perceptrons. In: 2015 11th international computer engineering conference (ICENCO). IEEE, pp 267–272
32. Majhi SK, Mahapatra P (2019) Classification of phishing websites using moth-flame optimized neural network. In: Emerging technologies in data mining and information security. Springer, pp 39–48
33. Heidari AA, Faris H, Mirjalili S, Aljarah I, Mafarja M (2020) Ant lion optimizer: theory, literature review, and application in multi-layer perceptron neural networks. Springer International Publishing, Cham, pp 23–46
34. Lichman M (2013) UCI machine learning repository. Retrieved from https://archive.ics.uci.edu/ml/index.php
35. Weile DS, Michielssen E (1997) Genetic algorithm optimization applied to electromagnetics: a review. IEEE Trans Antennas Propag 45:343–353
36. Lin S-W, Ying K-C, Chen S-C, Lee Z-J (2008) Particle swarm optimization for parameter determination and feature selection of support vector machines. Exp Syst Appl 35:1817–1824
37. Ilonen J, Kamarainen J-K, Lampinen J (2003) Differential evolution training algorithm for feed-forward neural networks. Neural Process Lett 17:93–105
38. Heidari AA, Faris H, Aljarah I, Mirjalili S (2018) An efficient hybrid multilayer perceptron neural network with grasshopper optimization. Soft Comput 1–18
39. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1:67–82

Autonomous Robot Navigation Using Moth-Flame-Based Neuroevolution

Seyed Mohammad Jafar Jalali, Rachid Hedjam, Abbas Khosravi, Ali Asghar Heidari, Seyedali Mirjalili and Saeid Nahavandi

Abstract Determining the best set of weights and biases for training neural networks (NN) using gradient descent techniques is a computationally challenging task. On the other hand, training of gradient descent algorithms suffers from being trapped in local optima and slow convergence speed in the last iterations. The moth-flame optimization (MFO) is a novel evolutionary method based on navigation paths of moths in nature. This algorithm showed its effectiveness in many real-world optimization problems. In this chapter, MFO is employed for training multilayer perceptron (MLP) to overcome the problems of gradient descent algorithms. This algorithm also investigates the application of the MFO for tackling the navigation of autonomous mobile robots. The results are compared with four powerful evolutionary algorithms including gray wolf optimizer (GWO), cuckoo search (CS), multiverse optimizer

S. M. J. Jalali (✉) · A. Khosravi · S. Nahavandi
Institute for Intelligent Systems Research and Innovation (IISRI),
Deakin University, Waurn Ponds, VIC 3216, Australia
e-mail: sjalali@deakin.edu.au

A. Khosravi
e-mail: abbas.khosravi@deakin.edu.au

S. Nahavandi
e-mail: saeid.nahavandi@deakin.edu.au

R. Hedjam
Sultan Qaboos University, Muscat, Sultanate of Oman
e-mail: rachid.hedjam@squ.edu.om

A. A. Heidari
School of Surveying and Geospatial Engineering, College of Engineering,
University of Tehran, Tehran, Iran

A. A. Heidari
Department of Computer Science, School of Computing, National University of Singapore,
Singapore, Singapore
e-mail: as_heidari@ut.ac.ir; aliasgha@comp.nus.edu.sg; t0917038@u.nus.edu

S. Mirjalili
Torrens University Australia, Fortitude Valley, Brisbane, QLD 4006, Australia
e-mail: ali.mirjalili@gmail.com

Griffith University, Brisbane, QLD 4111, Australia

S. Mirjalili et al. (eds.), *Evolutionary Machine Learning Techniques*,
Algorithms for Intelligent Systems, https://doi.org/10.1007/978-981-32-9990-0_5

algorithm (MVO), and particle swarm optimization (PSO). Moreover, the results are compared to two gradient-based training MLP algorithms including Levenberg–Marquardt (LM) and back-propagation (BP). The evaluation metrics used in this book chapter are accuracy and area under the curve (AUC). The experimental results show that MFO-based MLP algorithm outperforms other algorithms and showed its capabilities effectively.

Keywords Robot navigation · Neural network · Neuroevolution · Evolutionary algorithm · Moth-Flame optimization

1 Introduction

Advances in the field of ANN and machine learning have exploded in recent years. The excitement comes from the fact that ANN mimics the learning process of the human brain, which gives them the characteristic of being efficient in handling complex and nonlinear data. They are, therefore, ideal for modeling different real data.

ANNs are inspired by biological nervous systems. In general, ANN can be presented into two main architectures, recurrent, which involves feedback (loops) and nonrecurrent, which does not involve feedback [6]. ANNs differ not only in their architectures or structures but also in the learning process they use. They possess a relevant characteristic, namely the ability to learn from the input samples [2]. The learning process expresses the behavior of the network by finding a function that maps the inputs to the expected (correct) outputs. These can be expressed as a weighted combination of inputs.

The algorithm commonly used to estimate the optimal weights minimizing the error between the actual and predicted outputs is the gradient descent and its derivatives, in particular, back-propagation (BP). The latter is mainly used by deep learning architectures, the ANN, which obtained the state of the art in almost all applications. Nevertheless, gradient descent-based algorithms can result in a movement toward the local minimum [1]. Being stuck in the local minimum can be due to several reasons, in particular, those related mainly to the style of learning and the architectures of the networks used. This problem has been addressed in many papers, and it has been examined using different approaches, namely deterministic and probabilistic approaches. In the deterministic approaches, the primary descent technique is replaced by the global descent [9] and other related works addressing global optimization can be also found in the literature [9]. In the probabilistic approach, the focus was on the way to initialize the weights that can decrease the probability of converging toward local minima [24]. Although these approaches are fairly easy to implement and can also converge toward the global minimum, they unfortunately require a lot of running time.

Another interesting approach to learning the ANNs is to combine them with evolutionary algorithms (EAs). One of the strengths of the EAs over the two former ones is that they have enough resources to converge on the optimal solution. Metaheuristic

and evolutionary algorithms are known as stochastic approaches that are suitable to tackle complex, high dimensional, and NP-hard problems after spending a reasonable time. There are many swarm-based optimizers such as salp swarm algorithm (SSA) [15], Harris hawks optimization (HHO) [8], dragonfly algorithm (DA) [14], GWO [17], MFO [13], and whale optimizer (WOA) [16]. Neural networks using EAs have been studied for years, including genetic algorithms (GAs) [7], which are in fact the most widely used. The most popular way to combine EAs with ANNs is to use GAs to search globally in the space of solutions (weight configurations) and then improve this space in an iterative manner until the best solution is found (best configuration of the weights) [20]. Beside GA, other types of evolutionary methods have been used to learn ANNs [12].

In this chapter, we will focus more on the application of ANNs to autonomous robot navigation. Navigation is one of the most important elements in the design and development of any intelligent vehicle. Maintaining operation while avoiding collisions and staying safe is a priority in autonomous robots. Avoiding obstacles is the fundamental problem of the autonomous robot. Its purpose is to allow mobile robots to explore an unknown environment without colliding with objects. The most traditional approaches have been based on geometric models in which local cost maps have been constructed. These methods involve low-level intelligence, without any process of perception and/or sensing. Recently, navigation and robot trajectory planning have improved considerably with the help of ANNs, particularly MLP, reinforcement learning, and DNNs, because of their ability to automatically create the relevant features needed to control robots [21].

As discussed at the beginning of this section, evolutionary methods have also been used successfully to learn ANNs. Thanks to their ability to find global minimums, EAs are combined with ANNs to improve the development of smarter robots with greater navigation accuracy [3, 22, 27]. Our specific objective, in this chapter, is to empirically study other types of EAs and to demonstrate their effectiveness in autonomous robot navigation. More attention will be given to MFO.

MFO is a stochastic algorithm that inspires from transverse orientation of moths, a navigation method in nature. It has been proposed by Mirjalili [13]. Theorem of no free lunch (NFL) [25] states that there is no universal best optimizer. This theorem motivated us to train MLP using MFO algorithm and four other algorithms including GWO [17], cuckoo search (CS) [28], MVO [19], and PSO [10]. Besides, this algorithm is compared with two well-regarded gradient descent algorithm such as back-propagation and Levenberg–Marquardt (LM) to show how effective is MFO in training MLP in comparison with gradient descent methods. All the experiments were executed on a mobile robot navigation dataset. The evaluation metrics show the exploration and exploitation capabilities of MFO in comparison with all other methods used in this work.

The structure of this chapter is as follows: Sect. 2 zooms on the proposed MFO-based MLP trainer and the dataset used in this chapter. Section 3 describes the configuration of methods. Sections 4 and 5 are devoted to the results and discussions, respectively. Finally, the remarks and future research directions are concluded in Sect. 6.

2 Materials and Methods

In this section, the methods and dataset used in this research are described in detail.

2.1 Feedforward Neural Networks

Feedforward neural networks (FFNNs) are the most widely used of NNs that can perceive mathematical models using their fully connected layered structures. Among the neurons of FNNs, there are only one-directional and one-way connections arranging in various parallel layers [23]. FFNN networks contain three layers. Input layer is the first layer in FFNNs, and output layer is the last layer of FFNNs. Intermediate layers between input and output layer are called hidden layer. EA FFN consisting of the hidden layer is called MLP, which is illustrated in Fig. 1.

The connections among the layers in the MLP are represented as weights locating in the range of [−1, 1]. Two functions form the node structures of MLP networks including summation and activation. As presented in Eq. (1), the summation function is responsible in order to sum up for the product of each input, weight, and bias of the networks.

$$S_j = \sum_{i=1}^{n} \omega_{ij} I_i + \beta_j \tag{1}$$

where the total number of neuron inputs is represented by n , variable i as input is determined by I_i , β_j denotes to bias (threshold) weights of j-th hidden neuron, and the connection weights from the i-th node in the input layer to the j-th node in the hidden layer are determined by w_{ij}.

The output of Eq. (1) is an input to the activation function. There are several types of activation functions utilized for training MLPs. The most used activation function is the sigmoid function described in Eq. (2).

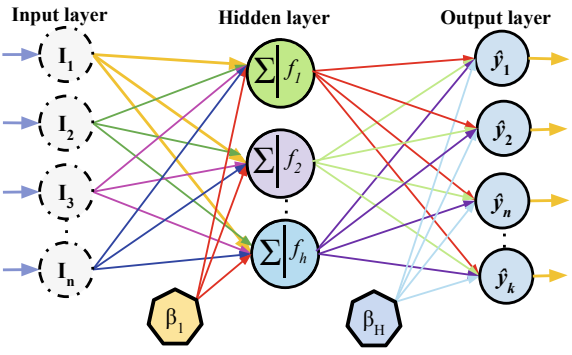


Fig. 1 Overall structure of MLP based on design in [7]

$$f_j(x) = \frac{1}{1 + e^{-S_j}} \tag{2}$$

Consequently, the final output of each neuron j can be described in Eq. (3) :

$$y_i = f_j(\sum_{i=1}^{n} \omega_{ij} I_i + \beta_j) \tag{3}$$

After designing the MLP structure, the procedure of training is accomplished for updating and tuning the set of biases and weights of the MLP network by meeting error criteria. The procedure of selecting appropriate sets of weight and bias between the inputs and outputs of MLPs is a challenging task and is considered as the definition of training MLPs. In the next, MFO is used as the trainer for the learning procedure of the MLPs.

2.2 MFO Algorithm

The MFO is a recent nature-inspired metaheuristic paradigm that attempts to imitate the navigation of moths in the night. MFO is a population-based algorithm, and there are several advanced variants of this method until now [26]. In this method, the moth's positions are managed using a matrix to setup the population of this algorithm as follows:

$$M = \begin{bmatrix} z_{1,1} & z_{1,2} & \cdots & z_{1,d} \\ z_{2,1} & z_{2,2} & \cdots & z_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ z_{n,1} & z_{n,2} & \cdots & z_{n,d} \end{bmatrix} \tag{4}$$

where the number of moths z is represented by index of n and d. Assume that for each moth of the problem, there is a sorted array according to the values of objective function represented as follows:

$$OM = \begin{bmatrix} om_1 \\ om_2 \\ om_3 \\ \vdots \\ om_n \end{bmatrix} \tag{5}$$

In which to the number of moths is denoted by n.

Another important component of this algorithm is the role of flames. The structure of the matrix of moths is very similar to the moths:

$$F = \begin{bmatrix} f_{1,1} & f_{1,2} & \cdots & f_{1,d} \\ f_{2,1} & f_{2,2} & \cdots & f_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ f_{n,1} & f_{n,2} & \cdots & f_{n,d} \end{bmatrix} \tag{6}$$

where the number of flames is represented by n index.

Moreover, we have an array for sorting the corresponding flames according to

$$OF = \begin{bmatrix} OF_1 \\ OF_2 \\ OF_3 \\ \vdots \\ OF_n \end{bmatrix} \tag{7}$$

where n denotes the flames's number.

MFO uses logarithmic spiral in order to update the position of moths with regard to flames as below:

$$S(M_i, F_j) = D_i e^{bz} \cos(2\pi z) + F_j \tag{8}$$

where M_i indicates the i-th moth, and F_j represents the j-th flame. In addition, the spiral function acting as the main component of MFO algorithm is represented by S and in order to define the shape of logarithmic spirals, b as a constant is considered. z is assumed as a random number in the range of [−1,1], and the distance between i-th moth and j-th flame is determined by D_i.

Generally, in MFO, both moths and flames are assumed as all feasible solutions in the search space. Moths act as the search agents and the best position of the moths is obtained using flames. Consequently, moths search around the position of flames and update them in order to obtain a better solution. This procedure makes keeping the best solutions.

The pseudo-code of MFO method is represented in the algorithm 1.

2.3 MFO for Training MLPs

In this section, we describe the proposed MFO-based MLP (MFO-MLP) trainer algorithm. In order to utilize the training procedure of MLP using MFO, two crucial aspects have to be addressed: how the individuals (moths) of MFO can be encoded in training of the MLP network and how the objective function (fitness function) can be formulated. In MFO-MLP, all individuals are encoded as the vectors of randomly real numbers within the [−1, 1] interval in order to represent a MLP candidate. Therefore, each vector in MFO consists of three parts. The format of solutions is shown in Fig. 2.

Algorithm 1 Pseudo-code of MFO approach

Input: iteration number (T_{max}) and Population size of the problem.
Output: The best obtained solution and related values of fitness
Initialize the population of agents $x_i\,(i = 1, 2, \ldots, n)$
Obtain the fitness rates.
while (maximum iterations) **do**
  Update flame number
  Define: OM = Fitness Function(M)
  if ($i == 1$) **then**
    F = sort(M), OF=sort(OM)
  else
    F = sort(M_{t-1}, M_t), OF = sort(M_{t-1}, M_t);
  end if
  for $i = 1 : n$ **do**
    for $j = 1 : d$ **do**
      Update r and t
      Obtain D
      Update $S(i, j)$
    end for
  end for
end while
Return the best agent

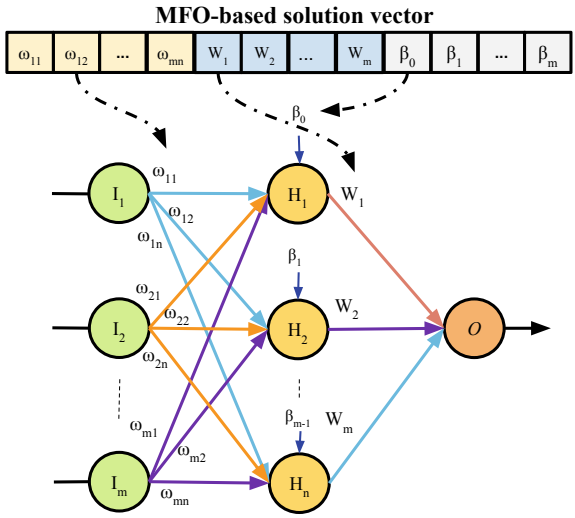


Fig. 2 Solution's structure in MFO-MLP based on work in [7]

The length of each vector is calculated as shown below:

$$Vector\_length = (n \times m) + (2 \times m) + 1 \tag{9}$$

where the number of input features in the dataset is indicated by n and in the hidden layer, m represents the number of neurons.

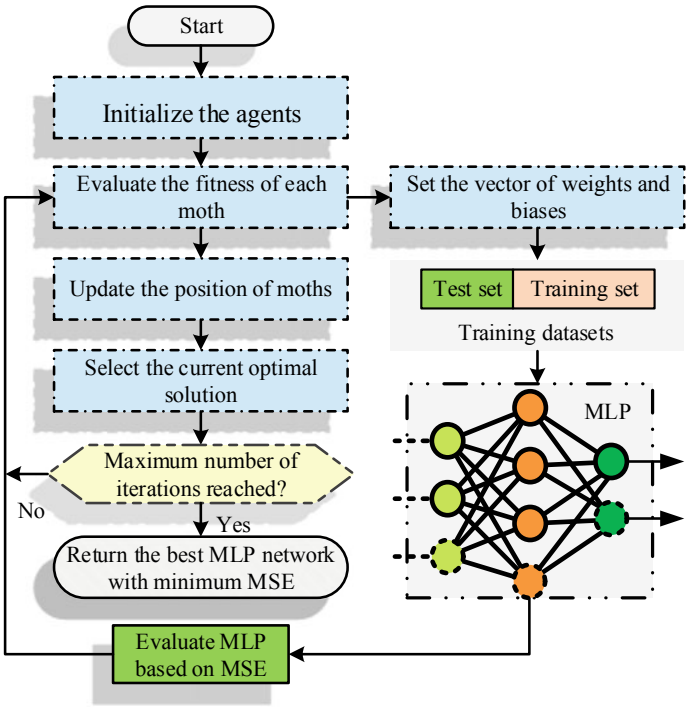


Fig. 3 Framework of MLP optimization using MFO algorithm

The second aspect that needs be addressed here is for selecting fitness function. In MFO algorithm, the vector of connection weights and biases is sent to the MLP network. Then, MLP evaluates those vectors using the training dataset. In this chapter, the mean squared error (MSE) is selected as the fitness function in order to calculate the fineness of the generated MLPs.

The MSE metric is obtained by

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{10}$$

where y_i and $\hat{y}_i$ indicate the actual and the predicted values, and the total number of instances in the training dataset is represented by n.

The overall steps of the MFO-based technique for the MLP neural network training are depicted in Fig. 3.

Table 1 Details of class features distribution of mobile robot navigation dataset

Name of class feature	No. of instances	Percentage (%)
Slight-right-turn	824	71
Slight-left-turn	330	29

2.4 Dataset

The dataset used in this study is for the experiments on the collection of multiple sensors extracted from a mobile robot navigating through a room [11]. The aim of this robot is for object's collision avoidance that surrounds it. This dataset is for the supervised learning tasks and contain 5426 instances with 24 features and four classes. These four classes in the dataset are move-forward, sharp-right-turn, slight-right-turn, and slight-left-turn. The algorithms used in this study are implemented for binary classification tasks. Therefore, we consider the slight-right-turn and slight-left-turn classes to reshape the dataset in a binary format. This procedure leads to a reduction in number of samples of the dataset into 1154 samples as represented in Table 1. More information about the used dataset in this research is represented in [4, 11].

3 Experimental Setup

This section compares the developed MFO-MLP trainer with other well-regarded EA-based MLP trainers and gradient-based training algorithms. Accuracy and area under the curve (AUC) are the evaluation metrics used in this work. We also used T-test.

All the experiments used in this work for implementing the MFO trainer and other algorithms are conducted using Python 3.7 on a 64-bit Windows operation system and an Intel 7 CPU processor at 1.9 GHz with a 16GB ram. It should be noted that in this work no commercial software for implementing the algorithms was utilized. The mobile robot dataset used in this study is partitioned into two components: 34% for testing and 66% for training using strategy of random sampling. Before training the EAs with MLPs, it is important to normalize the dataset for eliminating the influence of features having different scales. Therefore, all the features of the dataset are normalized using minmax normalization technique into the range of [0, 1]. This technique is given in the following rule:

$$W' = \frac{v_i - \min_J}{\max_J - \min_J}, \tag{11}$$

where W' represents the normalized value of W in the interval of $[\min_j, \max_j]$.

Table 2 Initial parameters of five EAs

Algorithm	Parameter	Value
MVO	Wormhole existence probability	[0.2, 1]
MFO	T	[−1, 1]
PSO	Acceleration constants Inertia weights	[2.1, 2.1]
CS	Discovery rate Pα	0.25
GWO	a	[2, 0]

In all experiments, each algorithm executes 30 independent runs, and in each run, there are 100 iterations. Besides, the population size is set to 200 for all EAs used in this study.

The controlling parameters for MFO, MVO, CS, PSO, and GWO are listed in Table 2. The initialization of these parameters for each EA is set as utilized and recommended in the specialized literature [5, 12, 18].

For determining how many neurons should be used in the hidden layer, researchers proposed different approaches in the literature. However, there are not standard principles that are agreed among the researchers about the superiority on which rule to be applied. In this study, it is followed by a common rule proposed by [5, 12, 18] where the rule set of the number of neurons in the hidden layer of MLP network is calculated by

$$E = 2 \times H + 1. \tag{12}$$

where H is the number of features in the dataset.

4 Experimental Results

MFO-based MLPs trainer is compared with different EAs and gradient-based trainers using three different classification metrics commonly popular in the machine learning field. These metrics are accuracy, AUC, and T-test.

4.1 Comparison with Other Well-Regarded Evolutionary Optimizers

This section describes the results of MFO-based MLP trainer in comparison with these EAs including CS, PSO, GWO, and MVO. As it can be seen from Table 3, the statistical results including average (AVG) of classification accuracy, standard

Table 3 Evaluation metrics of MLP training with different EAs for mobile robot navigation dataset

Algorithm	Metric	Accuracy	AUC
MFO	AVG	**0.944027**	**0.910519**
	STD	0.022473	**0.034488**
	Best	**0.954555**	**0.943893**
MVO	AVG	0.896862	0.824279
	STD	0.018695	0.039742
	Best	0.926209	0.880308
PSO	AVG	0.892112	0.814497
	STD	0.027771	0.048671
	Best	0.933842	0.884597
GWO	AVG	0.590843	0.583636
	STD	0.221569	0.191476
	Best	0.870229	0.907115
CS	AVG	0.881086	0.803507
	STD	**0.017859**	0.036756
	Best	0.910941	0.846466

deviation (STD) and the most accurate results of the each proposed EA show that MFO-based optimizer outperforms all other evolutionary trainers. Another important point that should be mentioned is that the obtained values by accuracy show that MFO has the lowest standard deviation and highest average giving this strong evidence that this algorithm can strongly prevent convergence into the local optima and obtain the optimal and best values for weights and biases of MLP networks. Moreover, MFO obtains the highest accuracy in comparison with other EAs showing that this EA has improvements in prediction of robot navigation.

AUC evaluation results for the proposed MFO and other EAs based on MLP networks are demonstrated in Table 3. According to the obtained results, the average, standard deviation, and best values reported for AUC denote that MFO-based trainer performs more robust than all the other EAs in the mobile robot dataset.

The boxplots for MFO as well as other EAs-based MLP models are represented in Figs. 4 and 5. These boxplots are created for reporting the accuracy and AUC rates of 30 independent runs for all EAs. The boxplots prove that MFO algorithm performs better than all other EAs for training MLP since this algorithm yields more compact boxes, and its median is higher than all other EAs.

To see whether the results of MFO-based trainer in terms of accuracy and AUC is significantly and statistically different from and greater than other evolutionary optimizers; the right one tailed T-test (greater than) experiment is implemented at 5% significance level. The obtained p-values by each EA-based trainer from T-test experiment are represented in Table 4. The results reveal that a significant difference exists between accuracy and AUC results obtained by MFO algorithm in comparison with other EAs. Besides, the average of accuracy and AUC values of MFO is always

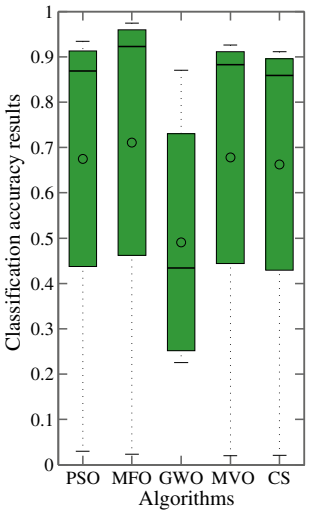


Fig. 4 Boxplot representation of the accuracy for different evolutionary optimizers

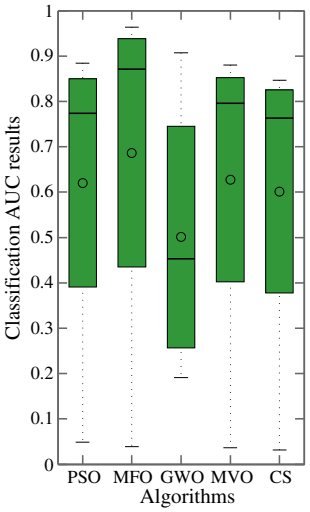


Fig. 5 Boxplot representation of the AUC for different evolutionary optimizers

Table 4 Obtained *p*-values of the T-test for MFO and other EA-based MLP trainers

Algorithm/Metric	PSO	GWO	MVO	CS
Accuracy	2.29E-06	7.09E-06	7.04E-06	1.63E-07
AUC	1.01E-06	2.36E-06	7.59E-06	1.21E-07

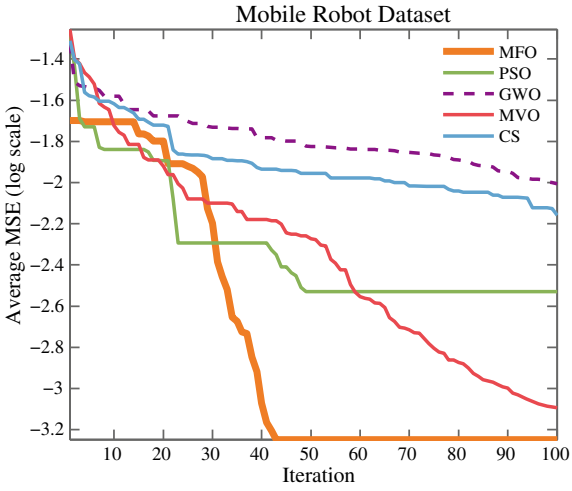


Fig. 6 MSE average convergence curves for different evolutionary optimizers

greater than other EAs in the experiment. These findings again justify that obtained results are not achieved by chance and show the robustness of MFO for the training of MLP neural networks.

The average convergence curves for all EAs over 100 iterations are demonstrated in Fig. 6. This figure shows that MFO has the fastest convergence speed for finding the global optimum.

In summary, the experimental results obtained in this section imply that MFO-based trainer has a better efficiency in comparison with other EAs employed.

4.2 Comparison with Gradient-Based Algorithms

This section verifies the performance of MFO in comparison with two well-regarded gradient-based techniques including back-propagation (BP) and Levenberg–Marquardt (LM) based on classification accuracy, AUC, and T-test evaluation metrics. In fact, this comparison is important because it reveals that this study does not only consider the comparison of proposed MFO-MLP-based trainer with other well-known evolutionary optimization algorithms, which are from its own type, but also makes a comparison with gradient descent algorithms which are able to challenge the performance of MFO.

The results of the average, best value, and standard deviation of accuracy and AUC of gradient decent algorithms for 30 different runs are presented in Table 5.

It can be seen that MFO obtains the highest average value of accuracy and AUC in comparison with BP and LM. Moreover, the lowest standard deviation values for accuracy and AUC are denoted to the MFO-MLP. It is worth mentioning that LM and BP obtain the higher standard deviation values in comparison with other EAs (PSO,

Table 5 Evaluation metrics of MFO-based MLP trainer with BP and LM for mobile robot dataset

Algorithm	Metric	Accuracy	AUC
MFO	AVG	**0.944027**	**0.910519**
	STD	**0.022473**	**0.034488**
	Best	**0.954555**	**0.943893**
BP	AVG	0.754194	0.76631
	STD	0.10873	0.117588
	Best	0.916274	0.929898
LM	AVG	0.802111	0.775844
	STD	0.100069	0.124485
	Best	0.894844	0.903664

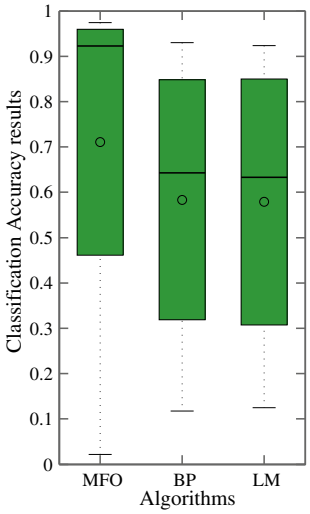


Fig. 7 Boxplot representation of the accuracy for MFO, BP and LM

CS, and MVO) except GWO that has been analyzed in the previous subsection. The boxplots for the results of accuracy and AUC are illustrated in Figs. 7 and 8, respectively. These plots clearly demonstrate that the MFO-MLP obtains the highest average of accuracy and AUC as well the most compact boxes. These results support the claim that MFO optimizer performs more robust and stable in training of MLP networks. It can be noted that the widest range of boxplots in BP and LM shows that these approaches have a tendency to be trapped into the local optima. Finally, in Table 6, two T-tests greater than experiments are conducted for accuracy and AUC evaluation metrics to statistically confirm that whether there are considerable differences among MFO and two other gradient descent algorithms. The p-values indicate that MFO has a significant difference with both BP and LM substantially, and its average is always greater than both of BP and LM.

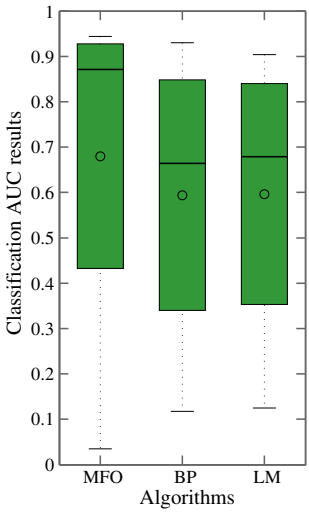


Fig. 8 Boxplot representation of the AUC for MFO, BP and LM

Table 6 Obtained *p*-values of the T-test for MFO and both gradient descent algorithms

Algorithm/Metric	BP	LM
Accuracy	2.47E-05	4.10E-05
AUC	7.04E-05	1.83E-04

5 Discussion of Results

In summary, the experimental results confirmed that the MFO-based trainer is not only able to outperform the evolutionary optimizers but also the gradient-based algorithms in terms of both avoidance of trapping into the local optima and convergence speed. The local optima stagnation in this algorithm is higher than other EA methods applied in this paper since moths assign a flame over the course of iteration, and this procedure leads to high exploration of the search space. This characteristic is the main reason for the better performance of MFO in comparison with other EAs.

Another interesting and important finding should be mentioned here is the capability of higher convergence speed of MFO in training MLP networks. This characteristic originates from the consideration of the promising solutions by the flames acting as the guides for each moth. This is one of the reasons why the results of MFO are superior and robust with other utilized EAs.

Comparing the capability of training MLPs using MFO with two of the most well-regarded methods from the family of gradient-based algorithms including BP and LM presents substantial findings. The results show that training MLP suing BP and LM allows the solutions to bend downward to the deepest steep. This mechanism is the reason why both algorithms suffer from getting stuck into the local optima. As MFO is a stochastic population-based optimizer and structure of this evolutionary

optimization leads to unexpected changes in the weights and biases of MLP network, it benefits from both local optima avoidance and higher exploration for the search space of the problem. Also, in most of the EAs, their performance is higher than both gradient descents used in this work. This is the main reason why we developed evolutionary algorithms for this robot navigation.

In overall, MFO outperformed all the other methods used in this book chapter, showing that its robustness in training of the problem of navigation for a mobile robot as a powerful black-box tool.

6 Conclusion and Future Directions

In this chapter, an attempt is made to employ of the recently proposed MFO optimizer in order to train MLP networks based on an autonomous navigation robot dataset. Since the problem of navigation in robots is a challenging task, MFO is utilized as a robust optimization algorithm to find the optimal weights and biases for obtaining more accurate navigation results with minimum error criteria (MSE). To show the higher capability of MFO in obtaining the most accurate and robust results, a series of experiments are performed using the well-regarded evaluation metrics: accuracy, AUC, and T-test. MFO is then compared with different bio-inspired evolutionary algorithms and gradient descent approaches. The findings report that MFO is strongly handling the problem of local optima avoidance alongside a fair-minded convergence speed when MLP network is trained. Besides, this algorithm shows its stability in providing the highest accurate results compared to other approaches used in this study.

Future works can investigate the framework used in this study for other mobile robot navigation datasets. Another suggestion can be using the employed EAs in this work and compare them with other types of neural networks such as radial basis function networks for similar datasets.

References

1. Cao W, Wang X, Ming Z, Gao J (2018) A review on neural networks with random weights. Neurocomputing 275:278–287
2. da Silva IN, Hernane Spatti D, Andrade Flauzino R, Liboni LHB, dos Reis Alves SF (2017) Artificial neural networks. Springer, Heidelberg
3. da Silva Assis L, da Silva Soares A, José Coelho C, Van Baalen J (2016) An evolutionary algorithm for autonomous robot navigation. Proc Comput Sci 80:2261–2265
4. Dash T, Nayak T, Ranjan Swain R (2015) Controlling wall following robot navigation based on gravitational search and feed forward neural network. In: Proceedings of the 2nd international conference on perception and machine intelligence. ACM, pp 196–200
5. Faris H, Aljarah I, Mirjalili S (2016) Training feedforward neural networks using multi-verse optimizer for binary classification problems. Appl Intell 45(2):322–332
6. Haykin S (1994) Neural networks, vol 2. Prentice Hall, New York

7. Heidari AA, Faris H, Aljarah I, Mirjalili S (2018) An efficient hybrid multilayer perceptron neural network with grasshopper optimization. Soft Comput 1–18
8. Heidari AA, Mirjalili S, Faris H, Aljarah I, Mafarja M, Chen H (2019) Harris hawks optimization: algorithm and applications. Future Gener Comput Syst 97:849–872
9. Jordanov IN, Rafik TA (2004) Local minima free neural network learning. In: 2004 2nd international IEEE conference on 'Intelligent Systems'. Proceedings (IEEE Cat. No. 04EX791), vol 1. IEEE, pp 34–39
10. Kennedy J (2010) Particle swarm optimization. Encyclopedia of machine learning, pp 760–766
11. Madi S, Baba-Ali R (2018) Classification techniques for wall-following robot navigation: a comparative study. In: International conference on advanced intelligent systems and informatics. Springer, Heidelberg, pp 98–107
12. Mirjalili S (2015) How effective is the grey wolf optimizer in training multi-layer perceptrons. Appl Intell 43(1):150–161
13. Mirjalili S (2015) Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm. Knowledge-Based Syst 89:228–249
14. Mirjalili S (2016) Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. Neural Comput Appl 27(4):1053–1073
15. Mirjalili S, Gandomi AH, Mirjalili SZ, Saremi S, Faris H, Mirjalili SM (2017) Salp swarm algorithm: a bio-inspired optimizer for engineering design problems. Adv Eng Softw 114:163–191
16. Mirjalili S, Lewis A (2016) The whale optimization algorithm. Adv Eng Softw 95:51–67
17. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61
18. Mirjalili S, Mirjalili SM, Lewis A (2014) Let a biogeography-based optimizer train your multi-layer perceptron. Inf Sci 269:188–209
19. Mirjalili S, Saremi S, Mirjalili SM, Coelho LS (2016) Multi-objective grey wolf optimizer: a novel algorithm for multi-criterion optimization. Expert Syst Appl 47:106–119
20. Reeves C, Rowe JE (2002) Genetic algorithms: principles and perspectives: a guide to GA theory, vol 20. Springer Science & Business Media
21. Shabbir J, Anwer T (2018) A survey of deep learning techniques for mobile robot applications. arXiv preprint arXiv:1803.07608
22. Song B, Wang Z, Zou L (2017) On global smooth path planning for mobile robots using a novel multimodal delayed pso algorithm. Cogn Comput 9(1):5–17
23. Souza F, Matias T, Araújo R (2011) Co-evolutionary genetic multilayer perceptron for feature selection and model design. In: 2011 IEEE 16th conference on Emerging Technologies & Factory Automation (ETFA), IEEE, pp 1–7
24. Wessels LFA, Barnard E (1992) Avoiding false local minima by proper initialization of connections. IEEE Trans Neural Netw 3(6):899–905
25. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82
26. Xu Y, Chen H, Heidari AA, Luo J, Zhang Q, Zhao X, Li C (2019) An efficient chaotic mutative moth-flame-inspired optimizer for global optimization tasks. Expert Syst Appl 129:135 – 155
27. Yamany W, Fawzy M, Tharwat A, Hassanien AE (2015) Moth-flame optimization for training multi-layer perceptrons. In: 2015 11th International Computer Engineering Conference (ICENCO), IEEE, pp 267–272
28. Yang X-S, Deb S (2009) Cuckoo search via lévy flights. In: 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC), IEEE, pp 210–214

Link Prediction Using Evolutionary Neural Network Models

Rawan I. Yaghi, Hossam Faris, Ibrahim Aljarah, Ala' M. Al-Zoubi, Ali Asghar Heidari and Seyedali Mirjalili

Abstract Link prediction aims to represent the dynamic networks' relationships of the real world in a model for predicting future links or relationships. This model can help in understanding the evolution of interactions and relationships between network members. Many applications use link prediction such as recommendation systems. Most of the existing link prediction algorithms are based on similarity measures, such as common neighbors and the Adamic/Adar index. The main disadvantage of these algorithms is the low accuracy of results since they depend on the application domain. Moreover, the datasets of link prediction have two significant problems: the imbalanced class distribution and the large size of the data. In this chapter, evolutionary neural network-based models are developed to solve this problem. Three optimizers are used for training feedforward neural network models including genetic

R. I. Yaghi · H. Faris · I. Aljarah · A. M. Al-Zoubi
King Abdullah II School for Information Technology,
The University of Jordan, Amman, Jordan
e-mail: rawanyaghi1993@gmail.com

H. Faris
e-mail: hossam.faris@ju.edu.jo

I. Aljarah
e-mail: i.aljarah@ju.edu.jo

A. M. Al-Zoubi
e-mail: alaah14@gmail.com

A. A. Heidari
School of Surveying and Geospatial Engineering, College of Engineering,
University of Tehran, Tehran, Iran
e-mail: as_heidari@ut.ac.ir; aliasgha@comp.nus.edu.sg; t0917038@u.nus.edu

A. A. Heidari
Department of Computer Science, School of Computing,
National University of Singapore, Singapore, Singapore

S. Mirjalili (✉)
Torrens University Australia,
Fortitude Valley, Brisbane, QLD 4006, Australia
e-mail: ali.mirjalili@gmail.com

Griffith University, Brisbane, QLD 4111, Australia

S. Mirjalili et al. (eds.), *Evolutionary Machine Learning Techniques*,
Algorithms for Intelligent Systems, https://doi.org/10.1007/978-981-32-9990-0_6

algorithm, particle swarm optimization, and moth search. For this purpose, the link prediction problem is formulated as a classification problem to improve the accuracy of the results by constructing features of the traditional link prediction methods and centrality measures in any given link prediction dataset. Also, this work tries to address two problems of the data in two ways: externally using sampling techniques (random and undersampling) and internally using the geometric mean as a fitness function in the proposed algorithms. The results reveal that the proposed model is superior in terms of the sensitivity and geometric mean measures compared to the traditional classifiers and traditional link prediction algorithms.

Keywords Algorithm · Optimization · Neural networks · Artificial intelligence · Machine learning · Data science

1 Introduction

Social communities and networks have recently attracted researchers' attention to understand social organizational structures by simplifying representations of real-world relationships [3, 4, 34]. The structure of these networks and communities consists of nodes and links, where nodes represent network entities, such as people, and are joined together by links which represent the correlations and associations between nodes. The properties of networks can be studied using concepts from network science, the aim of which is to understand the structure and evolution of complex network systems. One of the most important properties of these networks is its dynamic change over time. For example, in the Facebook social network, whenever a new user adds another as a friend, the network structure changes, and new link is added to the network to establish the new friendship; conversely, when a user removes a friend, a link is removed from the network [40]. Change over time is the key concept of dynamic networks, but leads to several questions:

- How do correlations and associations between nodes change over time in a network?
- How are the correlations and associations between two nodes influenced by other nodes in the same network?
- What features are responsible for the changing correlations and associations between nodes in a network?

These questions can be resolved by predicting the likelihood of future associations between two nodes in a network, which is called the link prediction problem. Link prediction is a major task in network analysis, especially in social networks. Moreover, due to the increasing use of networks, link prediction can help improve network performance by predicting the associations and correlation between users in a satisfying way. Many applications have utilized link prediction, such as social media [20], e-commerce [11], physics [13, 38], and citation networks [42].

Predicting the links between network nodes or social communities helps in understanding network associations, and the significant role they play in network science. Moreover, link prediction is used to know which interactions and correlations between entities will soon be established or removed. For example, in information retrieval, it can be used to predict links between words and documents within bipartite networks which represent word occurrence [12].

Traditional link prediction methods, such as common neighbors (CN) [33], the Jaccard index [25], and the Adamic/Adar index [1], are similarity-based methods. Each method has its advantages and disadvantages. For example, one major disadvantage of CN is that it assigns high similarity values to two nodes if they have many shared neighbors [33]. The Adamic/Adar index addresses this shortcoming by giving higher weight to less-connected neighbors [1]. The Adamic/Adar index's emphasis on paired nodes with rarely shared neighbors is considered an advantage when used in networks such as personal home pages. However, it produces poor results in co-authorship networks, which is one of its disadvantages [17]. The preferential attachment has two disadvantages: first, the similarity between two nodes depends only on their connectivity, which gives high similarity values to highly connected nodes. Second, maximization of network connectivity occurs when creating many links between all pairs of nodes with neighbors [8, 19]. Each of these methods excels in specific domains and gets poor results in others. For example, the Jaccard index performs well in IR applications and obtains lower-accuracy results in other cases [12].

In this work, links are predicted by constructing features with traditional link prediction methods and using centrality measures in each link prediction dataset to serve as input to a classification model. Then, the prepared datasets are classified using evolutionary multilayer perceptron (MLP) networks, which are optimized by algorithms such as moth search (MS), genetic algorithms (GAs), and particle swarm optimization (PSO).

The objective of this work is to overcome the weaknesses of traditional link prediction methods. In more details, this work formulates link prediction as a classification problem by constructing features using traditional link prediction methods and calculating centrality measures in the dataset, where each method represents a feature and each row represents a possible edge, with features and class label in an undirected graph of a given dataset. On the other hand, link prediction datasets are often large and imbalanced; we apply random sampling and undersampling techniques to the prepared datasets to solve these problems. Then, evolutionary MLP models are developed based on these processed datasets by utilizing three nature-inspired algorithms for training the neural network models which are: moth search (MS), genetic algorithms (GAs), and particle swarm optimization (PSO). The performance of the developed evolutionary-based models is compared to common classifiers which include decision trees (DT), support vector machine (SVM), multilayer perceptron (MLP), and naïve Bayes (NB) algorithms.

The rest of this chapter is organized as follows: Sect. 2 represents the previous works of the Link prediction. Section 3 describes the background of the utilized methods, while Sect. 4 introduced the approach of this work. Section 5 explains the experiments and obtained results and the conclusion presented in Sect. 6.

2 Literature Review

Link prediction is a popular data mining task in various application domains and has attracted attention in recent years. This section discusses briefly the recent progress about link prediction methods.

Link prediction methods can be divided into two major types: supervised and unsupervised methods. The authors in [43] proposed a new collective classification approach for predicting the existence and the type of links between entities in relational domains, they applied the relational Markov network framework to define a model of joint probabilistic of links in the graph and entity attributes. The work in [2] proposed a supervised link prediction method by identifying a set of features, which are topological features such as the shortest distance between the two nodes: semantic features such as the number of matching keywords and aggregated features such as the summation of common neighbors. They used the well-known classification algorithms (k-NN, multilayer perceptron, RBF network, decision tree, and SVM), and they found out that SVM slightly outperformed others classification algorithms using different evaluation measures (accuracy, precision–recall, F-values, squared error).

In addition, [47] used supervised link prediction methods in a metabolic network based on the integration between genomic and chemical data. The metabolic network is a type of protein network that consists of enzymes and chemical compounds. It focused on enzyme relationships, where enzymes represent nodes and enzyme to enzyme relations represent edges. In contrast, [29] proposed a novel unsupervised learning algorithm to predict links by using aggregative statistics. Aggregative statistics are used in their proposed algorithm to learn the parameters of unseen-type links and then predict the links based on the learned parameters.

Researchers in [31] used supervised or unsupervised methods and mentioned the advantages of both. They presented a supervised framework for link prediction, taking into consideration the following issues: an observational period of the network, existing methods generality, imbalance degrees, variance reduction, and sampling approaches. In addition, considering the previous issues, they proposed a flow-based predicting algorithm, and the results outperformed unsupervised link prediction by using the area under curve (AUC) as a performance evaluation by more than 30%.

Other researchers proposed different models for link prediction.

Liben-Nowell and Kleinberg [30] provided several mathematical methods for link prediction based on proximity measures for large networks of scientific collaboration (co-authorship networks). The mathematical methods were divided into three categories: The first one is based on node neighborhood, the second is based on the ensemble of all paths, and the third is “higher-level” approaches, which combine between the two previous methods. In addition, Murata and Moriyasu [35] proposed methods for link prediction based on weighted proximity measures. These methods consist of both the weights of the existing links in a social network and proximity measures of the graph. Moreover, Wang et al. [44] proposed a model based on machine learning called novel local probabilistic graphical model. The model was

used for estimating the joint co-occurrence probability of two nodes by scaling the large graphs. On the other hand, Liben-Nowell and Kleinberg [30] proposed the features using methods based on the contents of interactions or based on attributes of interactions. In addition, [41] used semantic similarity methods as features for the link prediction problem since these methods give accurate predictors of friendship links. Moreover, Chen and Chen [10] used ant colony optimization to provide a new method of link prediction.

3 Preliminaries

This section gives a brief background of the main concepts used in this work: link prediction, standard classification algorithms, and metaheuristic algorithms.

3.1 Multilayer Perceptron Networks (MLP)

Artificial neural networks (ANNs) are a kind of logical models developed based on biological neural systems which are used to approximate functions from many inputs. ANNs can be presented as a collection of interconnected processing elements, called "neurons," which exchange numeric weights. A common example of an ANN is the MLP. In MLP, neurons are distributed over multiple directed layers, like nodes in a directed graph. MLPs have proven several benefits including parallelism and generalization powers [7].

Figure 1 illustrates a simple MLP with only one hidden layer. In MLP, we should use a summation function as:

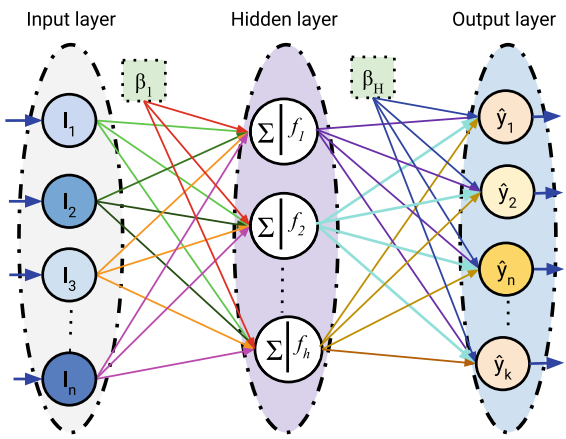


Fig. 1 Simple artificial neural network architecture

$$S_j = \sum_{i=1}^{n} w_{ij} I_i + \beta_j \tag{1}$$

where I_i is the input value of neurons in the input layer, w_{ij} is the connection weight connecting I_i to neuron j, β_j is a bias weight and n is the number of inputs. Results of this function will be inserted to the activation function. Usually, the activation function is nonlinear. Example of such a function is sigmoid, which is calculated as in Eq. 2.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2}$$

Hence, we have:

$$y_i = f_j \left(\sum_{i=1}^{n} w_{ij} I_i + \beta_j \right) \tag{3}$$

y_i is output of neuron j. The input of the activation function is the output of the summation function, whereas the output of the activation function is considered as the output of neuron j, which is also the weight that connects neuron j with the neurons in the next layer. When an ANN model is created, a training algorithm is applied to optimize its parameters (set of weights). These weights are optimized to approximate the results.

One of the principal factors affecting the results and efficacy of ANNs is the learning method utilized for training of the network. The goal of the training algorithm is to find the best set of weights to connect the processing elements of the network and to minimize some error criterion. We can use both gradient-based and stochastic methods for this purpose. The backpropagation algorithm is classified as a gradient-based technique [15, 22]. Backpropagation has two steps: the forward pass, which is meant to predict the output class label from the input data, and the backward pass, which propagates back the cost function.

In contrast, stochastic search methods rely on randomness. The advantage of these search methods is their ability to search for the global optimum rather than a local one [18]. However, their main drawback is the computation time.

3.2 Evolutionary and Swarm-Based Optimizers

The majority of evolutionary algorithms come from the behavior of creatures in natural or biological systems like GAs, PSO, and their variations. These algorithms are becoming dominant methods in solving various kinds of complex optimization problems [18, 24, 39]. The following subsections describe the algorithms used in the proposed models.

3.2.1 Genetic Algorithm (GA)

GA is an evolutionary algorithm inspired by Darwin's theory of natural selection and genetics in biological systems. GAs are used to discover the optimal solution from candidate solutions. The set of solutions is called a population and each solution is called a chromosome. GA discovers the optimal solution after a chain of iterative computations based on principles of evolution. The key computation phases in GAs are selection, crossover, and mutation. In selection, the individuals with the best fitness values are selected for use in GA operations. Crossover obtains new solutions by randomly exchanging genes between two selected chromosomes. There are three common types of crossover: one-point, two-point, and homologous. In mutation, a gene in one chromosome is randomly changed from 0 to 1 or vice versa. Through offspring, the old population is replaced using elitism or diversity strategies.

In the evolution step, the GA uses a fitness function to evaluate the quality of each candidate solution after passing the computations in each iteration of the algorithm until it reaches the end of iterations or meets a termination condition [14, 21].

3.2.2 Particle Swarm Optimization (PSO)

PSO is a metaheuristic inspired by bird flocking or fish schooling social behavior when trying to find an optimal food source. PSO was first introduced by Eberhart and Kennedy [16]. The current location of each bird is the best location of food that the bird finds. Additionally, the best location of food that any bird of the flock finds is controlled by the bird's movement. PSO is used to find optimal solutions using a group of particles. The group of particles is called a swarm and represents the population [6, 26, 27]. Each particle evaluates its location based on a fitness function and updates its speed (movement or velocity) by tracking two values in each iteration: the personal best (the optimal position that the particle has found) and the global best (the optimal position achieved by all particles). There are also some constants and particles inertia that affect the particle movement. However, the movement of all particles (the whole swarm) is directed toward the optimal global solution, until the maximum number of iterations is reached.

PSO uses the following equation to update particles's locations:

$$L_i(t+1) = L_i(n) + S_i(n+1) \tag{4}$$

where L_i is the location of particle i, n is the iteration number and S_i is the speed of particle i. Each particle's speed calculated by the following equation:

$$S_i(n+1) = W \cdot S_i(n) + c_2 \cdot r_2 \cdot [LP_i - L_i(n)] + c_2 \cdot r_2 \cdot [LG - L_i(n)] \tag{5}$$

LP_i is the current best location of particle i. LG is the current best global location of the whole swarm, W is inertia weight, r_1 and r_2 are random numbers between 0, and 1 and c_1 and c_2 are constant factors.

3.2.3 Moth Search Algorithm (MS)

MS is a recent metaheuristic algorithm which is inspired by phototaxis of moths [45].[1] MS models the phototaxic and flight behaviors of moths in nature. Moths instinctively tend to adjust their flight direction as much as possible to move toward a light source. Conversely, the function of Levy flight can be expressed by:

$$L(s) \sim |S|^{-\beta} \tag{6}$$

where $L(s)$ is the space drawn from the Levy distribution, S is the space, and β is an index in the range [1, 3].

The MS algorithm optimizes a known function using the exploration and exploitation operations based on the phototaxis of moths. In this method, Levy flights are used to update the moth's positions; for moth i, position can be updated as shown in the following equation.

$$x_i^{t+1} = x_i^t + \alpha L(s) \tag{7}$$

where x_i^{t+1} and x_i^t are represent the new and base location at generation t, and $t + 1$ is the current generation and parameter α is the scale factor.

To fly straight, some moths that are distant from a light source which will fly in a line toward it, a process that can be described as:

$$x_i^{t+1} = \lambda \times \left(x_i^t + \phi \times \left(x_{best}^t - x_i^t\right)\right) \tag{8}$$

where x_{best}^t represents the best moth at generation t, ϕ represents an acceleration factor, and λ is a scale factor. In contrast, the solution may move toward a destination target that is beyond the best agent; this case can be described as:

$$x_i^{t+1} = \lambda \times \left(x_i^t + \frac{1}{\phi} \times \left(x_{best}^t - x_i^t\right)\right) \tag{9}$$

[1] MATLAB code of MS is available at: https://www.mathworks.com/matlabcentral/fileexchange/59010-moth-search-ms-algorithm.

4 Methodology

This section describes the research methodology processes followed in this work. The four major processes in the research methodology are data collection, data preprocessing, model development, and model evaluation. In the following, each of the processes is described in details.

4.1 Data Collection and Preparation

The data under investigation in this work represent undirected graphs. The data were collected at two different time-stamps: t and $t + 1$, where the graph obtained at time t is used for training the classification model and the graph obtained at time $t + 1$ is for testing the trained model. Five benchmark datasets are selected from different networks' domains. Table 1 represents the number of nodes ($|V|$), edges ($|E|$), and the domain of each dataset. Moreover, the table shows the ratio between nodes and edges.

Each dataset is preprocessed using four sub-processes methods which are formatting the datasets, features extraction, normalization, and sampling. These subprocesses are described as follows.

Dataset's preparation: This phase aims at preparing the datasets in an appropriate format to be applied for the link prediction problem, which will be the matrix format (adjacency matrix). The rows and columns in the matrix represent the name of nodes. The matrix entries represent binary digits 1 or 0 based on the existence of the link between two nodes in specific row and column, where 1 represents the existence of a link between two nodes, and 0 represents the absence of the link. Figure 2 is a simple example of an adjacency matrix format and its corresponding graph.

Extracting the features: This phase aims at preparing the link prediction benchmark datasets by extracting features using similarity and centrality measures. The

Table 1 Dataset's description

Dataset name	$\|V\|$	$\|E\|$	Ratio (%)	Domain
BUP	105	441	24	Political blogs network
CEG	297	2148	14	Biological network
UAL	332	2126	16	Airport traffic network
INF	410	2765	15	Face-to-face contacts network
SMG	1024	4916	21	Co-authorship network

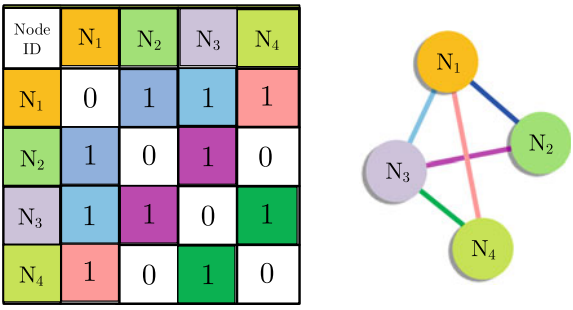


Fig. 2 Adjacency matrix and its corresponding graph

Table 2 Link prediction features

Feature number	Feature name
F1	Common neighbors
F2	Adamic/Adar
F3	Jaccard
F4	Salton
F5	Preferential attachment
F6	Leicht-Holme-Newman
F7	Hub depressed
F8	Hub promoted
F9	Sørensen index
F10	Degree centrality
F11	Closeness centrality

extraction is performed on any possible edge using the similarities and centrality measures that are mentioned in Table 2.

In this work, eleven features will be extracted, which are 9 similarity measures and other 2 centrality measures. The reasons behind using these similarity measures are because each of them is doing well in different domains, and overcoming the shortages of using each one alone. Moreover, degree centrality and closeness centrality measures are used to enhance the importance of a given edge that will be predicted by quantifying the importance of two nodes of a given edge. The probability of predicting the important links will be increased by quantifying the edge centrality measures.

Centrality measures represent the influence and the importance of a node in a network that depends on the relations between nodes. Centrality measures of nodes can be used to measure the node's significance and to determine which nodes are distinguished. The most popular centrality measures are degree, closeness, and betweenness [32]. Degree centrality and closeness centrality are used in this work.

Node degree centrality (DC): node degree refers to the number of relations or direct connections (links) with other nodes which is equal to the neighbors of the node in an undirected graph. If the number of the links of node increases, the significance

of node will be increased. It is a very effective measure because it can define the importance for each node alone; hence if two nodes have many connected links with other nodes in the network, they are more likely to be connected by link [32, 46]. DC is defined in Eq. 10.

$$DC = \frac{deg(x_1) + deg(x_2)}{2} \tag{10}$$

where $deg(x_1)$ denote the degree of node x_1.

Node closeness centrality (CC) measures the distances between a node and all other nodes in the network. If a node can access most of the other nodes through less distance pathway, the significance of this node will be increased. Some nodes are being connected by a large number of nodes while they are disconnected from the whole network. In this situation, degree measure is not enough because it does not take into consideration the links with all other nodes which can give an important indication of the node significance. Therefore, this centrality measure can overcome this situation [32, 46]. It is defined in Eq. 11, where d(x_1, y) denotes the shortest distance between node x_1 and all other nodes y.

$$CC = \frac{(\frac{N-1}{\sum_y d(x_1,y)}) + \frac{N-1}{\sum_y d(x_2,y)}}{2} \tag{11}$$

After the measures are applied on the datasets and the features are extracted, each column in each dataset represents a link prediction feature except the last column, which represents the class label for each edge. If it exists, the class label equals 1, if not, the class labels equal 0. The rows represent all possible edges with their features and their class labels.

Normalizing the datasets: After extracting the features, the datasets are normalized. Normalization is one of the most important preprocessing techniques. It helps in scaling the dataset attributes to fit into a specific and unified range. There are different kinds of normalization such as decimal scaling and Min–Max normalization [36]. In this work, the Min–Max normalization is used to scale the data to [0, 1]. The following equation represents the Min–Max normalization which transfers a value x to z which fits in the new range.

$$z = \frac{x - \min_A}{\max_A - \min_A} \times (new\_max_A - new\_min_A) + new\_min_A \tag{12}$$

where the max_A and min_A refers to the initial range and new_max_A and new_min_A refers to the new range.

Sampling the datasets: In general, link prediction datasets are imbalanced datasets, where the number of instances in nonexistent links' class is more than the number of instances in-existent links' class. This work deals with the distribution of the imbal-

anced dataset of link prediction to avoid the poor performance based on two different levels. The first one is by using sampling techniques: random and undersampling.

Sampling techniques are also used to handle the size problem of link prediction dataset since the large graphs consume more time and computation. Sampling techniques extract a sample subgraph with fewer nodes and edges than the full-graph. Thus, the performance in terms of running time will be improved. These sampling techniques are used in the model development and compared to the main approach, which is the full-graph approach. Thus, the model development in is performed using three approaches, which are training based the full-graph, training based on a random graph, and training based on an undersampling approach.

- **The full-graph**: This approach uses full-graph in training time. Since the full-graph approach does not solve the problems of link prediction datasets which are, as mentioned previously: the imbalanced dataset and the size problem, this approach is applied to benchmark the other approaches and to quantify their possible improvements.
- **Random graph**: This approach selects a random graph from the whole training graph. In this work, the number of nodes in the training graph is 100 nodes. The chosen random graph is the one that has the largest number of links among all possible random graphs.
- **Undersampling**: Undersampling is a technique used in data analysis to adjust the distribution of classes in imbalanced datasets [9]. The random undersampling is used only for the class of nonexistent links (class 0) in training datasets, where the number of instances in this class is more than the instances in the class of existent links (class 1). Five versions of each training dataset are created based on five undersampling ratios. The ratio of each version will be the number of instances in the class of existent links to the number of instances in the class of nonexistent links (ones: zeros). In the first version, the number of instances in the class of nonexistent links is equal to the instances in the class of existent links, thus the percentage in this version is 1:1. In the second version, the numbers of instances in class of nonexistent links are 2 multiplied by the instances in the class of existent links; thus, the percentage in this version is 1:2 and so on for the rest of ratios.

4.2 Model Development

In this process, hybrid models for link prediction datasets are proposed. The proposed models are MLP neural networks optimized by different nature-inspired algorithms: GA, PSO, and MS.

Nature-inspired algorithms are used to tune the MLP parameters (set of weights). In training MLP, there are two important points that should be mentioned and discussed; the first one is the encoding scheme of the candidate solutions, and the second one is the fitness function.

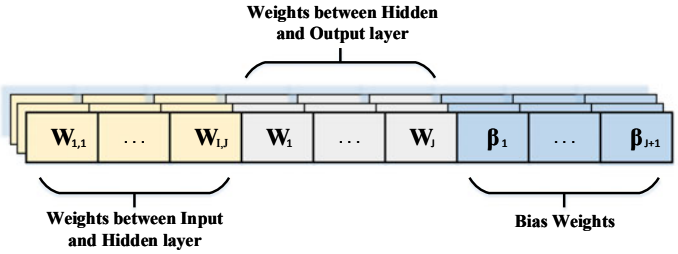


Fig. 3 Individual structure

The individuals are encoded as shown in Fig. 3, since a one hidden layer has good generalization performance for most of the problems, only one hidden layer is used to train the MLP. In this work, each individual should have the connection weights between the input and the hidden layer, the connection weights between the hidden layer and the output layer, and the bias weights. In each iteration, individuals are assigned to an MLP and then the MLP is evaluated based on a training dataset. This evaluation quantifies the fitness value of the network.

G-mean measure is used as a fitness function because it provides a balance of accuracy of the two classes at the same time. G-mean has a property of the distribution of instances between classes in an independent way. This property gives the G-mean the robustness in circumstantial situations where the distribution between classes changes over time or between the training and testing datasets [28]. G-mean can be calculated for each individual from the confusion matrix. G-mean is defined as:

$$G\text{-}mean = \sqrt{Sensitivity \times Specificity} \tag{13}$$

Where sensitivity is the number of positive data instances that are correctly classified and divided by the number of the positive data instances (P), which is also called true positive rate [5]. Sensitivity can be represented as given in Eq. 14.

$$Sensitivity = \frac{TP}{P} \tag{14}$$

And specificity is the number of negative data instances that are correctly classified and divided by the number of the negative data instances (P), which is also called true negative rate. Specificity can be represented as given in Eq. 15.

$$Specificity = \frac{TN}{N} \tag{15}$$

The goal of the training algorithm is to maximize the G-mean value. Nature-inspired algorithms: MS, GA, and PSO keep updating the individuals until the maximum number of iterations is met and is explained as the stopping criteria in this work.

As mentioned previously in the background section, GA has three operations to update the candidate solutions which are selection, crossover, and mutation. PSO updates the candidate solutions by tracking in each iteration two values which are personal best and global best, while, MS updates the candidate solutions by using phototaxis and Levy flights.

4.3 Model Evaluation

The trained neural networks are evaluated based on the testing dataset (the whole testing graph) using the optimized weights which resulted from the training phase. The evaluation of the neural networks is based on calculating the G-mean, sensitivity and specificity values.

More attention will be given to the results of the sensitivity measurement because there is a need to be concerned about whether or not the number of instances, in a class of existent links, are correctly classified. In contrast, specificity is not required to have outstanding results as in the case of the sensitivity, since this problem does not focus on the class of nonexistent links. However, it is important to be sure that the results of the sensitivity are not obtained without considering an acceptance results of the specificity. Thus, this work also considers G-mean as an evaluation method to have an acceptable balance between sensitivity and specificity.

The overall process of the proposed methodology in this work is illustrated as shown in Fig. 4.

5 Experimental Results and Discussions

This section provides information about the experimental setups in details, and it describes the results of the proposed algorithms based on three training approaches: the full-graph approach, the random approach, and the undersampling approach. This section discusses also the comparison between the proposed algorithms against traditional classifiers and traditional link prediction methods.

5.1 Experiments Environment and Setup

For all experiments, R software is used to perform the feature extraction of link prediction. The following packages are used in R software: `igraph` and `ppiPre`.[2] Weka software version 3.8 is used to normalize the extracted datasets and to apply

[2]Interested readers refer to: (a) igraph: https://igraph.org/r/ (b) ppiPre: https://cran.r-project.org/src/contrib/Archive/ppiPre/.

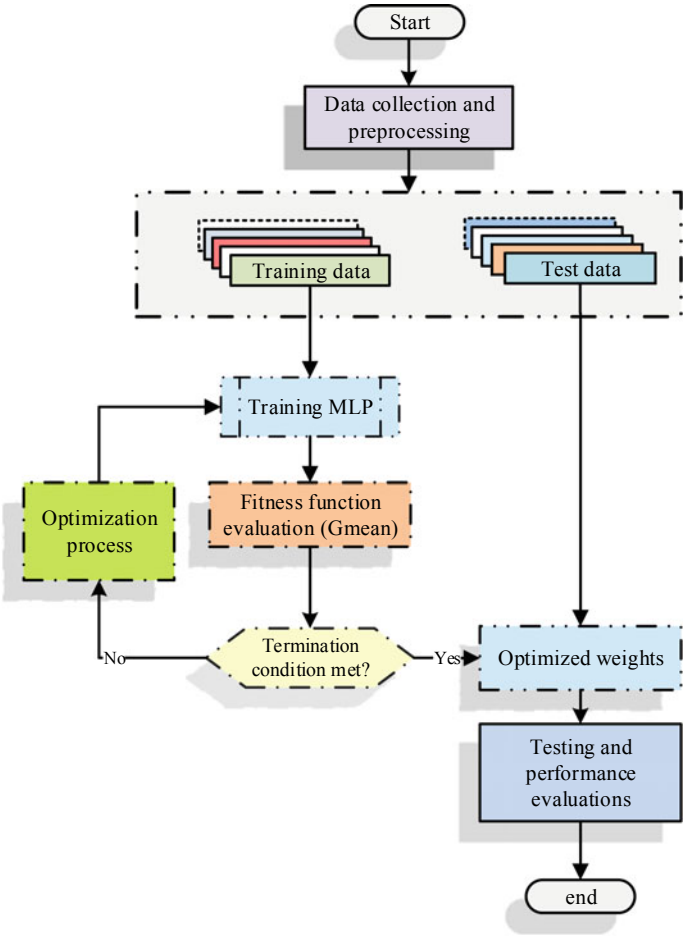


Fig. 4 Flowchart of the methodology

NB, DT, SVM algorithms [23, 28].[3] MATLAB R2010b version 9.1.0.4 is used to implement the following algorithms: MS-MLP, GA-MLP, PSO-MLP, and the traditional MLP which uses the backpropagation algorithm for training. In addition, a laptop computer with the following features is used: Windows 10 64-bit, 16GB of RAM, and Intel core i7.

[3] Readers may refer to (a) https://machinelearningmastery.com/how-to-run-your-first-classifier-in-weka/ (b) NEO Group website at http://neo.lcc.uma.es.

Table 3 Initial parameters of the metaheuristic optimizers

Algorithm	Parameter	Value
MS	The number of kept moths	2
	The index β	1.5
	Max walk step	1
	Acceleration factor	0.618
GA	Population size	50
	Crossover probability	1
	Mutation probability	0.01
	Selection	2
PSO	Population size	50
	Acceleration contacts	[2.1, 2.1]
	Inertia weights	[0.9, 0.6]
	Number of particles	50

5.2 Comparison of Evolutionary Neural Networks with Traditional Classifiers

This section is divided into three subsections based on the used training approach; it is either the full-graph approach is used for training, a random graph approach, or an undersampling approach. Each subsection will present the results of the following classification algorithms: MS-MLP, GA-MLP, PSO-MLP, MLP, NB, J48, SMO and LibSVM, where SMO and LibSVM are two different implementations of SVM. In addition, J48 is a DT developed in WEKA as an implementation of the ID3 algorithm[4] [37].

The parameters of the optimizers and classifiers are tuned empirically by selecting the parameters that give the best results from specific ranges. The initial parameters of MS, GA, and PSO are set as in Table 3 and the initial parameters of classification algorithms are set as in Table 4.

5.2.1 Results Based on the Full-Graph Approach

In this section, the results of MLP models optimized by the nature-inspired algorithms are evaluated and compared with other traditional classification algorithms based using the full-graph data for training. Table 5 shows the average sensitivity, specificity, and G-mean values of 10 independent runs for all classification algorithms. The best average sensitivity and G-mean measures for each dataset are highlighted in bold.

[4]Please refer to: http://weka.sourceforge.net/doc.dev/weka/classifiers/trees/J48.html.

Table 4 Initial parameters of classifiers

Classifiers	Parameter	Value
J48	Confidence factor	0.25
	Kernel function	RBF
SVM	Cost	53
	Gamma	0.3
	Hidden layer	1
MLP	Neurons in the hidden layer	23
BP	Learning rate	0.3
	Momentum rate	0.2

The results show that MS-MLP had the highest sensitivity results in BUP, UAL, and SMG datasets. In addition, MS-MLP had very competitive results in terms of G-mean values. GA-MLP achieved best G-mean results in BUP, UAL and SMG. It also reached good sensitivity results in all datasets except for CEG. PSO-MLP comes next with competitive results in BUP, CEG, and SMG datasets. On the other hand, the traditional classifiers had very poor results in terms of sensitivity and G-mean measures. Based on the previous results, we can conclude that the MS-MLP algorithm obtained favorable results compared to all other classification algorithms especially in 3 out of 5 datasets which are BUP, UAL, and SMG.

5.2.2 Results Based on Random Graph Approach

This section discusses the results of the evolutionary-based MLP networks and compares them to the other traditional classification algorithms based on training all algorithms using the random graph approach. Table 6 shows the average sensitivity, specificity, and G-mean of 10 independent runs for each dataset. The best average for each dataset in sensitivity and G-mean measures is highlighted in bold.

The results show that MS-MLP, GA-MLP, and PSO-MLP models outperformed all other classification algorithms in all benchmark datasets. MS-MLP obtained the highest sensitivity results in 3 datasets which are BUP, INF, and SMG, and it achieved the best G-mean in 3 datasets which are BUP, UAL, and INF. PSO-MLP achieved the best results in only one dataset which is CEG. On the other hand, we can see that the classical classification approaches performed poorly especially in terms of G-mean results. In summary, MS-MLP algorithm shows superior performance in most of the datasets compared to the other algorithms.

Table 5 Evaluation measures of the full-graph approach for training

Datasets	Measures	MS-MLP	GA-MLP	PSO-MLP	MLP	NB	SMO	J48	LibSVM
BUP	Sensitivity	**0.7124**	0.6764	0.5899	0.0079	0.079	0.000	0.034	0.000
	Specificity	0.5000	0.5367	0.5603	0.9857	0.9480	0.9990	0.9750	0.9990
	G-mean	0.5648	**0.588**	0.5459	0.03910	0.2665	0.0000	0.1798	0.0000
CEG	Sensitivity	0.4128	0.3209	**0.5174**	0.0170	0.1589	0.0140	0.1280	0.0164
	Specificity	0.8474	0.9171	0.7626	0.9908	0.9474	0.9987	0.9330	0.9986
	G-mean	0.5684	0.5340	**0.5804**	0.0747	0.3880	0.1183	0.3330	0.1278
UAL	Sensitivity	**0.718**	0.6351	0.5899	0.1652	0.4188	0.348	0.1280	0.3459
	Specificity	0.6577	0.8166	0.7418	0.9447	0.8868	0.9620	0.9330	0.9619
	G-mean	0.6408	**0.6964**	0.5939	0.2424	0.6094	0.5786	0.3330	0.5768
INF	Sensitivity	0.4081	0.4291	**0.5758**	0.0250	0.2690	0.0380	0.0800	0.0380
	Specificity	0.7866	0.7625	0.5808	0.9971	0.9720	0.998	0.9960	0.9980
	G-mean	0.4679	0.4659	0.4236	0.1258	**0.5041**	0.1947	0.2817	0.1947
SMG	Sensitivity	**0.8220**	0.8014	**0.8280**	0.1007	0.105	0.0000	0.1380	0.0000
	Specificity	0.5607	0.5727	0.5120	0.8995	0.9780	1.0000	0.9880	1.0000
	G-mean	**0.6363**	**0.6381**	0.6074	0.0156	0.3205	0.0000	0.3692	0.0000

Table 6 Evaluation measures of the random sampling approach

Dataset	Measures	MS-MLP	GA-MLP	PSO-MLP	MLP	NB	SMO	J48	LibSVM
BUP	Sensitivity	**0.5393**	0.4539	0.2899	0.0112	0.0790	0.0000	0.0000	0.0000
	Specificity	0.5637	0.6333	0.7328	0.9865	0.9480	0.9980	0.9960	0.9980
	G-mean	**0.5514**	0.4428	0.3960	0.0646	0.2740	0.0000	0.0000	0.0000
CEG	Sensitivity	0.1447	0.1919	**0.1988**	0.0121	0.0740	0.0000	0.0000	0.0000
	Specificity	0.9702	0.9603	0.9638	0.9957	0.9810	1.0000	1.0000	1.0000
	G-mean	0.3610	0.4216	**0.4322**	0.0496	0.2960	0.0000	0.0000	0.0000
UAL	Sensitivity	0.6005	0.5518	**0.6972**	0.1052	0.3340	0.3510	0.6090	0.3460
	Specificity	0.8932	0.9046	0.7731	0.8961	0.9640	0.9610	0.9150	0.9620
	G-mean	**0.7271**	0.7011	0.7054	0.0476	0.5670	0.5810	0.7460	0.5770
INF	Sensitivity	**0.3772**	0.2808	0.3170	0.0114	0.2960	0.1150	0.0490	0.0110
	Specificity	0.8525	0.9341	0.8893	0.9990	0.9150	0.9850	0.9950	0.9990
	G-mean	**0.5180**	0.4930	0.5151	0.0602	0.5200	0.3370	0.2210	0.1050
SMG	Sensitivity	**0.4228**	0.3216	0.2826	0.1023	0.1000	0.0000	0.1110	0.0000
	Specificity	0.8267	0.9612	0.9692	0.8963	0.9760	1.0000	0.9970	1.0000
	G-mean	0.5294	**0.5540**	0.5182	0.0374	0.3120	0.0000	0.3330	0.0000

5.2.3 Results Based on Undersampling Approach

This section reports and discusses the results of the metaheuristic-based MLP networks and the other classical classifiers after training them on undersampled datasets with different ratios. That is, the ratios of the major class to the minor class will be 1:1, 1:2, 1:3, 1:4, and 1:5. The best results based on the G-mean value is reported for each algorithm along with its specificity, sensitivity, and their corresponding undersampling ratio.

The results in Table 7 show that evolutionary neural networks are very competitive to each other and with the decision tree algorithm J48. In addition, the results of the neural networks show more stability across the datasets while the results of J48 are more fluctuating. For example, it can be seen that J48 obtained very poor results for CEG dataset.

5.3 Results based on the Comparison Between All Model Development Approaches

This section discusses the results by comparing between all model development approaches: the full-graph approach, the random graph approach, and the undersampling approach. Based on the previous experimental results, Table 8 is constructed. It can be noticed that the results of the full-graph and the undersampling method are better than those of the random graph approach. It can be noticed also that the results of the undersampling approach are slightly better than the full-graph approach. Taking into consideration, the reduction in computation time in the case of the undersampling approach, this gives extra credit for this approach over the other approaches.

Moreover, the second important point that should be considered when comparing between all model development approaches is the running time of the training process of the classification models. The running time of the training process is one of the most challenging problems in link prediction problem due to the huge size of the training datasets. The running time of the classification process for the three training approaches is reported in Table 9.

It can be noticed that the full-graph approach consumed more time than other approaches, especially when the size of datasets increases, such as in the case of SMG dataset, which consumed around 11 hours more than the other approaches. Thus, this work recommended using undersampling approach for two reasons: first it lessen the problem of imbalanced class distribution and it improves the classification results, and it significantly reduces the training time of the classification models.

Table 7 Evaluation measures based on the best undersampling ratios for all datasets

Data	Evolution measures	MS-MLP	GA-MLP	PSO-MLP	MLP	NB	SMO	J48	LibSVM
BUP	Sensitivity	**0.6910**	0.6831	0.5315	0.2398	0.1359	0.0485	0.5506	0.0485
	Specificity	0.4855	0.4700	0.6705	0.8333	0.9535	**0.9606**	0.6619	0.9581
	G-mean	0.5568	0.5383	0.5605	0.2853	0.3600	0.2159	**0.6037**	0.2157
	Ratio	1:3	1:3	1:2	1:1	1:3	1:3	1:5	1:3
CEG	Sensitivity	**0.4867**	0.4228	0.4340	0.3130	0.1548	0.1156	0.0849	0.1462
	Specificity	0.7849	0.8636	0.8745	0.8041	0.9505	0.9537	**0.9819**	0.9624
	G-mean	0.5780	0.5904	**0.6100**	0.3556	0.3835	0.3363	0.2887	0.3765
	Ratio	1:1	1:1	1:5	1:2	1:5	1:4	1:2	1:4
UAL	Sensitivity	0.6805	0.6838	0.7576	0.3320	0.4198	0.1929	**0.7665**	0.4198
	Specificity	0.7759	0.7933	0.6187	0.8516	0.8814	**0.9710**	0.8295	0.9463
	G-mean	0.7005	0.7204	0.6383	0.3442	0.6083	0.4328	**0.7974**	0.6303
	Ratio	1:4	1:2	1:4	1:3	1:5	1:1	1:4	1:2
INF	Sensitivity	0.4732	0.3667	0.4430	**0.5049**	0.2681	0.2622	0.4991	0.2391
	Specificity	0.4264	0.4910	0.5293	0.5049	0.5106	**0.9743**	0.7326	0.4842
	G-mean	**0.7005**	0.6820	0.6383	0.4194	0.6058	0.5054	0.6047	0.5248
	Ratio	1:4	1:4	1:4	1:1	1:4	1:1	1:1	1:4
SMG	Sensitivity	0.6718	0.7695	0.7226	0.4783	0.3420	0.4120	**0.7930**	0.4593
	Specificity	0.7740	0.7324	0.7305	0.6816	0.9490	**0.9540**	0.7480	0.9392
	G-mean	0.7004	0.7324	0.7305	0.3859	0.5697	0.6269	**0.7702**	0.6568
	Ratio	1:4	1:5	1:5	1:1	1:1	1:1	1:1	1:2

Table 8 Evaluation results of evolutionary neural networks of three different training approaches

Approaches		The full-graph			Random graph			Undersampling		
Data	Measures	MS-MLP	GA-MLP	PSO-MLP	MS-MLP	GA-MLP	PSO-MLP	MS-MLP	GA-MLP	PSO-MLP
BUP	Sensitivity	**0.7124**	0.6764	0.5899	0.5393	0.4539	0.2899	0.6910	0.6831	0.5315
	Specificity	0.5000	0.5367	0.5603	0.5637	0.6333	**0.7328**	0.4855	0.4700	0.6705
	G-mean	0.5648	**0.5880**	0.5459	0.5514	0.4428	0.3960	0.5568	0.5383	0.5605
CEG	Sensitivity	0.4128	0.3209	**0.5174**	0.1447	0.1919	0.1988	0.4867	0.4228	0.4340
	Specificity	0.8474	0.9171	0.7626	**0.9702**	0.9603	0.9638	0.7849	0.8636	0.8745
	G-mean	0.5684	0.5340	0.5804	0.3610	0.4216	0.4322	0.5780	0.5904	**0.6100**
UAL	Sensitivity	0.7180	0.6351	0.5899	0.6005	0.5518	0.6972	0.6805	0.6838	**0.7576**
	Specificity	0.6577	0.8166	0.7418	0.8932	**0.9046**	0.7731	0.7759	0.7933	0.6187
	G-mean	0.6408	0.6964	0.5939	**0.7271**	0.7011	0.7054	0.7005	0.7204	0.6383
INF	Sensitivity	0.4081	0.4291	**0.5758**	0.3772	0.2808	0.3170	0.4732	0.3667	0.4430
	Specificity	0.7866	0.7625	0.5808	0.8525	**0.9341**	0.8893	0.4264	0.4910	0.5293
	G-mean	0.4679	0.4659	0.4236	0.5180	0.4930	0.5151	**0.7005**	0.6820	0.6383
SMG	Sensitivity	0.8220	0.8014	**0.8280**	0.4228	0.3216	0.2826	0.6718	0.7695	0.7226
	Specificity	0.5607	0.5727	0.5120	0.8267	0.9612	**0.9692**	0.7740	0.7324	0.7305
	G-mean	0.6363	0.6381	0.6074	0.5294	0.5540	0.5182	0.7004	**0.7324**	0.7305

Table 9 Time evaluation of all approaches for all datasets

Dataset	Approaches	MS-MLP	GA-MLP	PSO-MLP
BUP	The full-graph	2:44:00	2:41:50	2:45:50
	Random graph	2:43:00	2:41:40	2:44:10
	Undersampling	2:42:30	2:39:20	2:48:10
CEG	The full-graph	3:29:20	3:27:50	3:32:10
	Random graph	2:44:00	2:41:50	2:48:50
	Undersampling	2:40:40	2:38:40	2:45:00
UAL	The full-graph	3:36:30	3:36:30	3:46:00
	Random graph	2:41:10	2:48:10	2:56:00
	Undersampling	2:41:30	2:38:40	2:42:50
INF	The full-graph	4:15:50	4:15:50	4:27:30
	Random graph	2:45:10	2:41:30	2:53:30
	Undersampling	2:45:20	2:40:40	2:46:00
SMG	The full-graph	13:36:40	13:22:40	13:30:40
	Random graph	2:40:50	2:41:20	2:42:40
	Undersampling	2:46:10	2:45:20	2:51:30

5.4 Comparison with Traditional Link Prediction Methods

In this section, the neural network models MS-MLP, GA-MLP, and PSO-MLP are compared with the traditional link prediction methods: common neighbors, Adamic/Adar similarity, Jaccard similarity, Salton index, preferential attachment index, Leicht-Holme-Newman index, hub depressed index, hub promoted index, and Sørensen index. The results of all approached are obtained based on the full-graph approach.

The quantitative results of sensitivity, specificity, and the G-mean measures for all approaches are given in Table 10. The best results of sensitivity and G-mean measures for each dataset are highlighted in bold.

From Table 10, it can be clearly seen that the results of MS-MLP, GA-MLP, and PSO-MLP outperform all traditional link prediction methods in all datasets. Thus, it can be concluded that the combination of the traditional link prediction in the form of neural network model will outperform the usage of traditional link prediction methods separately.

6 Conclusions and Future Directions

In this chapter, neural networks optimized by three nature-inspired algorithms (MS, GA, and PSO) were proposed and developed for the link prediction problem. The optimized neural networks are used as classifiers for the prepared link prediction

Table 10 Evaluation results of the neural network-based models compared to the traditional link prediction methods

Methods	Evaluation measures	BUP	CEG	UAL	INF	SMG
MS-MLP	Sensitivity	**0.7124**	**0.4128**	**0.7180**	**0.4081**	**0.8220**
	Specificity	0.5000	0.8474	0.6577	0.7866	0.5607
	G-mean	**0.5648**	**0.5684**	**0.6408**	**0.4679**	**0.6363**
GA-MLP	Sensitivity	**0.6764**	**0.3209**	**0.6351**	**0.4291**	**0.8014**
	Specificity	0.5367	0.9171	0.8166	0.7625	0.5727
	G-mean	**0.5880**	**0.5340**	**0.6964**	**0.4659**	**0.6381**
PSO-MLP	Sensitivity	**0.5899**	**0.5174**	**0.5899**	**0.5758**	**0.8280**
	Specificity	0.5603	0.7626	0.7418	0.5808	0.5120
	G-mean	**0.5459**	**0.5804**	**0.5939**	**0.4236**	**0.6074**
Common neighbors	Sensitivity	0.0337	0.0256	0.0612	0.0217	0.0020
	Specificity	0.9535	0.9967	0.9947	0.9996	0.9999
	G-mean	0.1793	0.1597	0.2467	0.1473	0.0451
Adamic/Adar	Sensitivity	0.0000	0.0070	0.0024	0.0380	0.0030
	Specificity	0.9934	0.9994	0.9993	0.9982	0.9999
	G-mean	0.0000	0.0835	0.0485	0.1947	0.0552
Jaccard	Sensitivity	0.0000	0.0000	0.0000	0.0000	0.0000
	Specificity	0.9925	0.9981	0.9925	0.9990	0.9943
	G-mean	0.0000	0.0000	0.0000	0.0000	0.0000
Salton	Sensitivity	0.0000	0.0070	0.0047	0.0036	0.0010
	Specificity	0.9779	0.9918	0.9816	0.9968	0.9892
	G-mean	0.0000	0.0832	0.0680	0.0600	0.0317
Preferential attachment	Sensitivity	0.0449	0.0023	0.0141	0.0145	0.0010
	Specificity	0.9922	0.9994	0.9989	0.9993	1.0000
	G-mean	0.2112	0.0482	0.1188	0.1202	0.0319
Leicht-Holme-Newman	Sensitivity	0.0000	0.0000	0.0000	0.0000	0.0000
	Specificity	0.9931	0.9983	0.9930	0.9991	0.9944
	G-mean	0.0000	0.0000	0.0000	0.0000	0.0000
Hub depressed	Sensitivity	0.0000	0.0279	0.0447	0.0506	0.0163
	Specificity	0.9655	0.9731	0.9321	0.9877	0.9774
	G-mean	0.0000	0.1648	0.2041	0.2236	0.1262
Hub promoted	Sensitivity	0.0000	0.0023	0.0047	0.0000	0.5000
	Specificity	0.9902	0.9964	0.9890	0.9984	0.9930
	G-mean	0.0000	0.0481	0.0682	0.0000	0.7046
Sørensen index	Sensitivity	0.0000	0.0070	0.0047	0.0036	0.0010
	Specificity	0.9819	0.9938	0.9842	0.9975	0.9908
	G-mean	0.0000	0.0833	0.0681	0.0601	0.0315

datasets. For this, features were extracted from the datasets using the traditional link prediction methods: common neighbors, Adamic/Adar similarity, Jaccard similarity, Salton index, preferential attachment index, Leicht-Holme-Newman index, hub depressed index, hub promoted index, Sørensen index, and centrality measures. Moreover, this work handled the imbalanced dataset distribution of link prediction to avoid poor performance based on two different approaches: sampling techniques (random and undersampling) and using G-mean as a fitness function in the neural networks. In addition, three training approaches for the developed neural network models where experimented: in the first a full-graph was used for training, in the second a random graph was used, and in the third an undersampling approach was used. The neural networks resulted from the three training approaches were evaluated and compared to other popular classifiers and other traditional link prediction methods. The evaluation results show that neural networks developed based on the full-graph and the undersampling approach are very competitive and outperform the other methods in majority of the datasets. However, the advantage of the undersampling approach is a significant reduction in running time required for training the neural network model. Overall, we can conclude that nature-inspired neural networks trained based on the undersampling approach can be a promising candidate for the link prediction problem.

References

1. Adamic LA, Adar E (2003) Friends and neighbors on the web. Soc Netw 25(3):211–230
2. Al Hasan M, Chaoji V, Salem S, Zaki M (2006) Link prediction using supervised learning. In: SDM06: workshop on link analysis, counter-terrorism and security
3. Ala'M A-Z, Faris H et al (2017) Spam profile detection in social networks based on public features. In: 2017 8th International conference on information and communication systems (ICICS). IEEE, pp 130–135
4. Ala'M A-Z, Faris H, Hassonah MA et al (2018) Evolving support vector machines using whale optimization algorithm for spam profiles detection on online social networks in different lingual contexts. Knowl-Based Syst 153:91–104
5. Ala'M A-Z, Rodan A, Alazzam A (2018) Classification model for credit data. In: 2018 Fifth HCT information technology trends (ITT). IEEE, pp 132–137
6. Alian S, Ghatasheh N et al (2014) Multi-agent swarm spreading approach in unknown environments. Int J Comput Sci Issues (IJCSI) 11(2):160
7. Azzini A, Tettamanzi AGB (2011) Evolutionary ANNs: a state of the art survey. Intelligenza Artificiale 5(1):19–35
8. Barabási A-L, Albert R (1999) Emergence of scaling in random networks. Science 286(5439):509–512
9. Chawla NV (2009) Data mining for imbalanced datasets: an overview. In: Data mining and knowledge discovery handbook. Springer, pp 875–886
10. Chen B, Chen L (2014) A link prediction algorithm based on ant colony optimization. Appl Intell 41(3):694–708
11. Chen H, Li X, Huang Z (2005) Link prediction approach to collaborative filtering. In Proceedings of the 5th ACM/IEEE-CS joint conference on digital libraries, 2005. JCDL'05. IEEE, pp 141–142
12. Chowdhury GG (2010) Introduction to modern information retrieval. Facet Publishing

13. Clauset A, Moore C, Newman MEJ (2008). Hierarchical structure and the prediction of missing links in networks. Nature 453(7191):98
14. Davis L (1991) Handbook of genetic algorithms. CUMINCAD
15. Ding S, Su C, Yu J (2011) An optimizing bp neural network algorithm based on genetic algorithm. Artif Intell Rev 36(2):153–162
16. Eberhart R, Kennedy J (1995) A new optimizer using particle swarm theory. In: MHS'95. Proceedings of the sixth international symposium on micro machine and human science. IEEE, pp 39–43
17. Esslimani I, Brun A, Boyer A (2011) Densifying a behavioral recommender system by social networks link prediction methods. Soc Netw Anal Mining 1(3):159–172
18. Faris H, Aljarah I et al (2015) Optimizing feedforward neural networks using krill herd algorithm for e-mail spam detection. In: 2015 IEEE Jordan conference on applied electrical engineering and computing technologies (AEECT). IEEE, pp 1–5
19. Gao F, Musial K, Cooper C, Tsoka S (2015) Link prediction methods and their accuracy for different social networks and network metrics. Sci Program 2015:1
20. Gilbert E, Karahalios K (2009) Predicting tie strength with social media. In: Proceedings of the SIGCHI conference on human factors in computing systems. ACM, pp 211–220
21. Golberg DE (1989) Genetic algorithms in search, optimization, and machine learning. Addison Wesley, Reading
22. Gupta JND, Sexton RS (1999). Comparing backpropagation with a genetic algorithm for neural network training. Omega 27(6):679–684
23. Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH (2009) The WEKA data mining software: an update. SIGKDD Explor 11(1):10–18
24. Ismail AT, Sheta A, Al-Weshah M (2008) A mobile robot path planning using genetic algorithm in static environment. J Comput Sci 4(4):341–344
25. Jaccard P (1901) Étude comparative de la distribution florale dans une portion des alpes et des jura. Bull Soc Vaudoise Sci Nat 37:547–579
26. Kennedy J (1997) The particle swarm: social adaptation of knowledge. In: IEEE International conference on evolutionary computation, 1997. IEEE, pp 303–308
27. Kennedy J (2011) Particle swarm optimization. In: Encyclopedia of machine learning. Springer, pp 760–766
28. Kubat M, Holte RC, Matwin S (1998). Machine learning for the detection of oil spills in satellite radar images. Mach Learn 30(2-3):195–215
29. Kuo T-T, Yan R, Huang Y-Y, Kung P-H, Lin S-D (2013). Unsupervised link prediction using aggregative statistics on heterogeneous social networks. In: Proceedings of the 19th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 775–783
30. Liben-Nowell D, Kleinberg J (2007) The link-prediction problem for social networks. J Am Soc Inf Sci Technol 58(7):1019–1031
31. Lichtenwalter RN, Lussier JT, Chawla NV (2010) New perspectives and methods in link prediction. In: Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, pp 243–252
32. Liu H, Hu Z, Haddadi H, Tian H (2013) Hidden link prediction based on node centrality and weak ties. EPL (Europhys Lett) 101(1):18004
33. Lorrain F, White HC (1971) Structural equivalence of individuals in social networks. J Math Soc 1(1):49–80
34. Madain A, Ala'M A-Z, Al-Sayyed R et al (2017) Online social networks security: Threats, attacks, and future directions. In: Social media shaping e-publishing and academia. Springer, pp 121–132
35. Murata T, Moriyasu S (2007) Link prediction of social networks based on weighted proximity measures. In: Proceedings of the IEEE/WIC/ACM international conference on web intelligence. IEEE Computer Society, pp 85–88
36. Patro S, Sahu KK (2015) Normalization: a preprocessing stage. arXiv preprint arXiv:1503.06462

37. Quinlan R (1993) C4.5: programs for machine learning. Morgan Kaufmann Publishers, San Mateo, CA (1993)
38. Redner S (2008) Networks: teasing out the missing links. Nature 453(7191):47
39. Rodan A, Faris H et al (2016) Optimizing feedforward neural networks using biogeography based optimization for e-mail spam identification. Int J Commun Netw Syst Sci 9(1):19–28
40. Schall D (2014) Link prediction in directed social networks. Soc Netw Anal Mining 4(1):157
41. Schifanella R, Barrat A, Cattuto C, Markines B, Menczer F (2010) Folks in folksonomies: social link prediction from shared metadata. In: Proceedings of the third ACM international conference on Web search and data mining. ACM, pp 271–280
42. Shibata N, Kajikawa Y, Sakata I (2012) Link prediction in citation networks. J Am Soc Inf Sci Technol 63(1):78–85
43. Taskar B, Wong M-F, Abbeel P, Koller D (2004) Link prediction in relational data. In: Advances in neural information processing systems, pp 659–666
44. Wang C, Satuluri V, Parthasarathy S (2007) Local probabilistic models for link prediction. In: Seventh IEEE international conference on data mining (ICDM 2007). IEEE, pp 322–331
45. Wang G-G (2016) Moth search algorithm: a bio-inspired metaheuristic algorithm for global optimization problems. Memetic Comput 1–14
46. Xie Z (2005) Centrality measures in text mining: prediction of noun phrases that appear in abstracts. In :Proceedings of the ACL student research workshop. Association for Computational Linguistics, pp 103–108
47. Yamany W, Fawzy M, Tharwat A, Hassanien AE (2015) Moth-flame optimization for training multi-layer perceptrons. In: 2015 11th International computer engineering conference (ICENCO). IEEE, pp 267–272

Evolving Genetic Programming Models for Predicting Quantities of Adhesive Wear in Low and Medium Carbon Steel

Rana Faris, Bara'a Almasri, Hossam Faris, Faris M. AL-Oqla and Doraid Dalalah

Abstract Wear loss prediction is still essential in various industrial applications particularly the cutting tools. This process is quite sophisticated due to the relation between the interrelated variables. In this work, a genetic programming optimization model for predicting and optimizing the quantities of adhesive wear in low and medium carbon steel was generated. Carbon steel material was subjected to dry sliding wear experiments using a pin-on-disc module. Several parameters including the applied load, sliding speed and time were involved in the model. The proposed model was capable of predicting and optimizing the wear loss in carbon steel and was evaluated and tested using different performance criteria to ensure its reliability. The generated model can be utilized to monitor wear in mechanical components without requiring any human efforts to enhance the monitoring efficiency and reduce human errors.

Keywords Genetic programming · Adhesive wear · Carbon steel · Prediction

R. Faris · B. Almasri
Industrial Engineering Department, Jordan University of Science and Technology, Irbid, Jordan
e-mail: eng.ranafaris90@gmail.com

H. Faris (✉)
King Abdullah II School for Information Technology, The University of Jordan, Amman, Jordan
e-mail: hossam.faris@ju.edu.jo

F. M. AL-Oqla
Department of Mechanical Engineering, Faculty of Engineering, The Hashemite University, Zarqa 13133, Jordan
e-mail: fmaloqla@hu.edu.jo

D. Dalalah
Industrial Engineering and Engineering Management, University of Sharjah, Sharjah, United Arab Emirates
e-mail: ddalalah@sharjah.ac.ae

S. Mirjalili et al. (eds.), *Evolutionary Machine Learning Techniques*,
Algorithms for Intelligent Systems, https://doi.org/10.1007/978-981-32-9990-0_7

1 Introduction

To stay competitive in modern industries and markets, engineers have to keep coming up with new techniques and materials and work on improving processes to develop higher-quality products. Many traditional materials have helped in engineering applications for long period of time. However, they are cautiously improved to contribute meeting the demand of weight reduction, performance enhancement as well as customer satisfaction attributes [3]. Thus, selecting a proper material type for a particular application requires possessing desired characteristics. This in order make the continues improvement as well as predicting and controlling the manufacturing characteristics of materials using modern techniques are mandatory for their improvement process to maintain competitive for modern design possibilities [4]. It was estimated that there are more than 80,000 material types including metallic alloys and nonmetallic engineering materials [15].

Wear phenomenon is still considered as of the serious problems in the design of mechanical components particularly the cutting tools, as it usually make defects in the mechanical parts resulting in a functionality failure. Several factors, in fact, affect the wear phenomenon such as the applied load on the contact area, sliding speed and sliding time. However, some other conditions are hard to be controlled such as temperature and vibration, a problem that can be solved by ensuring appropriate mechanical rigidity when measuring wear. Since wear is a system derived property, the wear results and correct interpretation of the wear data are crucial issue. Due to the complex nature of wear and large number of affecting parameters, the process of investigation has been delayed and resulted in isolated studies towards a particular wear mechanism [14]. Some common wear mechanisms include:

- Adhesive wear
- Abrasive wear
- Fatigue wear
- Fretting wear
- Erosive wear.

Adhesive wear which is site of interest in this study is defined as the resulting material loss of relative sliding or rolling movement of two contacting solid surfaces; when contact pressure is high, it causes permanent deformation of rubbing component. The adhesion wear is usually considered as one of the most spread wears as it accounts for 15% of the industrial wear [2] , which occurs when the surfaces are on a sliding movement one over the other so that the pressure between the adjacent projections is enough to yield some local formation adhesion and plastic [9].

A commonly used configuration to test adhesive wear specifications is pin-on-disc method due to its simplicity. In order to reduce the effect of many uncontrollable variables, some practices could be followed before and after conducting a wear test using the pin-on-disc. For example, before implementing pin-on-disc configuration, the rotating disc must be grind and polished to a mirror surface to erase the tracks of previous work and prevent vibration problem along with realizing correct measures.

However, the process stays complex and the effect of uncontrollable variables remains. Most industries face the problem of wear on parts in services, and any moving part in services will suffer from wear at contact point hence it is needed to be replaced which costs money and causes downtime for the equipment. Therefore, developing an accurate wear prediction model can dramatically enhance maintaining surface quality, reduce cost and increase productivity [21].

Typically, mathematical models were developed in order to simulate material behaviours and properties. These models were derived from a number of experimental data obtained in special conditions and with controllable variables [1]. Besides mathematical modelling, several researchers have developed wear prediction models based on artificial intelligence (AI) and machine learning approaches. One of the most adopted approaches is artificial neural networks (ANN). In [13], authors used ANN-based prediction model for predicting the quantities of wear loss for Mo coating in a pin-on-plate configuration. The Mo coatings were subjected to sliding wear under various loads and environment conditions. ANN showed acceptable performance results. Another use of ANN was utilized to estimate the wear loss of AA 6351 aluminium alloy with different ageing temperature and environmental conditions using a pin-on-disc module [8]. Other methods were also reported for predicting the wear loss other material based upon ANN, neuro-fuzzy as well as others [7, 20, 21, 24]. Although neural networks have advantages like their predictive power, they still have some drawbacks such as their limitation in giving an insight about the undergoing interaction between the interrelated variables. Thus, they much work as a block-box models.

Genetic programming (GP), on the other hand, is another bio-inspired AI approach which evolves a number of mathematical equations through evolutionary operators like crossover and mutation. GP has in addition extra advantages compared to neural networks which are the capabilities of giving an insight into the underlying system by producing compact and being easy to evaluate mathematical models. GP is then reliable and logical in modelling various engineering applications under uncertainty environments.

Therefore, the aim of this study is to generate and optimize a GP model capable of predicting the quantities of adhesive wear for both low and medium carbon steel. The results of this model will be as a function of the load, sliding speed and sliding time based upon the pin-on-disc configuration. This model would dramatically enhance developing and conducting suitable monitoring systems for wear when using such type of steel and imitate and being a guideline for wider monitoring systems regarding other material types in various industrial applications. The generated model can be automatically constructed by a genetic algorithm to monitor wear in mechanical components without requiring any human efforts to enhance the monitoring efficiency and reduce human errors.

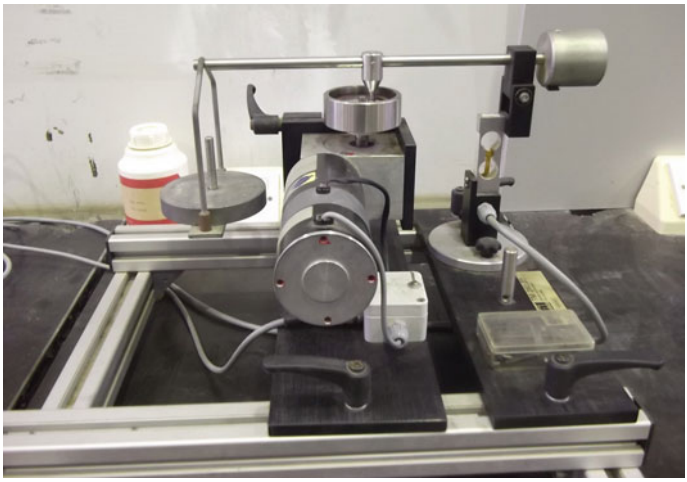


Fig. 1 Pin-on-disc experimental module

2 Pin-on-Disc Module Description

The pin-on-disc experimental module is usually used to investigate the friction forces between a vertical pin and rotating disc. The end of each pin specimen made of steel, aluminium or brass rubs against the rotating, hardened, ground steel disc. The contact pressure between the parts can be controlled utilizing a lever with weights. The friction force is estimated utilizing a strain gauge force transducer in the bearing point of the loading mechanism. The disc is encased in an open cup that can be loaded up with various lubricants for the experiments. The Pin-on-disc experimental module used in this work is shown in Fig. 1. Some technical specifications of the utilized pin-on-disc in this work can be summarized in the following points:

- Friction disc: stainless steel, hardened, ground
- Operating speed: 0–0.42 m/s
- Friction pin diameter: 4 mm
- Load: 0 to maximum load of 80 N
- Friction force measuring range: 0–450 N.

For more information on this module, the reader can refer to (https://www.gunt.de).

3 Symbolic Regression via Genetic Programming

The concept of symbolic regression (SR) was first coined by Koza in [18]. SR is a type of supervised regression modelling technique. Unlike conventional regression

analysis methods, the idea of SR is to search the space for a structure of the model (mathematical formulas that are represented as trees) that best fits a given training data set along with its parameters. Therefore, SR does not impose any priori assumptions on the structure of the model.

Genetic programming is widely applied as SR method [19]. GP is a special extension of the well-known Darwinian-inspired genetic algorithm. GP evolves automatically a predetermined number of mathematical models in order to describe a number of independent input variables x and some numeric weights w as a function $f(x, w)$ while minimizing some error criteria.

Implementing SR via GP has some powerful modelling advantages [17]. In contrast to other modelling techniques like neural networks, GP generates interpretable mathematical models that are relatively more compact and easier to evaluate and understand. In GP, the complexity of the generated produced tree models can be constrained by specifying the maximum width and length of the trees. In addition, GP enjoys a powerful embedded and automatic feature selection method. In GP, strong feature that helps in producing strong models will survive while unrelated and weak features will be dropped and not used in generating new models. In the next section, the evolutionary cycle of GP is described.

4 Evolutionary Cycle of Genetic Programming

GP can be described as an evolutionary cycle where its inherited GA processes (i.e. initialization, fitness evaluation and reproduction operators) are iteratively applied based on some learning examples. The main processes of the GP evolutionary cycle can be explained as follows (Fig. 2):

1. **Initialization**: In its initial state, GP starts by initializing a predefined number of individuals randomly. Each of these individuals is also called a computer program or a model or a solution. GP individuals are commonly represented as trees or LISP expression. Figure 3 shows a simple example of a GP model which is equivalent to Eq. 1. The set of individuals that GP starts with is called population. The population size parameter of GP defines their number.

$$z = \frac{(X * 2) + 5}{\sin(Y)} \tag{1}$$

2. **Fitness evaluation**: After initialization, the quality of the generated individuals is assessed using a predetermined evaluation measure. In this work, we use the Pearson R^2 to measure the correlation between the predicted values obtained by the candidate individual and the desired output values. By this process, the quality of the generated individuals in terms of prediction power is determined.
3. **Reproduction** :At this stage, a set of GP operator are applied on probabilistically selected individuals according to their fitness and their fitness value. The way this process is conducted is known as the selection mechanism. The higher value the individual has the more probable to be selected. Genetic operations include:

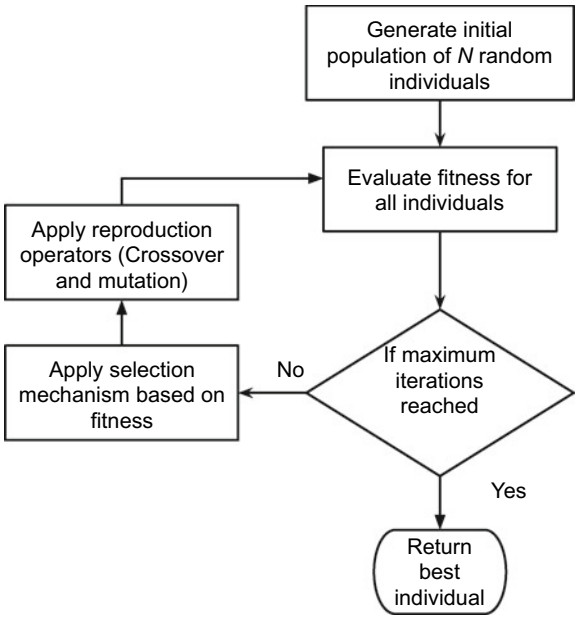


Fig. 2 Flow chart of the evolutionary process based GP

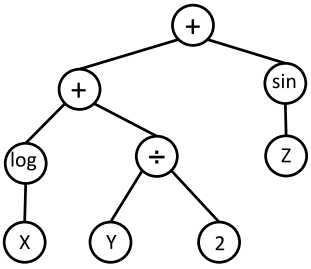


Fig. 3 A simple GP model represented as tree

a. *Crossover:* this operator is considered as one of the most important genetic operators. By applying crossover, two individuals exchange some parts of their structure. There are different types of crossover operators range from simple to complex ones. Figure 4 illustrates a simple example of crossover where two individuals exchange randomly selected subtrees from their model.
b. *Mutation*: This operation is applied on a single individual by randomly choosing a point in the tree representation of the individual and replacing it with another randomly generated subtree as shown in Fig. 5. Usually, the mutation operator is applied with a rate that is much smaller that the one for crossover. That is to avoid significant destruction of quality across individuals caused by the major changes made by the mutation operator. On the other side, mutation operator helps in maintaining the diversity over the

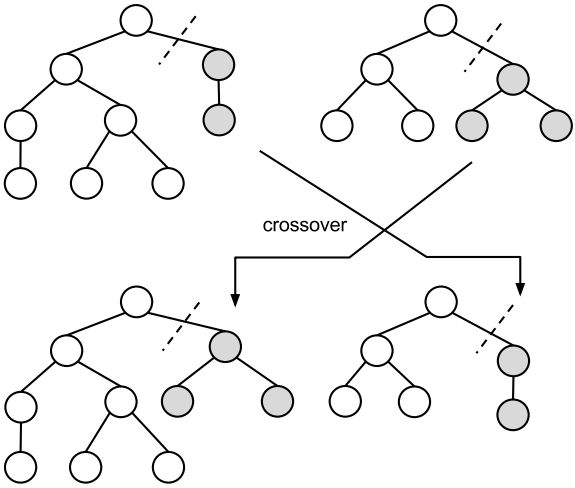


Fig. 4 An example of GP crossover operator

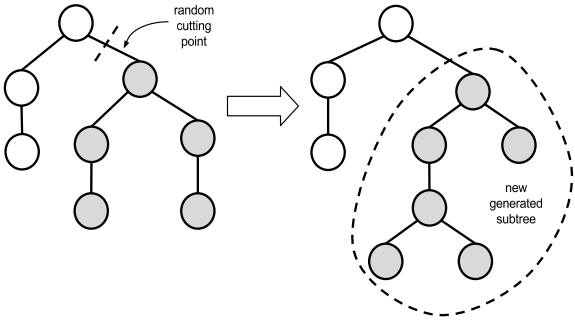


Fig. 5 Example of GP mutation operator

course of iterations. After applying genetic operations iteratively, the new generated populations replace the old ones.

c. *Elitism*: This is a simple mechanism commonly applied to preserve best n individuals in the population and transfer them to the next generation without any modification. In GP, the value of n is usually set to 1, 2 or 3.

4. **Termination condition**: New populations are generated iteratively by the last process until one of the following conditions is met;

   - Number of generations is reached, which is a predetermined number specified by the user to end the iterative process after a number of loops.
   - An individual with a specific fitness value is reached. Finally, the best-so-far individual is chosen to be the solution of the problem.

In the last two decades, GP has been applied to a wide range of industrial and manufacturing problems showing high modelling capability [6, 10–12, 16, 22].

5 Model Evaluation

When the maximum number of iteration is reached or if the model satisfied a predefined quality level, then the evolutionary process should be stopped; otherwise, another model structure should be tried again. To assess the quality and the prediction power of the generated GP models in predicting the wearing of carbon steel, two measurements are applied which are the variance accounted for (VAF) and root mean squared error (RMSE). VAF and RMSE are represented in Eqs. 2 and 3, respectively.

$$VAF = \left[1 - \frac{var(y - \hat{y})}{var(y)}\right] \times 100\% \tag{2}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{3}$$

Where y *and* $\hat{y}$ are the actual output and the predicted value obtained by GP model, respectively. n is the total number of experiments.

6 Experiments and Results

6.1 Experiments Preparation and Data Collection

Two samples of different types of steel with a known carbon percentage were prepared:

- Medium carbon steel.
- Low carbon steel.

In order to conduct a wear test, working variables were specified. In the experiments, variables were limited to: sliding speed, sliding time and load. Other variables were difficult to be controlled although they affect wearing process such as working environment temperature. The other variable was vibration of the machine arm due to different operating speeds and different loads. This problem was solved by placing a piece of tissue over the arm of the machine before setting the load. Before starting the experiments, carbon steel samples were grinded to an even surface using sandpaper. Moreover, after taking each wear measurement, machine rotating disc was grinded and polished to a mirror surface to erase the tracks of former work then the next experiment starts. Twenty-seven experiments were conducted for each of low and medium samples with different values of operating variables each time. Figures 6 and 7 show the values for the operating variables along with their corresponding wear loss quantities.

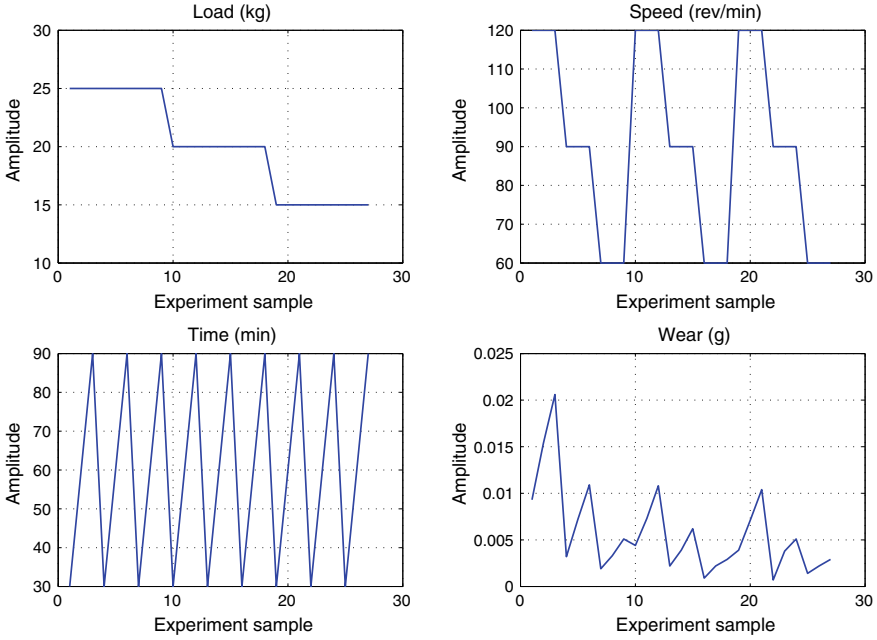


Fig. 6 The input–output data for the low-carbon wear model

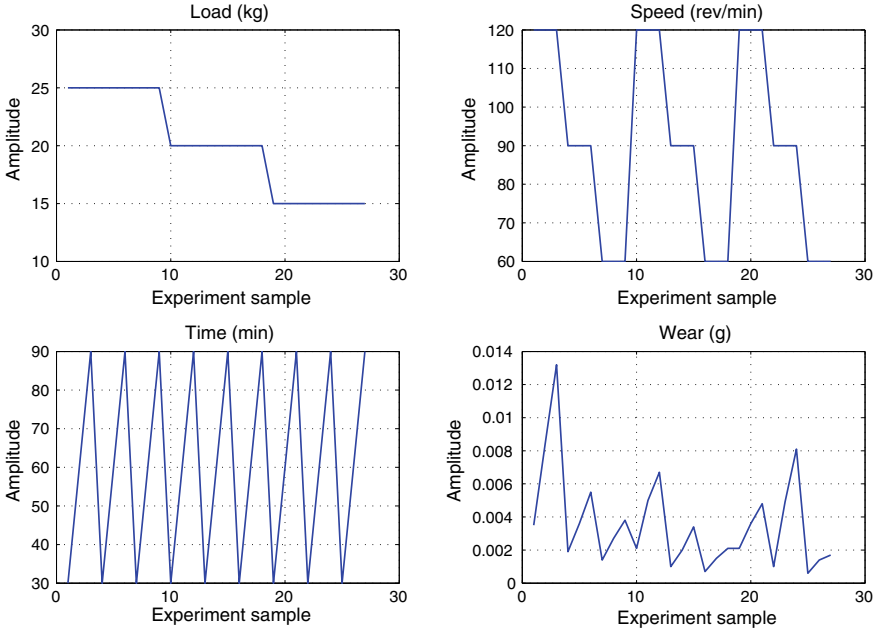


Fig. 7 The input–output data for the medium-carbon wear model

Table 1 Settings of GP parameters

Parameter	Value
Mutation rate	15%
Population size	1000
Maximum generations	200
Selection mechanism	Tournament selector
Elites	1
Operators	{+, -, *, /, sin, cos, tan}

6.2 GP Wear Model Development

For the development of the prediction models for the low and medium carbon wear, the HeuristicLab framework is used. HeuristicLab is a framework and environment for heuristic optimization that is written in C# language and developed by members of the Heuristic and Evolutionary Algorithms Laboratory (HEAL)[1] [5, 23]. In our experiments, two types of models are developed using HeuristicLab; the GP models and linear regression models. The linear models are employed for comparison as baseline models. The initial parameter values of GP are set as given in Table 1. For training and testing, 70% of the collected data set is randomly selected for training, while the rest 30% of the data set is left for testing.

The convergence curves GP for the low- and medium-carbon wear models are shown in Figs. 12 and 13, respectively. In the case of the low carbon model, GP was able to converge after 110 generations. While in the case of the medium carbon model, GP was converged after 45 generations. The best obtained GP models for both cases are presented in Figs. 8 and 9, respectively. The best GP model for the case of low carbon steel was able to model the wear loss weight with a VAF value of 99.5% and RMSE of 3.23×10^{-4}, while it was capable of predicting for the testing part with a VAF value of 90.6% and RMSE of 0.0015. While for medium carbon steel, the best GP model values were VAF of 94% and RMSE of 0.000734 for training, and VAF of 94.3% and RMSE of 0.000606 for the testing case. The results of GP and linear regression (LR) models for low and medium samples are summarized in Tables 2 and 3, respectively. Actual versus estimated wear predicted quantities for low and medium samples along with absolute errors are shown in Figs. 10 and 11, respectively. Compared to the results of LR model, GP shows superior prediction power. Based on the obtained results, it can be concluded that GP showed good prediction capabilities using compact mathematical models which can be easily evaluated and deployed compared to other approaches adopted in the literature such as neural networks.

[1] http://dev.heuristiclab.com.

$$Wear_{low} = (\mathrm{t} \cdot \mathrm{s} \cdot (c_0 \cdot \mathrm{s} + \sin((c_1 \cdot \mathrm{t} + c_2 \cdot \mathrm{s}))) \cdot (c_3 \cdot \mathrm{s} + c_4 \cdot \mathrm{t} + \tan(c_5 \cdot \mathrm{s}) \cdot c_6 + \tan(c_7) \cdot c_8) \cdot c_9 + c_{10})$$

$$\begin{aligned}
c_0 &= 0.27029 \\
c_1 &= 0.95369 \\
c_2 &= -0.41068 \\
c_3 &= 0.75134 \\
c_4 &= 0.95369 \\
c_5 &= 1.0216 \\
c_6 &= -1.0 \\
c_7 &= 1.4368 \\
c_8 &= 1.2596 \\
c_9 &= 1.8049E-10 \\
c_{10} &= 0.00053583
\end{aligned}$$

Fig. 8 Best generated model for low carbon wear using GP

$$Wear_{medium} = \left(\frac{\mathrm{s} \cdot \mathrm{t} \cdot c_0}{\sin((c_1 \cdot \mathrm{l} + c_2)) \cdot \cos(\cos((c_3 \cdot \mathrm{s} + c_4 \cdot \mathrm{l}))) \cdot c_5} + c_6 \right)$$

$$\begin{aligned}
c_0 &= -8.9815E{-}08 \\
c_1 &= -1.1003 \\
c_2 &= 8.0088 \\
c_3 &= 1.827 \\
c_4 &= -1.1003 \\
c_5 &= 0.2108 \\
c_6 &= -0.0005508
\end{aligned}$$

Fig. 9 Best generated model for medium carbon wear using GP

Table 2 Evaluation results of GP wear model for low carbon

	LR		GP	
	VAF	RMSE	VAF	RMSE
Training	78.569%	0.0022	99.535%	3.236e−04
Testing	82.0416%	0.0018	90.617%	0.0015

Table 3 Evaluation results of GP wear model for medium carbon

	LR		GP	
	VAF	RMSE	VAF	RMSE
Training	68.922%	0.0017	94.045%	0.000734
Testing	47.393%	0.0017	94.279%	0.000606

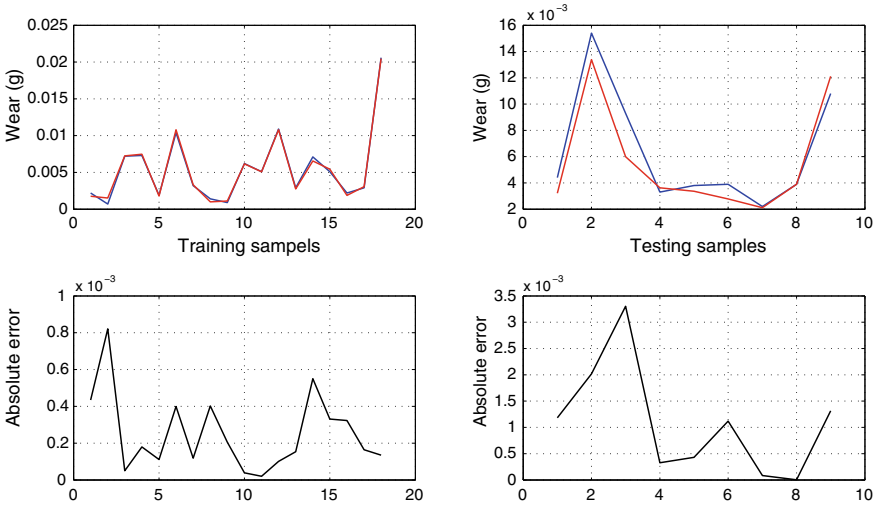


Fig. 10 Actual and GP estimated wearing results for low carbon

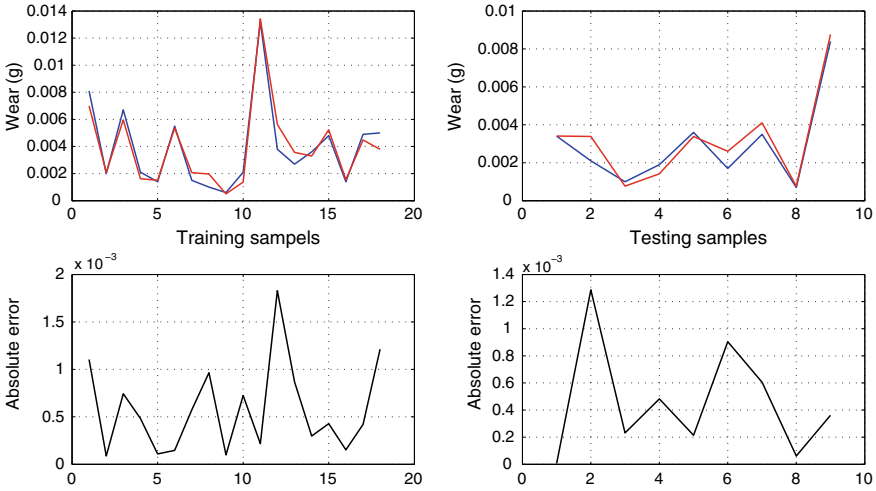


Fig. 11 Actual and GP estimated wearing results for medium carbon

7 Conclusion

In this chapter, a genetic programming-based model was developed and applied to predict the wear loss quantities of low and medium carbon steel. Two samples of different types of carbon steel with a known carbon percentage were investigated (i.e., low and medium). Carbon material was subjected to dry sliding wear experiments

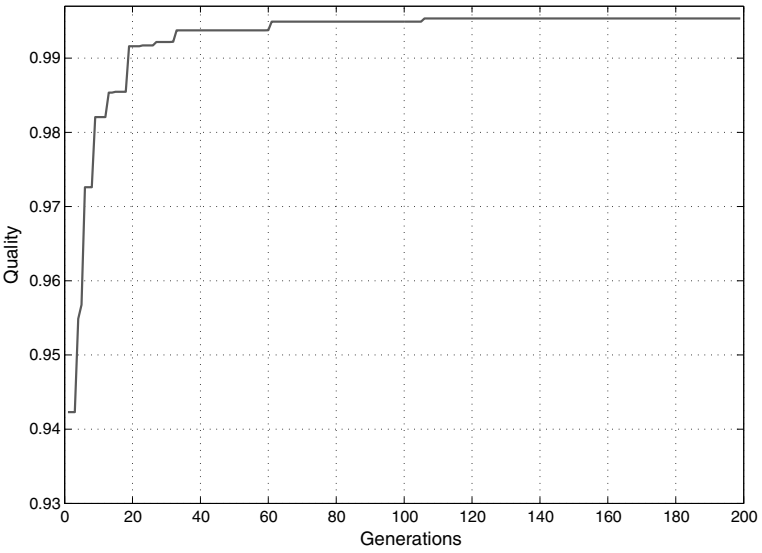


Fig. 12 Best-so-far convergence curve for the low carbon wear GP prediction model

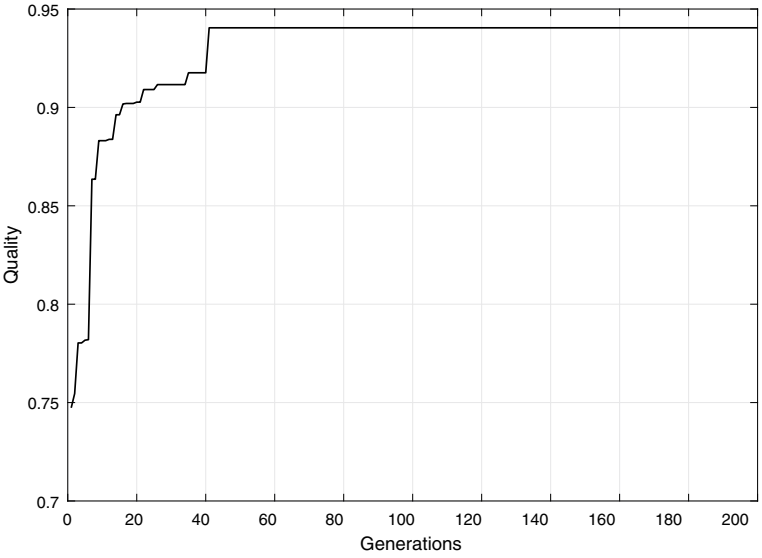


Fig. 13 Best-so-far convergence curve for the medium carbon wear GP prediction model

using a pin-on-disc module based on three attributes; they are load, sliding speed and time. GP showed good performance in prediction results by evolving simple mathematical models which are easy to evaluate.

References

1. Abdelbary A, Abouelwafa MN, El Fahham IM, Hamdy AH (2012) Modeling the wear of polyamide 66 using artificial neural network. Mater Des 41:460–469
2. Adamiak M, Górka J, Kik T (2009) Comparison of abrasion resistance of selected constructional materials. J Achieve Mater Manuf Eng 37(2):375–380
3. Al-Oqla FM, Omar AA (2012) A decision-making model for selecting the gsm mobile phone antenna in the design phase to increase over all performance. Progr Electromagnetics Res C 25:249–269
4. Al-Oqla FM, Salit MA, Ishak MR, Aziz NA (2015) Selecting natural fibers for bio-based materials with conflicting criteria. Am J Appl Sci 12(1):64
5. Beham A, Affenzeller M, Wagner S, Kronberger GK (2008) Simulation optimization with heuristiclab. In: The 20th European modeling and simulation symposium (EMSS2008), pp 75–80
6. Chan KY, Kwong CK, Dillon TS, Tsim YS (2011) Reducing overfitting in manufacturing process modeling using a backward elimination based genetic programming. Appl Soft Comput 11(2):1648–1656
7. Chang L, Friedrich K (2010) Enhancement effect of nanoparticles on the sliding wear of short fiber-reinforced polymer composites: a critical discussion of wear mechanisms. Tribol Int 43(12):2355–2364
8. Durmus HK, Ozkaya E, Dotc CM (2006) The use of neural networks for the prediction of wear loss and surface roughness of AA 6351 aluminium alloy. Mater Des 27(2):156–159
9. Eyre TS (1976) Wear characteristics of metals. Tribol Int 9(5):203–212
10. Faris H, Sheta A (2013) Identification of the tennessee eastman chemical process reactor using genetic programming. Int J Adv Sci Technol 50:121–140
11. Faris H, Sheta A (2016) A comparison between parametric and non-parametric soft computing approaches to model the temperature of a metal cutting tool. Int J Comput Integr Manuf 29(1):64–75
12. Faris H, Sheta A, Öznergiz E (2013) Modelling hot rolling manufacturing process using soft computing techniques. Int J Comput Integr Manuf 26(8):762–771
13. Hakan C, Öztürk H, çelik E, Karlık B (2006) Artificial neural network-based prediction technique for wear loss quantities in mo coatings. Wear 261(10):1064–1068
14. Hsu SM, Shen MC, Ruff AW (1997) Wear prediction for metals. Tribol Int 30(5):377–383
15. Jahan A, Ismail MY, Sapuan SM, Mustapha F (2010) Material screening and choosing methods—a review. Mater Des 31(2):696–705
16. Jakobović D, Jelenković L, Budin L (2007) Genetic programming heuristics for multiple machine scheduling. In: European Conference on Genetic Programming. Springer, Berlin, pp 321–330
17. Kotanchek M, Smits G, Kordon A (2003) Industrial strength genetic programming. In: Riolo RL, Worzel B (eds) Genetic programming theory and practice. Kluwer, New York, pp 239–256
18. Koza JR (1991) Evolving a computer program to generate random numbers using the genetic programming paradigm. In: Proceedings of the Fourth International Conference on Genetic Algorithms. Morgan Kaufmann, La Jolla
19. Koza JR (1992) Genetic programming: on the programming of computers by means of natural selection. The MIT Press, Cambridge

20. Ren Qun, Balazinski Marek, Baron Luc, Jemielniak Krzysztof (2011) Tsk fuzzy modeling for tool wear condition in turning processes: an experimental study. Eng Appl Artif Intell 24(2):260–265
21. Rizal M, Ghani JA, Nuawi MZ, Che Haron CH (2013) Online tool wear prediction system in the turning process using an adaptive neuro-fuzzy inference system. Appl Soft Comput 13(4):1960– 1968
22. Sheta AF, Rausch P, Al-Afeef AS (2012) A monitoring and control framework for lost foam casting manufacturing processes using genetic programming. Int J Bio-Inspired Comput 4(2):111–118
23. Wagner S Affenzeller M (2004) The heuristiclab optimization environment. Technical report, Johannes Kepler University Linz, Austria
24. Wang W (2007) A prognosis model for wear prediction based on oil-based monitoring. J Oper Res Soc 58(7):887–893

Feature Selection

EvoloPy-FS: An Open-Source Nature-Inspired Optimization Framework in Python for Feature Selection

Ruba Abu Khurma, Ibrahim Aljarah, Ahmad Sharieh and Seyedali Mirjalili

Abstract Feature selection is a necessary critical stage in data mining process. There is always an arm race to build frameworks and libraries that ease and automate this process. In this chapter, an EvoloPy-FS framework is proposed, which is a Python open-source optimization framework that includes several well-regarded swarm intelligence (SI) algorithms. It is geared toward feature selection optimization problems. It is an easy to use, reusable, and adaptable framework. The objective of developing EvoloPy-FS is providing a feature selection engine to help researchers even those with less knowledge in SI in solving their problems and visualizing rapid results with a less programming effort. That is why the orientation of this work was to build an open-source, white-box framework, where algorithms and data structures are being explicit, transparent, and publicly available. EvoloPy-FS comes to continue our path for building an integrated optimization environment, which was started by the original EvoloPy for global optimization problems, then EvoloPy-NN for training multilayer perception neural network, and finally the new EvoloPy-FS for feature selection optimization. EvoloPy-FS is freely hosted on (www.evo-ml.com) with a helpful documentation.

Keywords Nature-inspired algorithm (NIA) · Swarm intelligence (SI) · Evolutionary algorithm (EA) · Feature selection (FS) · Transfer function (TF) · Framework · Library · Optimization

R. A. Khurma · I. Aljarah · A. Sharieh
King Abdullah II School for Information Technology, The University of Jordan, Amman, Jordan
e-mail: rubaabukhurma82@gmail.com

I. Aljarah
e-mail: i.aljarah@ju.edu.jo

A. Sharieh
e-mail: sharieh@ju.edu.jo

S. Mirjalili (✉)
Torrens University Australia, Brisbane, QLD 4006, Australia
e-mail: ali.mirjalili@gmail.com

Griffith University, Brisbane, QLD 4111, Australia

S. Mirjalili et al. (eds.), *Evolutionary Machine Learning Techniques*,
Algorithms for Intelligent Systems, https://doi.org/10.1007/978-981-32-9990-0_8

1 Introduction

In this chapter, we present a new Python framework that consists of a set of well-regarded and recent nature-inspired algorithms in their binary representation for handling feature selection (FS) optimization problem. FS is a preprocessing stage applied to a dataset for finding a near-optimal subset of features from the original feature superset [1]. The generated reduced dataset out of the FS stage may yield the same or even better performance compared with the complete set of features [2].

Reducing the size of the dataset by minimizing the number of features is considered a critical step in either supervised or unsupervised learning. In the supervised learning (classification and regression), it means exploring which features have the highest ability to predict the class labels [3–5], while in the unsupervised learning (clustering), FS strives to find the most influential features which can perfectly discriminate samples (observations) into separated clusters [6, 7].

For the training purposes, FS restricts the decision of the data mining or machine learning task to those effective features (relevant features) and excludes the less effective ones (irrelevant or redundant features) by considering them as noise [8]. Logically, this is expected to improve the generalization capability and enhance the quality of the generated model in terms of simplicity and comprehensibility during the training stage. On the other hand, during the testing stage this will be reflected through increased classification performance and reduced error rate [9].

Due to data explosion and increasing the dimensionality, FS becomes a serious requirement in the learning domains. For example, FS is widely applied in the era of pattern recognition applications such as bioinformatics, text mining, and computer vision, where the curse of dimensionality phenomena (Hughes effect) is frequently encountered [10].

FS finds its opportunities in these applications to mitigate the high-dimensionality consequences and relax their significant effect on the classifier job. The other reason why FS becomes paramount is the cost of the classification system, which includes the processing time of the current learning algorithms. According to the computation issues, these algorithms are not always responding in an adequate time; hence, they need to be adapted to scale with high-dimensionality problems without causing a significant performance degradation [11].

There are different approaches to carry out FS including the following three approaches:

1. **Filter-based approach**: (algorithm-independent and data-driven approach): It statistically studies the intrinsic characteristics of the dataset and computes the relevance score for each feature in isolation of the other features (univariate filters) and independently from any predictor /classifier (univariate and multivariate filters) [12].
2. **Wrapper-based approach**: (algorithm-dependent and classifier-driven approach): In contrast to filters, wrappers are algorithm-specific and computation-intensive. They utilize the classifier predictive capability to score each generated subset from the searching process [13].

3. **Embedded approach**: The search process for the optimal subset is implicitly integrated into the classifier construction such as the decision tree classifier [14].

FS is regarded as a complex search problem, where the main target is identifying the most informative and representative features(variables or properties) in the feature space. Formally, FS can be expressed as follows:

Given a superset of features $F = \{\overline{F_1},\overline{F_2},\overline{F_3},\ldots\overline{F_n}\}$, and Y is the class label, the objective of FS problem is to find $S \subset F$ such that S produces the best performance, when predicting Y (minimum error rate and minimum number of features) [15].

This can be done in three main steps [16]:

1. ***Searching***: Traversing the feature space to generate the subsets of features from the original pool of features.
2. ***Feature evaluation***: Measuring the predictive level of each subset of features and how much they are relevant to the class label.
3. ***Stopping criterion***: This is the stop point (termination condition) at which the search process halts.

The search process can be performed using different methods. Conventionally, the brute force method generates all the potential subsets of features which consumes a large search space and results in exponential running time [17]. On the other hand, metaheuristic algorithms are considered a suitable alternative solution for solving complex optimization problems in the time the conventional methods are incapacitate to give a reasonable solution within a polynomial running time. Due to their stochastic and nondeterministic nature, metaheuristic algorithms have been a widespread practice and many academic researchers advocated to use them for solving complex search problems [18].

Nature-inspired algorithm (NIA) is a category of metaheuristic optimization algorithms inspired by the natural phenomena and includes two main subcategories [19]: evolutionary algorithm (EA) and swarm intelligence (SI). EA adopts the evolution and natural selection ideas advocated by Darwin's theory such as genetic algorithm (GA) [20] and genetic programming (GP) [21]. It uses a number of operators to evolve solutions such as crossover, mutation, and elitism.

SI, on the other hand, mimics the natural behavior model of agents where the collective (social) intelligence is sketched out through a swarm system, which composed of abundant, homogeneous, self-organized, and decentralized agents distributed in the environment such as schools of fish, flocks of birds, and colonies of ants [22]. Reaching to the best solution in SI depends on exchanging information between the individuals in the swarm system.

SI is commonly utilized for solving optimization problems with a large search space. Well-regarded algorithms that fall in this category are particle swarm optimization (PSO) [23] and ant colony optimization (ACO) [24]. Recent SI algorithms include cuckoo search (CS) [25], gray wolf optimizer (GWO) [26], multi-verse optimizer (MVO) [27], moth-flame optimization (MFO) [28], whale optimization algorithm (WOA) [29], bat algorithm (BAT) [30], firefly algorithm (FFA) [31], and many others [32–37].

According to no free lunch (NFL) theorem [38], no algorithm is the best for all optimization problems. An algorithm which has the superior performance for some optimization problems may degrade its performance when used with other types of optimization problems. This theorem motivated the researchers to develop novel algorithms and investigate existing ones when adopted to solve various challenging optimization problems in different applications.

The motivation to develop EvoloPy-FS is to achieve the following prime advantages:

1. Utilizing, jointly, the power of Python and SI algorithms to solve FS problem in several applications.
2. Extending the range of users by empowering inexperienced users from different domains to employ this framework in their applications.
3. Relieving the technical work overhead so that the researchers are enabled to focus more on analyzing the generated learning models instead of losing the time starting programming from scratch.
4. Optimizing the time of configuring and testing the experiments.
5. Practically speaking, using a fast programming language shipped with libraries can help making rapid prototyping and so relieving the implementation matters.
6. Achieving the adaptability by customizing algorithms and amending parameters to tackle a new family of problems.
7. Providing separability which means separating the problem to be solved from the benchmark algorithms, meanwhile allowing a user to play and plug different datasets into the framework in order to assess the performed data mining task.
8. Avoiding the obscurity by assembling the different components of the SI system within a white-box framework which is provided as an open-source code.
9. Offering a ready-to-use test suite for researchers to validate their new methodologies and comparing them with the published benchmark algorithms.
10. Increasing the reliability and lessening the errors by providing a quite generic tool which is not tailored to any application and leave the researchers free to make their instantiations and specifications.

This chapter is organized as follows: Sect. 2 describes the related works. Section 3 explains the Python properties. Section 4 shows EvoloPy versions. Section 5 presents an overview for the framework. Section 6 discusses some related design issues, Sect. 7 compares wrapper and filter–wrapper and filter approaches, and finally Sect. 8 concludes the chapter and shows the future directions.

2 Related Works

This section brings to the light a summary of the main accomplishments in the literature which are related to metaheuristic frameworks and FS software developments. The details of these works are presented in Tables 1 and 2, respectively.

Table 1 Evolutionary computation frameworks, tools, and libraries

Library	PL	Year	Optimization algorithm(s)	Description
GAlib [39]	C++	1996	GA	GAlib was developed over UNIX operating system . The library was utilized in parallel systems and employed within a distributed computing. It is built on platforms and is implemented to support distributed/parallel environments
Evolving object (EO) centering [40]	C++	2001	Several ECs paradigms including: GA and PSO	1. Was utilized to solve several combinatorial problems: binary strings, permutations, vectors
				2. Implemented different evolutionary selection operators such as tournaments and roulette
				3. Several operators: uniform initialization, Gaussian mutation, sub-tree crossover and different combination strategies including: proportional combination and sequential call
				4. Parallelization tools: OpenMP and openMPI
				5. There were many EO versions which were developed and implemented over different operating systems including Windows and UNIX
TEA centering [41]	C++	2001	GAs and evolutionary programming	Supports complex genotypes and nonstandard representations with mixed chromosome types and parallel population structures

(continued)

Table 1 (continued)

Library	PL	Year	Optimization algorithm(s)	Description
OpenTS [42]	Java	2001	Tabu search algorithm	1. It can work with any problems such as vehicle routing problems and assignment problems
				2. Additional features are presented including the ability to run over different modern computers with multiprocessing systems
				3. Take the advantage of Java features such as cross-platform and object-oriented programming
				4. Ability to be delivered on the Internet
PISA [43]	PL independence	2003	Most EAs and simulated annealing	PISA was basically used to solve multi-objective problems for optimizing more than one conflicting criterion. Goals
ParadisEO [44]	C++	2004	Different EAs	1. A white-box object-oriented framework
				2. Integrates many distributed metaheuristics platforms
				3. Includes the local searches (LS) and hybridization mechanisms
HeuristicLab [45]	C#	2005	EAs algorithms including: PSO, GA, and GP	A generic optimization environment
JavaEvA [46]	Java	2005	GA	Two main parts: EvAClient (GUI) and EvAClient (optimization kernel)

(continued)

Table 1 (continued)

Library	PL	Year	Optimization algorithm(s)	Description
UOF [47]	C++	2006	LM, GA, PSO, SA, etc.	It basically separates the problem and the solver and makes these two parts independent but supports a bridging between them. This made the library more general and able to tackle other optimization problems
GEATbx [48]	MATLAB	2007	Many GA and GP variants	Provides many global optimization capabilities
CIlib [49]	Java	2008	SIs, ECs, and NN	1. Very generic, plug, and simulate
				2. Uses XML files
				3. Uses simple Java class
				4. The code was open source which allowed for the continuous update for the code and made the error discovery and handling to be done in short time
JCLEC [50]	Java	2008	EAs: GA, GP	1. It applied many of the object-oriented programming concepts such as abstractions and reusability (reuse code)
				2. Open source which indicates it is cheaper for sharing and modifications
				3.Portable among any platform with Java virtual machine(JVM)
Pyevolve [51]	Python	2009	GA, next versions support GP	1. It supported multiple operating systems, and it could be run on cell phones
				2. It included several evolutionary operators including the selection operators
EvA2 [52]	Java	2010	Population-based, such as: GAs, DE, PSO, and SA	1. Heuristic operators are easily interchanged between different optimization modules and easily adapted to any optimization problem
				2. Support multimodal and multi-objective optimization methods
				3. It supported object-oriented Java architecture which could be implemented over on the client–server paradigms

(continued)

Table 1 (continued)

Library	PL	Year	Optimization algorithm(s)	Description
JMetal [53]	Java	2011	Many metaheuristic algorithms	Employs the object-oriented architecture, multi-objective algorithms, and parallel algorithms
Drools Planner [54]	Java	2011	Set of metaheuristic algorithms	1. Uses metaheuristic algorithms to tackle planning problems
				2. Uses simple mechanism for writing constraints in a clear manner
Opt4J [55]	Java	2011	DE, PSO, and SA	1. It provides SPEA2 and NSGA2 multi-objective algorithms
				2. It supports the knapsack problem
				3. Module-based implementation; offers GUI to configure and visualize the optimization process
DEAP [56]	Python	2012	GA, GP, ES, PSO, DE	It supports parallelism and supply benchmarks which contain different test functions that could be utilized for evaluation
PYGMO and PYKEP [57]	C++ Python	2012	Adaptive version of differential evolution (jDE)	Support scripting for massively parallel optimization of aerospace-related problems (interplanetary trajectory optimization)
ECJ [58]	Java, C++	2017	GA and GP	Widely used EC library with particular strengths in GP, massive distributed computation, and coevolution
PlatEMO [59]	MATLAB	2017	Multi-objective evolutionary algorithms	Open-source tool supported by graphical user interface. It includes many multi-objective EAs and test problems

Table 2 Feature selection frameworks, tools, and libraries

Library	PL	Year	Description
MLC++ [60]	C++	1994	A library of C++ classes and tools for supervised learning. It provides different implementations for feature selection methods including a wrapping approach with different methods of searching and selection
Weka [61]	Java	1999	The process of feature selection depends on implementing a search method plus an evaluation step such as ranker search which was applied together with CorrelationAttributeEval technique. The following are the feature selection methods supported by Weka: 1. Correlation FS 2. Information gain FS 3. Learner FS
PyMVPA [62]	Python	2009	It is a free software package which was designed to ease the statistical and the analytical task for the large datasets and support an interface for a variety of algorithms including classification, regression, and feature selection
Infosel++ [63]	C++	2010	1. It is a large package in C++ library for feature selection and ranking 2. It includes a collection of classes for feature selection, and it aids the users for determining the suitable algorithm to be applied for solving a given problem 3. FS algorithms in the Infosel++ can be categorized into four sets: ranking methods, ranking with shifting of highly correlated features, ranking with removal of highly correlated features, and other methods such as Markov blanket approximation
PyBrain [64]	Python	2010	Bybrain supports feature Gaussian processes, the evolving algorithm and SVM wrapper
KEEL [65]	Java	2011	KEEL tool has a dedicated family of algorithms for feature selection including: MIFS, LVF, Focus, Relief, Las Vegas, LVW, ABB–IEP, ABB–LIU, ABB–MI, Full–IEP, Full–LIU, Full–MI, Relief–F, LVF–IEP, SA–IEP, SA–LIU, SA–MI, SBS–IEP, SBS–LIU–, SBS–MI–, SFS–IEP, SFS–LIU, and SFS–MI
Scikit-learn [66]	Python	2011	1. It is a tool that could be utilized in the fields of data mining and machine learning. Furthermore, the code is open-source which helps and eases the researchers job 2. Built on NumPy, SciPy, and Matplotlib 3. Includes modules such as sklearn library which can be used for FS
Mlpy [67]	Python	2012	1. mlpy is a Python library which constructed using other libraries such as NumPy, SciPy, and GNU. In addition, it provided open-source code implementation 2. mlpy was used in the multiple applications including FDA, SRDA, and PCA.
mlpack [68]	C++	2013	It is a fast, flexible library which supports C++ libraries. Its modules can be integrated into a larger-scale machine learning solutions. The algorithms are implemented as a simple command line programs and support PCA for feature reduction
Orange [69]	C++	2013	It is a structured library used as a data mining tool. The lower-level procedures include data filtering, probability assessment, and feature scoring. These are assembled to serve the upper levels in the hierarchy such as classification. The components of the lower levels (classes and data structures) are coded using C++ where the components of the higher levels are written in Python

(continued)

Table 2 (continued)

Library	PL	Year	Description
FeatureIDE [70]	C++	2014	It is a framework which used open-source code and utilized clips for developing systems
DWFS [71]	MPI	2015	1. A Web-based tool for feature selection that is tailored to solve several types of problems
			2. DWFS is wrapper-based with GA implemented as a search strategy
			3. A large feature space can be examined using parallel GA
			4. GA parameters are tuned based on the application
			5. Different filter approaches can be utilized
FSLib [72]	MATLAB	2016	It is a MATLAB FS library which includes many algorithms such as CFS, DGUFS, ECFS, Fisher, FSASL, FSV, ILFS, LASSO, LLCFS, LS, MCFS, MI, Relief–FRFE, UDFS, UFSOL, L0, Inf–FS, and mRMR

In [39], the authors presented GAlib library which applied GA for parallel environment. Another work in [40] was called evolving objects (EOs). In the same year, two other libraries were developed, namely TEA centering [41], which supported complex genotypes, and OpenTS [42], which implemented tabu search algorithm. PISA library was presented as a language-independent library which included most EAs and simulated annealing [43]. Later, ParadisEO was developed [44]. C# HeuristicLab library [45] and JavaEvA [46] were developed in the next year. UOF [47] was developed as a generic framework to solve new optimization problems. A GEATbx [48] was developed for global optimization using MATLAB. CIlib [49] provided Java open-source structure and easily refined code. JCLEC [50] followed the principal of object-oriented. A Python framework, namely Pyevolve, was presented in [51]. Later on, several evolutionary Java frameworks were developed, namely EvA2 [52], jMetal [53], Drools Planner [54], and Opt4J [55]. DEAP Python framework was developed in [56] for applying optimization with parallel features. PYGMO and PYKEP frameworks [57] with C++ and Python programming languages were used for massively parallel optimization of aerospace-related problems. An EC library, namely ECJ, was developed in [58] with a particular strengths in GP. Recently, a MATLAB library called PlatEMO [59] was developed as multi-objective evolutionary open-source tool.

All the aforementioned works summarized the main EC frameworks in the literature. On the other hand, several specialized FS frameworks were developed separately. Examples on these FS frameworks were MLC++ [60], Weka, Java tool [61], PyMVPA [62], Infosel++ library [63], PyBrain, Python FS [64]. A useful tool, namely KEEL, was developed in Java with dedicated family of algorithms for FS [65]. scikit-learn [66] is a Python tool, which contains many FS methods, which was built on NumPy, SciPy, and Matplotlib. One year later, another Python library called mlpy [67] was developed and used in different applications. Several C++ FS frameworks were developed successively, namely mlpack [68], Orange [69], and FeatureIDE [70]. DWFS [71] was used as a FS Web-based tool tailored to solve several types of problems. A recent MATLAB FS library called FSLib was presented in [72].

3 Why Python

Python is a high-level programming interpreted language which was designed by Rossum in 1990 [72]. It was built based on scripts to achieve multiple general targets. Python has a considerable community since its characteristics and properties have attracted serious followers and researchers. The following points are some of these properties.

1. Extensibility means the possibility of extending the legacy code by adding new functionalities and gluing multiple components.
2. It is popularly used in the context of data analysis.
3. Elegant syntax (pseudo-code like syntax).
4. Promotes code readability and maintainability (intensive work with less code).
5. Availability of substantial number of libraries.
6. Supports modularity.
7. Platform independent(cross-platform) so it is able to span multiple platforms such as Linux and different versions of Windows operating system.
8. Availability of data structures such as lists, arrays, and dictionaries.
9. Dynamic typing and dynamic binding.
10. Overhead and high cost in the execution of the interpreted program.

The decision for choosing a Python depends on our concerns. If we are interested in speed of development and postponed coding, then Python is a good choice, while it is not recommended when the main concern is reducing the cost of code execution [73].

4 EvoloPy Versions

This framework is built upon our previous EvoloPy version [74]. The aim of the original EvoloPy was providing a Python framework which is open source , supports multiple platforms, and implements a considerable number of well-regarded and state-of-the-art metaheuristic algorithms. It was utilized by researchers for optimizing their problems and helped them to design new algorithms and improve current ones. Python was used for implementing the optimizers and increasing their robustness and portability. EvoloPy source code was published on GitHub - https://github.com/7ossam81/EvoloPy .

The framework was designed based on four basic components: the optimizer component, which is the main interface where the user can select the optimization algorithms to run and adjust the parameters such as the number of runs, population size, and maximum iterations. In the same script, the user can select which benchmark function to run. The second component in EvoloPy framework is the metaheuristic algorithms. It included eight recent nature-inspired metaheuristic optimizers, namely particle swarm optimization (PSO), firefly algorithm (FFA), gray wolf optimizer

(GWO), whale optimization algorithm (WOA), multi-verse optimizer (MVO), moth-flame optimization (MFO) algorithm, bat algorithm (BAT), and cuckoo search (CS). The third component is the benchmark functions, which included several common benchmark problems. The last component is result management, which was used to export the results and store them in CSV files.

Experiments were designed to compare the performance of metaheuristic algorithms implemented in Python and their peers implemented in MATLAB in terms of the running time. The other dimension that was investigated in the experiments was the influence of the size of the problem (dimension size) on running time. It was appeared from the results that there was MATLAB and Python does not significantly differ when using small dimension size. On the other hand, this difference was remarkably increased when the size of dimensionality was increased.

The other variation of EvoloPy was EvoloPy-NN framework which provided classical and recent nature-inspired metaheuristic for training a single-layer multilayer perceptron neural network (NN) [75]. The list of implemented optimizers that were used to train NN was PSO, MVO, GWO, and MFO [76, 77]. The main file in EvoloPy-NN is main.py, which serves as an interface for the framework where a user can set up his/her experiment by selecting optimizers, datasets, number of runs, maximum number of iterations, number of neurons, and population size. The source code is freely available on https://github.com/7ossam81/EvoloPy-NN.

5 Framework Overview

Starting from our objective, building an integrated environment that provides various optimization capabilities, we will continue on the same path by developing a new EvoloPy-FS for tackling different FS-related aspects. EvoloPy-FS interface consists of five components, which are optimizer, swarm intelligence algorithms, fitness functions, transfer functions (TFs), and results. Each one of these components is discussed in a special subsection.

5.1 The Optimizer

This is the main script, where a user can configure the experiments and adjust initial settings. For example, a user can select the optimization algorithm, specify the fitness function to be applied, and set up the parameters including number of runs, size of population, and number of iterations.

5.2 SIs Algorithms

Each implemented SI algorithm has a separated script. A set of SI algorithms that have been added to the framework until now are:

1. Binary particle swarm optimization (BPSO)
2. Binary firefly algorithm (BFFA)
3. Binary gray wolf optimizer (BGWO)
4. Binary whale optimization algorithm (BWOA)
5. Binary multi-verse optimizer (BMVO)
6. Binary moth-flame optimizer (BMFO)
7. Binary bat algorithm (BBAT)
8. Binary cuckoo search algorithm (BCS).

5.3 Fitness Functions

The framework consists of a set of user-defined fitness functions. These functions stand for doing a set of issues; for example, some fitness functions are implemented in different ways for loading the data. Other functions focus on applying different evaluation criteria. Moreover, some fitness functions were used to show the possibility of applying different training and testing methodologies. The most important fitness functions are those implement wrapper and filter–wrapper approaches (F10 and F3), respectively, as will be discussed next.

5.3.1 Data Loading Techniques

It shows the possibility of applying different techniques for loading the data using Python:

1. Using standard Python libraries: Python application programming interface (API) provides modules and functions that can be used to load data. For example, we used:
   $from\ sklearn.datasets\ import\ load\_iris$
   $iris = load\_iris()$
   $x = iris.data$
   $y = iris.target$
   $n\_samples, n\_features = iris.data.shape$
2. Load CSV file with pandas from URL and using read_csv() function:
   $from\ pandas\ import\ read\_csv$
   url="https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv".
   $names = ['preg','plas','pres','skin','test','mass','pedi','age','class']$.

$data\,frame = read_csv(url, names = names)$
$array = data\,frame.values.$
$A = array[:, 0 : 8]$
$B = array[:, 8]$
$no\_featuresd = len(names) - 1$

3. Load CSV file with pandas from the current directory and using read_csv() function:
   import pandas as pd
   data_set:pd.read_csv("C:/Users/tc/Desktop/breast–
   cancer–wisconsin.csv")
   data_set.head()
   data=data_set.iloc[:,0:10].values
   target=data_set.iloc[:, -1].values

5.3.2 Evaluation Measurements

There are a set of fitness functions that are applied to return different evaluation measurements:

1. Accuracy:

   It can be computed using Python in two ways:

   - Computed directly:

   $$acc = float((y_test == y_pred).sum())/y_pred.shape[0]$$

   - Importing accuracy_score from sklearn.matrices:
     $from\ sklearn.metrics\ import\ accuracy\_score.$
     $acc = float(accuracy\_score(y_test, y_pred)).$

2. Area under the curve (AUC):
   AUC=$roc\_auc\_score(B_test, B_pred)$.
3. Confusion matrix:
   $from\ sklearn\ import\ metrics.\ metrics.confusion\_matrix(y_test, y_pred)$.
4. Sensitivity, specificity, and Gmean:
   TP=0 %true positive
   FP=0 %false positive
   TN=0 %true negative
   FN=0 %false positive

   $for\ i\ in\ range(len(B_pred))$:

   $if\ B_test[i] == 1\ and\ B_pred[i] == 1$:
   TP+= 1

$if B_pred[i] == 1 \ and \ B_test[i] != B_pred[i]:$
$FP += 1$
$if B_test[i] == 0 \ and \ B_pred[i] == 0:$
$TN += 1$
$if B_pred[i] == 0 \ and \ B_test[i] != B_pred[i]:$
$FN += 1$

$TPR = TP/np.sum(B_test) \ \%Recall/Sensitivity$
$TNR = TN/np.sum(B_test) \ \%Specificity$
$Gmean = np.sqrt(TPR * TNR)$

5.3.3 Training and Testing Methodologies

The fitness function script shows different training and testing methodologies:

1. Simple split:
   $from \ sklearn.model\_selection \ import train\_test\_split.$
   $x_train, x_test, y_train, y_test = train\_test\_split(x, y, test\_size = 0.33, random\_state = 1)$
2. Cross validation:
   $scores = cross\_val\_score(knn, reduced\_x\_dataset, y, cv = 10, scoring = 'accuracy').$
   $average\_scores = np.mean(scores).$
3. Checking for overfitting:
   $metrics.f1\_score(y_test, y_pred, average = ''macro'') \quad metrics.f1\_score(y_train, knn.predict(reduced\_x_train), average = '' macro'').$

5.3.4 Wrapper Approach

The fitness functions also implement a wrapper approach in different ways:

1. By doing the binary conversion for the received real chromosome inside the object function in two ways:
   - Using binarizer:
     $from \ sklearn.preprocessing \ import \ Binarizer \ I = np.reshape(I, (1, -1))$
     $binarizer = Binarizer().fit(I)$
     $t = Binarizer().fit\_transform(I)$
   - Using threshold:
     $threshold = .2$
     $for \ i \ in \ range(0, len(I)): \ if I[i] \geq threshold:$
     $I[i] = 1$
     $else:$
     $I[i] = 0$

2. Build binary optimizers so that the generated individuals are binary. In this case, there is no need for all Python statements in point 1 above since the received individual by the object function is binary. For example, the initialization statement in BPSO which defines a set of binary individuals is:
   pos=$numpy.random.randint(2, size$=$(PopSize, dim))$.
3. After performing binary conversion for individuals, there is a need to make sure that the dimension of the individual equals the number of features in the dataset by configuring the dimension in $getFunctionDetails$.The individual and the population are represented by 1-D and 2-D arrays, respectively, using the *NumPy* library.
4. Generate the reduced dataset in such a way the value 1 in the individual corresponds to a selected feature and the value 0 indicates that the feature is not selected.
   $reduced\ features = [\ ]$
   $for\ index\ in\ range(0, n\_features)$:
   $if(I[index] == 1)$
   $reduced\ features.append(index)$
   $reduced\_x_train$=$x_train[:, reduced\ features]$ $reduced\_x_test = x_test[:, reduced\ features]$
5. Call a rapid classifier such as Knn to evaluate the candidate solution and generate the fitness value based on the reduced dataset.
   $knn = KNeighborsClassifier(n\_neighbors$=4)
   $knn.fit(reduced\_x_train, y_train)$
   $y_pred = knn.predict(reduced\_x_test)$
6. These steps are repeated until the maximum number of iterations is reached.

5.3.5 Filter Approach

Fitness functions implement different filter approaches, and a user can also apply a hybrid filter–wrapper approach. The filter is first applied to do a rapid feature reduction based on some sorts of statistics without any learning mechanism (filtering stage), and then a second stage of feature reduction is applied to the reduced dataset generated by a filter (wrapper stage). This approach can be applied with a large size datasets. These filters were downloaded from a scikit-learn FS Python library hosted on http://featureselection.asu.edu/index.php with a little bit modification on ensemble filter.

The applied filtering techniques in the framework are:

1. chi-square filter.
   $from\ sklearn.feature\_selection\ import\ SelectKBest$
   $from\ sklearn.feature\_selection\ import\ chi2$
   $x\_after\_filter1$=$SelectKBest(chi2, k$=2$).fit_transform(x, y)$.

2. Variance filter.
   $from\ sklearn.feature\_selection\ import VarianceThreshold.$
   $selector = VarianceThreshold(.2)$
   $x\_after\_filter2 = selector.fit\_transform(x)$
3. Ensemble-based filter:
   New filter is proposed in this framework. The filter works by generating the importance of all features using the *ExtraTreeClassifier*, and then the features with importance greater than the computed average importance are selected.
   $reduced\_feature\_list = [\ ]$
   $from\ sklearn.ensemble\ import\ ExtraTreesClassifier$
   $model=ExtraTreesClassifier().$
   $model.fit(x, y).$
   $feature\_importances\_list=np.array(model.feature\_importances\_).\ \ average\_feature\_importance=np.mean(feature\_importances\_list).\ \ for\ i\ in\ range(0, len(model.feature\_importances\_)) :$
   $if\ model.feature\_importances\_[i] \geq average\_feature\_importance:reduced\_feature\_list.append(i)$

5.4 Transfer Functions (TFs)

These include V-shaped TFs and S-shaped TFs. TFs are applied as a mechanism for binary conversion of the individuals after applying the updated operators. Table 3 shows the Python implementation for these functions. The libraries necessary to implement these functions are *scipy*, *math*, *numpy*, *skfuzzy*, and *Matplotlib* is necessary for plotting the functions. Figures 1 and 2 depict these functions.

5.5 Result Component

All results are stored in CSV file for result exportation purposes. Figure 3 shows the main components and relations in the EvoloPy-FS framework. It represents the framework using twelve classes. The optimizer, benchmarks, and solution are used to

Table 3 Transfer functions

V-shaped TFs	S-shaped TFs
$v1 = abs(erf((pi/2) * x))$	$s1 = 1/(1 + np.exp(-2 * x))$
$v2 = abs(np.tanh(x))$	$s2 = 1/(1 + np.exp(-x))$
$v3 = abs(x/np.sqrt(1 + np.square(x)))$	$s3 = 1/(1 + np.exp(-x/3))$
$v4 = abs((2/pi) * np.arctan((pi/2) * x))$	$s4 = 1/(1 + np.exp(-x/2))$

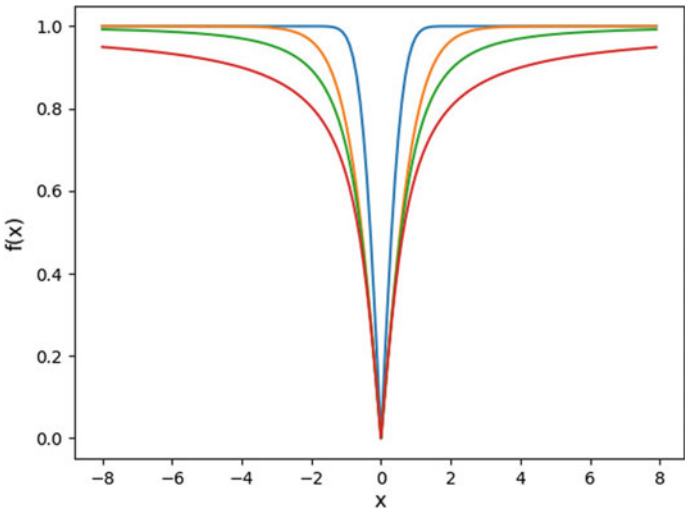


Fig. 1 V-shaped TFs

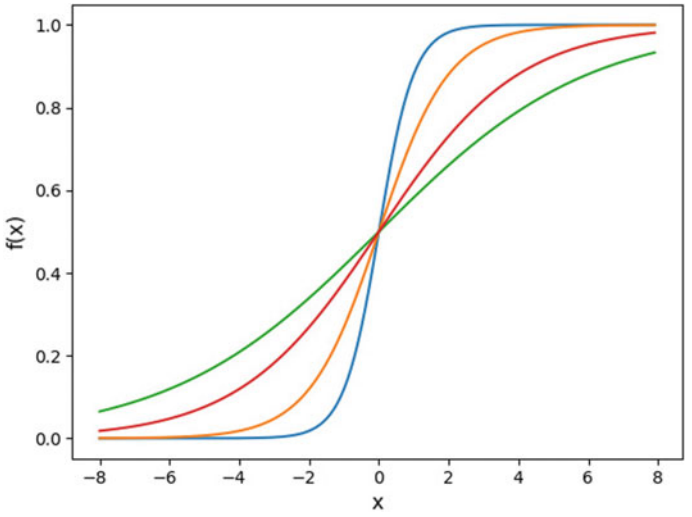


Fig. 2 S-shaped TFs

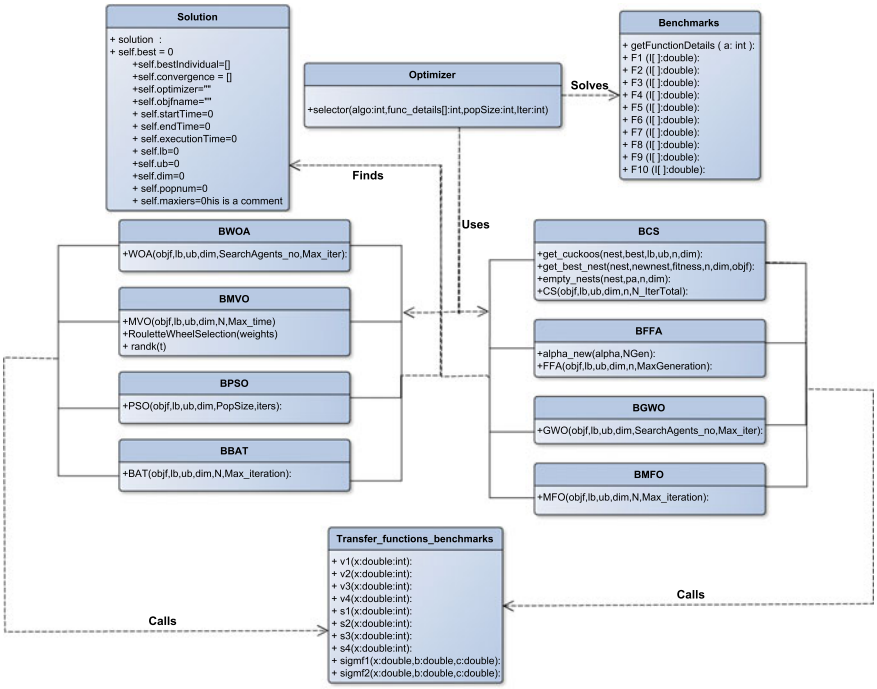


Fig. 3 EvoloPy-FS framework

manage the optimization process. The TF benchmark is used for binary conversion. The other eight classes are corresponding to eight SI algorithms; each one performs its optimization job independently.

6 Design Issues

NIAs in general are classified under the population-based metaheuristic umbrella. In contrast to single-solution local search category, the population-based algorithms adopt multi-solution global search strategy. Population-based approach starts by initializing a set of possible (candidates) solutions that satisfy the constraints of the problem under investigation. Normally, the population is represented by 2-D array structure and each candidate solution is represented by 1-D array structure (vector). With regard to FS problem, the length of the chromosome (individual and solution) represents the dimensionality of the problem (number of features or problem size). Each gene value in the chromosome represents the feature (either selected or not selected). The gene value is assigned to 1 when the feature was selected or 0 when the feature was not selected. This representation scheme for the individuals is called binary encoding, where each gene is assigned to either 1 or 0.

7 Experiments and Discussion

The experiments of previous version of EvoloPy [74] focused on comparing the Python and MATLAB and showed the effect of the programming code type on the optimization problem. The results were positives in terms of running time and demonstrated the power and efficiency of EvoloPy tool for solving high-dimensionality problems.

In this version of EvoloPy, the target is to focus more on FS problem and utilizing EvoloPy-FS for tackling different aspects related to FS problem. For achieving this purpose, we started by investigating the behavior of applying different wrapper approaches on different datasets with different characteristics. These approaches were compared in terms of three evaluation measurements: error rate, number of features, and the running time. The effect of integrating the filter approach with wrapper approach was demonstrated by selected one popular filtering method called chi-square from a scikit-learn FS Python library hosted on http://featureselection.asu.edu/index.php. Chi-square filter is well known for its robustness with respect to the data distribution and the simplicity in computations.

The experiments were performed on a personal machine with AMD Athlon Dual-Core QL-60 CPU at 1.90 GHz and a memory of 2 GB running Windows 7 Ultimate 64-bit operating system. In the conducted experiments, the population size, the number of iterations, and the number of runs were set to 10, 100, and 20, respectively, for the eight tested algorithms. KNN classifier with Euclidean distance measure ($K = 5$ [78]) has been used for classification evaluation in the experiments. A simple split training and testing methodology was applied in the experiments with 33% of instances in every case were reserved for testing and the rest of them were employed for training.

The experiments were based on applying a single objective error rate fitness function as appears in Eq. 1. This was the same as the fitness function implemented in [79]. The reason behind that is that our attention is minimizing the classification error rate and discovering the most accurate wrapper with the minimal average classification error over all datasets. Another reason is that most of the existing studies adopt the classification performance as a fitness function in their works [80]. A third reason is that the experiments are not tailored to a certain application; instead, we have worked on common benchmark datasets. So that, we could not determine a priori the application-specific parameters that balance the classification error and the number of features as used in other works. This adopts a fitness function that simultaneously increases the classification performance and minimizes the number of selected features [81, 82].

$$Fitness = ErrorRate \tag{1}$$

The experiments were performed on three stages as follows.

1. *Wrapper-based experiments (without filters)*:
   In this stage, the behavior of the 8 optimization algorithms (BPSO, BMVO, BMFO, BGWO, BWOA, BFFA, BBAT, and BCS) was investigated on **30** different well-regarded datasets with different specifications. The datasets were downloaded from different sources and repositories such as https://archive.ics.uci.edu, https://www.kaggle.com/datasets, and http://featureselection.asu.edu/index.php. Tables 4 and 5 show the details of these datasets. Three evaluation measurements were used for assessing the performance of each optimizer and its capability in performing FS. These evaluation metrics are error rate, number of selected features, and the running time. Tables 6, 7 and 8 show the results of this stage of experiments.
2. *Hybrid filter–wrapper-based experiments* (filters integrated with wrappers):
   The chi-square filter was integrated with the previous eight optimizers (wrappers). For the chi-square filter, there is one significant parameter called K which represents the predetermined number of features to which the filter should make the reduction (number of selected features by chi-square filter). For example, if the dataset consists of 7000 features and K assigned to 1000 features, then the new size of the reduced dataset after applying chi-square filter is 1000 features. The question that can be asked in this context is what is the best value that can be assigned to K? and how will it affect the wrapper results when the chi-square filter integrated with wrappers? To show the influence of the value of the K parameter on the different datasets, 3 different FS ratios (FSR) were experimented 25, 35, and 75%. For example, when $FSR = 25\%$ and the original dataset dimensionality is 325 features, then $K = 0.25 * 325 = 81$ features which is the size of the reduced dataset after applying the chi-square. In this stage of experiments, we have selected a sample that consists of six datasets with different specifications regarding the number of instances and the dimensionality: *Breastcancer*, *penglung*, $Wave\,fromEW$, $KrvskpEW$, $orlraws10P$, and CLL_SUB_111. Tables 9, 10, and 11 show the results of this stage of experiments.
3. *Filter-based experiments (without wrappers)*:
   In these experiments, the chi-square filter was applied separately without wrappers on the previously selected datasets and using also the three *FSRs*. Table 12 shows the results of this stage of experiments.

The results of first-stage experiments showed that BMFO registered the minimum error rate over 11 datasets and the minimum standard deviation over 13 datasets, and then BMVO achieved the minimum error rate over 6 datasets. BPSO was the highest stability in error rate results by achieving the minimum error standard deviation over 15 datasets. On the other hand, with respect to the number of features, BMVO was the best optimizer in feature reduction over 16 datasets, and then BPSO was the best over 9 datasets. BPSO achieved the least standard deviation with respect to the number of selected features over 17 datasets, and then BMFO was the least over 10 datasets. From Table 7, we can compute the *FSR* ratio for each dataset. *FSR* indicates the

Table 4 List of the used 30 datasets, sorted based on the number of features

No.	Dataset	No. of features	No. of instances
1	Vertebral	6	310
2	liver	6	345
3	diabetes	8	769
4	Breastcancer	9	699
5	Tic-tac-toc	9	958
6	WineEW	13	178
7	HeartEW	13	270
8	Exactly	13	1000
9	Exactly2	13	1000
10	M-of-n	13	1000
11	Zoo	16	101
12	Vote	16	300
13	CongressEW	16	435
14	Lymphography	18	148
15	parkinsons2	22	196
16	SpectEW	22	267
17	BreastEW	30	569
18	Ionosphere	34	351
19	KrvskpEW	36	3196
20	WavefromEW	40	5000
21	SonarEW	60	208
22	clean1	166	476
23	semeion	265	1593
24	penglung	325	73
25	lung_discrete	325	73
26	Colon	2000	62
27	GLIOMA	4434	40
28	Leukemia	7129	72
29	orlraws10P	10, 304	100
30	CLL_SUB_111	11, 340	111

number of selected features to the number original dataset features (reduced dataset size: original dataset size). Table 5 illustrates *FSRs* for all datasets.

From Table 5, we can notice that the range for *FSRs* was between [36% and 69%] and that *FSR* was small for large dimensionality datasets and it became larger for small dimensionality dataset. The average *FSR* for all datasets was 48.5% which indicates that the size of the reduced dataset is almost equal **half** of the size of the original dataset.

Table 5 FS ratio summary of 30 selected datasets

Breastcancer	BreastEW	clean1	Colon	CongressEW
4:9 = 44%	14:30 = 47%	74:166 = 45%	740:2000 = 37%	7:16 = 44%
diabetes	Exactly	Exactly2	HeartEW	Ionosphere
4:8 = 50%	6.5:13 = 50%	6.1:13 = 47%	9:13 = 69%	13:34 = 38%
KrvskpEW	Leukemia	liver	Lymphography	M-of-n
18.7:36 = 52%	2728:7129 = 38%	4:6 = 67%	8.8:18 = 49%	7.5:13 = 58%
parkinsons2	penglung	semeion	SonarEW	SpectEW
9.8:22 = 45%	149:325 = 46%	124:265 = 47%	27:60 = 45%	9.6:22 = 44%
Tic-tac-toc	Vertebral	Vote	WavefromEW	WineEW
5:9 = 56%	2.7:6 = 45%	6.5:16 = 41%	21:40 = 53%	6.5:13 = 50%
Zoo	orlraws10P	CLL_SUB_111	GLIOMA	lung_discrete
8.4:16 = 53%	3698:10304 = 36%	4303:11340 = 38%	1700:4434 = 38%	142:325 = 44%

Finally, with respect to the running time measurement, the BBAT optimizer was the fastest over 12 datasets, then BPSO was the fastest over 9 datasets, and BFFA also was the fastest over 8 datasets. BPSO was the most stable optimizer with respect to the running time by achieving the least standard deviation in time over 12 datasets.

The results of the second-stage experiments were as follows.

- *Breastcancer* results: The error rate for all wrappers was increased when the chi-square with *FSR* assigned to either 25% or 35% was integrated with them. In contrast, the error rate was decreased when the *FSR* was assigned to 75%. So, the integration of chi-square with wrappers was better than applying the wrappers alone only when the *FSR* was 75%.
- *penglung* results: The error rate increased when chi-square was integrated with all optimizers regardless of *FSR*. So, best error rate results were achieved when wrappers applied alone without chi-square.
- $WavefromEW$ results: For almost all optimizers, the error rate increased when *FSR* was assigned to 25%. On the other hand, the error rate decreased dramatically for all optimizers when the *FSR* was assigned to 35% and 75% and the minimum error rates achieved when *FSR* was 35%. For *FSR* of 35%, the error rate decreased 3.3%, the feature reduction ratio was 75%, and the running time reduced by 47% of the time needed for applying the wrapper without chi-square.
- *KrvskpEW* results: For all optimizers, the error rate increased when *FSR* was assigned to 75%. On the other hand, the error rate decreased dramatically for all optimizers when the *FSR* was assigned to 35% and 25% and the minimum error rates achieved when *FSR* was 35%. For *FSR* of 35%, the error rate decreased by 42%. The feature reduction ratio was 80.6%, and the running time reduced by 38% of the time needed for applying the wrapper without chi-square.

Table 6 Wrapper results based on error rate evaluation metric

Benchmark	BPSO		BMVO		BMFO		BGWO		Avg. error rate for all wrappers
	Avg_error	Std_error	Avg_error	Std_error	Avg_error	Std_error	Avg_error	Std_error	
Breastcancer	0.05731	0.00384	0.04571	0.01189	0.04769	*0.00343*	**0.03831**	0.00512	0.04605
BreastEW	0.08856	0.00758	0.05337	0.01690	**0.05152**	*0.00700*	0.05160	0.01771	0.05908
clean1	0.31163	*0.00659*	**0.26482**	0.04299	0.30452	0.03051	0.29903	0.05165	0.29965
Colon	0.33379	0.01098	0.34843	0.08478	**0.32729**	*0.00882*	0.32857	0.06717	0.35291
CongressEW	0.14335	*0.01085*	**0.07225**	0.01658	0.08330	0.01170	0.09670	0.03511	0.09574
diabetes	0.32911	0.00701	0.27274	0.01224	0.26262	0.00075	0.31072	0.04012	0.28326
Exactly	0.40063	*0.00308*	0.36902	0.05988	**0.31462**	0.05449	0.38764	0.04435	0.36430
Exactly2	**0.30456**	*0.00378*	0.31453	0.02649	0.31998	0.01461	0.31550	0.05551	0.31485
HeartEW	0.19855	*0.00903*	0.16185	0.04990	**0.12977**	0.01791	0.14680	0.05958	0.15835
Ionosphere	0.19973	*0.00820*	0.20004	0.04220	0.18831	0.02704	0.18979	0.04589	0.19030
KrvskpEW	0.26534	0.01305	0.06992	0.03114	**0.05415**	0.00888	0.12766	0.05444	0.09977
Leukemia	**0.60588**	*0.00800*	0.60800	0.03628	0.61863	0.01684	0.62500	*0.00000*	0.61751
liver	0.37334	0.00854	0.30167	0.01253	**0.29822**	*0.00197*	0.34868	0.01247	0.31964
Lymphography	0.39744	*0.00954*	0.36876	0.08574	0.40735	0.04827	0.38247	0.06742	0.38963
M-of-n	0.29773	*0.01139*	0.16624	0.05170	**0.14275**	0.04612	0.22069	0.05490	0.18724
parkinsons2	0.35114	*0.00598*	**0.31764**	0.03028	0.33507	0.03451	0.38866	0.03449	0.36191
penglung	0.45206	0.00993	0.40019	0.05121	0.40456	*0.00521*	0.40000	0.05130	0.40637
semeion	0.07605	*0.00178*	0.06709	0.00956	0.06165	0.00295	**0.06093**	0.00799	0.06463
SonarEW	0.51241	*0.01447*	0.45328	0.09979	**0.42972**	0.05277	0.53696	0.07883	0.48029
SpectEW	0.24045	*0.00321*	0.24518	0.01893	0.23848	0.01172	0.24665	0.02267	0.23997
Tic-tac-toc	0.34068	0.00634	0.30190	0.02390	0.33906	*0.00403*	0.29393	0.02416	0.31600

(continued)

Table 6 (continued)

Benchmark	BPSO		BMVO		BMFO		BGWO		Avg. error rate for all wrappers
	Avg_error	Std_error	Avg_error	Std_error	Avg_error	Std_error	Avg_error	Std_error	
Vertebral	0.15487	0.00576	0.11781	0.00050	0.11768	*0.00009*	**0.11478**	0.01624	0.12372
Vote	0.13573	*0.00989*	**0.08077**	0.03266	0.10579	0.02149	0.10305	0.04856	0.09799
WavefromEW	0.35087	0.01005	0.26312	0.01542	**0.24068**	*0.00996*	0.25945	0.01750	0.26474
WineEW	0.10598	0.01351	0.05483	0.04787	0.03503	*0.01283*	0.03245	0.04346	0.04449
Zoo	0.29805	0.01840	0.19068	0.04935	0.20432	*0.00738*	0.24545	0.10260	0.21455
orlraws10P	0.58186	0.00990	0.56136	0.03143	**0.54545**	*0.00000*	**0.54545**	*0.00000*	0.55398
CLL_SUB_111	0.66054	0.01926	**0.61773**	0.09378	0.76092	*0.01583*	0.75385	0.05353	0.70758
GLIOMA	0.21383	0.02294	0.16908	0.01081	**0.16667**	*0.00000*	**0.16667**	0.00000	0.17526
lung_discrete	0.76575	*0.01600*	**0.71956**	0.13046	0.75775	0.03044	0.74375	0.07560	0.75785
Benchmark	BWOA		BFFA		BBAT		BCS		Avg. error rate for all wrappers
	Avg_error	Std_error	Avg_error	Std_error	Avg_error	Std_error	Avg_error	Std_error	
Breastcancer	0.04501	0.01091	0.04469	0.01093	0.04301	0.00924	0.04663	0.00802	0.04605
BreastEW	0.05662	0.01306	0.05373	0.01332	0.06245	0.02035	0.05477	0.00895	0.05908
clean1	0.29934	0.03021	0.30633	0.02731	0.31283	0.02460	0.29869	0.02126	0.29965
Colon	0.34207	0.07054	0.32836	0.06667	0.36843	0.07252	0.44636	0.17960	0.35291
CongressEW	0.08333	0.01625	0.08942	0.03133	0.11352	0.03593	0.08405	0.01847	0.09574
diabetes	0.26558	0.00374	**0.26190**	*0.00000*	0.29753	0.02755	0.26589	0.00368	0.28326
Exactly	0.35417	0.04136	0.35335	0.05286	0.38890	0.03986	0.34608	0.02960	0.36430
Exactly2	0.30861	0.02291	0.31485	0.01467	0.32091	0.06218	0.31984	0.02628	0.31485
HeartEW	0.14737	0.04954	0.15928	0.04713	0.17058	0.05249	0.15262	0.03634	0.15835
Ionosphere	0.18813	0.02959	**0.18027**	0.02729	0.19412	0.03260	0.18204	0.02921	0.19030
KrvskpEW	0.05966	0.01144	0.05874	0.01152	0.10135	0.03054	0.06133	0.01284	0.09977

(continued)

Table 6 (continued)

Benchmark	BWOA		BFFA		BBAT		BCS		Avg. error rate for all wrappers
	Avg_error	Std_error	Avg_error	Std_error	Avg_error	Std_error	Avg_error	Std_error	
Leukemia	0.61706	0.02669	0.62500	*0.00000*	0.61850	0.00993	0.62200	0.01057	0.61751
liver	0.30259	0.01400	0.30805	0.02360	0.32367	0.05707	0.30087	0.00644	0.31964
Lymphography	0.41185	0.07492	0.40026	0.06383	**0.36441**	0.07548	0.38450	0.06951	0.38963
M-of-n	0.15610	0.03604	0.16949	0.02728	0.19863	0.04803	0.14625	0.03756	0.18724
parkinsons2	0.38041	0.02968	0.38398	0.02308	0.37284	0.02183	0.36555	0.03135	0.36191
penglung	0.40019	0.05121	0.39381	0.04577	**0.39831**	0.05550	0.40181	0.01313	0.40637
semeion	0.06237	0.00633	0.06112	0.00668	0.06534	0.00729	0.06251	0.00575	0.06463
SonarEW	0.48278	0.05963	0.47400	0.07383	0.47043	0.09620	0.48276	0.07012	0.48029
SpectEW	0.23932	0.02910	0.23722	0.02423	0.24290	0.03005	**0.22958**	0.01614	0.23997
Tic-tac-toc	0.31697	0.01797	0.31805	0.02452	**0.28861**	0.03324	0.32880	0.01387	0.31600
Vertebral	0.11787	0.00100	0.11806	0.00145	0.13051	0.01925	0.11817	0.00144	0.12372
Vote	0.09180	0.01925	0.08862	0.01963	0.08864	0.03971	0.08952	0.01656	0.09799
WavefromEW	0.24492	0.01195	0.25111	0.01425	0.26248	0.02274	0.24527	0.01128	0.26474
WineEW	0.02580	0.02678	**0.01775**	0.02776	0.04438	0.04296	0.03970	0.03672	0.04449
Zoo	**0.18636**	0.05482	0.19550	0.04447	0.19823	0.04673	0.19782	0.02447	0.21455
orlraws10P	**0.54545**	0.00000	**0.54545**	0.00000	0.55350	0.03453	0.55327	0.03475	0.55398
CLL_SUB_111	0.75592	0.02370	0.75669	0.02646	0.72885	0.08147	0.62612	0.09941	0.70758
GLIOMA	**0.16667**	*0.00000*	**0.16667**	*0.00000*	0.17717	0.03135	0.17533	0.02752	0.17526
lung_discrete	0.77056	0.05880	0.77406	0.07570	0.75569	0.05559	0.77569	0.03571	0.75785

Table 7 Wrapper results based on number of selected feature evaluation metric

Benchmark	BPSO		BMVO		BMFO		BGWO		Avg. FS for all wrappers
	Avg_FS	Std_FS	Avg_FS	Std_FS	Avg_FS	Std_FS	Avg_FS	Std_FS	
Breastcancer	4.27600	*0.20449*	4.03900	0.80677	3.97850	0.22177	4.94950	1.09950	4.28975
BreastEW	12.56450	*0.71544*	**9.04850**	3.17600	14.71200	1.67435	16.19750	2.58540	14.13969
clean1	59.09750	5.19143	**44.23350**	13.27380	81.23650	*3.59099*	83.74950	6.67188	74.38500
Colon	819.21100	30.38989	641.32200	155.65769	999.21500	*2.69674*	994.35000	19.65565	740.49125
CongressEW	7.29950	*0.38703*	**4.02700**	2.15064	6.61350	0.80038	9.34700	1.92564	7.20619
diabetes	**3.78250**	0.18396	3.83500	0.37371	3.98950	*0.02282*	4.34950	1.42380	4.08781
Exactly	5.90000	*0.33353*	**5.47250**	1.20796	6.77700	0.49887	6.94950	2.43796	6.58144
Exactly2	5.90950	*0.33668*	5.74950	1.19470	7.07950	0.66337	5.44800	2.47207	6.16613
HeartEW	**5.88350**	*0.25742*	6.01250	1.01534	6.16400	1.08723	7.55100	1.38927	9.38888
Ionosphere	15.10750	0.66967	**4.05000**	0.86212	13.04600	1.35739	16.39950	2.41301	13.00163
KrvskpEW	14.63550	*0.92231*	**12.23600**	5.70800	20.04750	1.94722	20.04500	2.47797	18.68206
Leukemia	2233.52750	183.12796	1417.38500	358.88673	3552.08900	*26.08108*	3568.85000	47.07136	2728.15819
liver	**2.88000**	*0.11774*	4.05850	0.94223	4.34550	0.15629	3.15050	1.26715	4.00231
Lymphography	7.87050	*0.38865*	**6.66500**	1.59885	9.26500	1.06637	9.24600	2.28626	8.75150
M-of-n	**5.82700**	*0.33904*	7.37250	1.08614	7.16750	0.51632	7.59750	1.38985	7.51094
parkinsons2	10.09600	*0.42172*	**4.41100**	2.64579	8.29500	2.26027	11.54850	2.21079	9.81519
penglung	**107.15700**	3.67943	156.68450	17.40886	161.69700	*1.25262*	164.11820	3.45954	149.28721
semeion	**90.48300**	3.28617	97.77450	28.71833	133.49700	*3.17395*	133.49150	8.18166	123.94375
SonarEW	23.79800	*1.53167*	**16.87350**	5.88118	28.30800	2.56669	29.69400	3.39115	26.79800
SpectEW	9.24800	*0.53394*	**7.32825**	1.42327	9.15750	0.99697	10.40000	2.47876	9.64209
Tic-tac-toc	**4.24550**	0.23869	4.76700	0.65975	5.01200	*0.09362*	5.69550	1.17447	5.12531

(continued)

Table 7 (continued)

Benchmark	BPSO		BMVO		BMFO		BGWO		Avg. FS for all wrappers
	Avg_FS	Std_FS	Avg_FS	Std_FS	Avg_FS	Std_FS	Avg_FS	Std_FS	
Vertebral	2.98831	*0.05045*	**2.36950**	0.49555	2.47350	0.06683	3.10400	0.78426	2.73710
Vote	7.29200	*0.39359*	**4.67950**	1.18541	6.25550	0.71228	6.80000	1.54238	6.55994
WavefromEW	**16.20800**	*0.71393*	17.33800	2.84485	21.89550	1.83077	22.19650	3.20550	21.28356
WineEW	5.84000	*0.22658*	**5.75200**	1.66376	6.41250	0.38135	6.45600	1.35113	6.53750
Zoo	**6.96550**	0.26094	8.36550	1.96264	8.76550	0.18662	7.85000	1.56525	8.41506
orlraws10P	3085.66500	263.86619	**910.06450**	378.19981	5160.40450	56.79716	5163.40000	50.70077	3698.46144
CLL_SUB_111	3795.08000	407.64927	**1242.09200**	555.68430	5675.09550	*24.71139*	5696.70000	47.71086	4303.42700
GLIOMA	1464.25800	168.60283	785.90850	252.48067	2216.98300	*15.37067*	2199.45000	33.72407	1699.65063
lung_discrete	113.69150	9.14689	**69.42200**	26.59010	160.08050	*2.53049*	167.30950	10.00637	141.49419
Benchmark	BWOA		BFFA		BBAT		BCS		Avg. FS for all wrappers
	Avg_FS	Std_FS	Avg_FS	Std_FS	Avg_FS	Std_FS	Avg_FS	Std_FS	
Breastcancer	4.25000	0.80879	**3.91750**	0.95308	4.81000	1.31225	4.09750	0.61301	4.28975
BreastEW	14.92750	2.10310	15.53150	2.10406	14.97000	3.79449	15.16600	1.94300	14.13969
clean1	81.61750	4.01546	81.96650	4.33514	81.00000	16.99767	82.17900	4.41300	74.38500
Colon	998.05450	22.05258	1003.44450	22.88838	427.78200	404.34718	**40.55100**	34.28339	740.49125
CongressEW	6.67000	1.36816	8.07100	1.76304	8.57150	2.35545	7.05000	0.91796	7.20619
diabetes	4.00400	0.20340	3.81500	0.30624	4.99400	0.93225	3.93300	0.19992	4.08781
Exactly	6.87150	0.88945	6.90450	1.09640	6.34850	3.22701	7.42800	0.91311	6.58144
Exactly2	6.80000	1.10281	6.97400	0.82865	**4.56400**	2.87585	6.80450	0.81456	6.16613
HeartEW	7.54950	1.27537	7.19250	1.22089	27.84907	95.82213	6.90900	0.95138	9.38888
Ionosphere	14.32450	2.11429	15.40450	2.53687	11.52700	6.87931	14.15400	*0.13817*	13.00163
KrvskpEW	20.31150	1.90557	20.08400	1.93013	22.73900	4.25945	19.35800	1.36367	18.68206

(continued)

Table 7 (continued)

Benchmark	BWOA		BFFA		BBAT		BCS		Avg. FS for all wrappers
	Avg_FS	Std_FS	Avg_FS	Std_FS	Avg_FS	Std_FS	Avg_FS	Std_FS	
Leukemia	3420.54800	681.97570	3567.85000	46.62423	3254.33650	802.76964	**810.67950**	798.25483	2728.15819
liver	4.57550	0.80514	4.51100	0.81139	4.19300	0.94852	4.30450	0.47982	4.00231
Lymphography	8.93300	2.21585	9.29950	1.21551	10.18100	3.60483	8.55200	1.20729	8.75150
M-of-n	7.99700	0.81783	8.09150	0.82342	8.30550	1.41580	7.72900	0.73706	7.51094
parkinsons2	10.97950	2.67342	11.34450	2.31419	11.70350	3.44535	10.14350	1.65069	9.81519
penglung	164.40500	8.55271	159.40900	8.75646	119.97150	49.20093	160.85550	3.45664	149.28721
semeion	132.73300	5.71409	132.85700	6.71270	137.09500	23.88630	133.61900	7.51771	123.94375
SonarEW	29.50450	2.63927	29.23000	4.19634	28.04350	9.87927	28.93250	2.78921	26.79800
SpectEW	9.72050	1.69353	9.64550	1.32032	11.92850	3.04656	9.70850	1.00248	9.64209
Tic-tac-toc	4.99150	0.42682	5.01750	0.44163	6.28600	1.13166	4.98750	0.29250	5.12531
Vertebral	2.48300	0.50411	2.45300	0.48756	3.40300	0.99414	2.62250	0.32244	2.73710
Vote	6.55550	1.28605	6.62100	1.99015	7.86200	2.02952	6.41400	1.10439	6.55994
WavefromEW	22.24750	1.48031	22.95300	2.05065	25.12250	5.08223	22.30750	2.27616	21.28356
WineEW	6.84250	1.12260	7.03250	1.50575	7.43800	2.07055	6.52650	0.91614	6.53750
Zoo	8.84650	1.80794	8.34850	2.00514	9.68600	2.53140	8.49300	*0.11115*	8.41506
orlraws10P	4950.14600	891.17269	5163.85000	*50.07182*	2063.85850	1169.98979	3090.30300	1145.14116	3698.46144
CLL_SUB_111	5680.91900	43.57653	5676.66150	46.08112	4219.13400	2273.34642	2441.73400	1813.23969	4303.42700
GLIOMA	2208.81600	32.60156	2220.40200	32.96888	**767.02550**	730.56999	1734.36200	424.06189	1699.65063
lung_discrete	161.93250	7.99468	155.80750	6.27096	142.81500	56.17989	160.89500	7.44850	141.49419

Table 8 Wrapper results based on running time evaluation metric (seconds per run)

Benchmark	BPSO		BMVO		BMFO		BGWO		Avg. running time for all wrappers
	Avg_time	Std_time	Avg_time	Std_time	Avg_time	Std_time	Avg_time	Std_time	
Breastcancer	**7.13380**	0.15470	10.28270	0.12364	31.47600	0.35449	7.93095	*0.09892*	12.25522
BreastEW	9.27400	*0.12668*	11.62795	0.27920	43.01425	0.36191	12.23510	0.17417	15.90885
clean1	21.08960	0.80296	19.03600	0.71303	105.98430	0.83346	36.79160	0.43045	36.08978
Colon	134.80095	0.61398	80.24920	0.90308	408.78090	3.74128	243.61730	13.55800	131.95820
CongressEW	6.69755	0.12632	9.27865	0.15085	28.52065	0.29965	7.98680	0.14314	11.42631
diabetes	**7.07685**	*0.09762*	10.34910	0.10479	30.82800	0.21795	7.77570	0.10515	12.36188
Exactly	**10.16510**	0.12796	13.35480	0.42686	48.17775	0.42172	12.12135	0.47120	17.82254
Exactly2	10.37020	0.46621	13.48120	0.30716	48.48310	0.45567	11.99755	0.39657	17.75914
HeartEW	5.76025	*0.07167*	8.34180	0.08134	23.58960	0.24415	6.60070	0.08339	9.80266
Ionosphere	8.09340	0.09423	10.13655	*0.09360*	34.05360	0.29216	10.72370	0.17625	13.06545
KrvskpEW	95.43855	16.69085	**83.78125**	10.21809	531.49180	*3.92489*	117.93470	6.16393	181.60093
Leukemia	468.72441	7.07622	269.41850	5.74707	1448.76690	22.37034	745.86345	21.25838	448.97026
liver	**5.55020**	*0.05807*	8.70110	0.12992	23.45335	0.15252	5.94480	0.07411	9.91338
Lymphography	6.48795	3.48706	8.27490	*0.05173*	23.31545	0.25977	6.92740	0.12720	9.80135
M-of-n	10.34865	0.60677	13.40275	0.30019	46.41860	0.39572	11.41760	0.34523	17.61491
parkinsons2	6.38715	*0.08032*	8.86540	0.14463	25.63025	0.14654	7.75740	0.08587	10.67310
penglung	26.01990	*0.20295*	20.20280	0.39516	84.36395	0.82773	46.25370	5.98604	29.39892
semeion	**112.19665**	19.88140	116.87500	24.76154	1253.99040	*5.67611*	280.35370	11.55743	391.39309
SonarEW	9.06250	*0.07240*	10.28545	0.11832	34.86515	0.49672	13.18695	0.10775	13.48201
SpectEW	6.48795	*0.04811*	8.95950	0.08777	26.23385	0.22266	7.97920	0.07863	10.73842
Tic-tac-toc	**8.32030**	*0.10923*	12.05905	0.26314	38.72000	0.32505	9.70450	0.35626	15.13844

(continued)

Table 8 (continued)

Benchmark	BPSO		BMVO		BMFO		BGWO		Avg. running time for all wrappers
	Avg_time	Std_time	Avg_time	Std_time	Avg_time	Std_time	Avg_time	Std_time	
Vertebral	**5.87270**	*0.06053*	9.19770	0.10495	24.98535	0.18750	6.25375	0.08952	10.50414
Vote	6.04395	*0.08376*	8.80310	0.17003	25.38275	0.33818	7.31600	0.09539	10.55543
WavefromEW	**213.91195**	18.54486	214.55120	23.02820	1727.02300	39.38153	374.79180	18.19121	585.70909
WineEW	**5.39335**	*0.05328*	8.22025	0.08197	22.49395	0.23881	6.41650	0.07884	9.47096
Zoo	5.62130	0.12037	8.48075	0.11356	22.98315	0.22777	6.72875	0.08613	9.66266
orlraws10P	698.99865	5.30714	387.53760	22.20256	2139.25769	30.58137	1301.40103	333.40668	693.22410
CLL_SUB_111	710.66885	18.43630	407.30210	8.30082	2363.59005	14.52444	1399.51979	16.75518	760.77610
GLIOMA	283.63855	4.58025	151.22510	2.52308	807.22161	8.22924	482.23025	22.00605	261.01438
lung_discrete	25.57780	0.17161	20.07705	0.25148	82.34460	0.41533	43.15785	0.89546	28.93498
Benchmark	BWOA		BFFA		BBAT		BCS		Avg. running time for all wrappers
	Avg_time	Std_time	Avg_time	Std_time	Avg_time	Std_time	Avg_time	Std_time	
Breastcancer	14.89470	0.26048	7.13455	0.17182	7.15925	0.28652	12.02980	0.12006	12.25522
BreastEW	18.26050	0.41761	8.94705	0.15856	**8.65590**	0.64392	15.25605	0.13063	15.90885
clean1	38.40080	1.90922	**16.78245**	*0.24823*	18.13370	2.45826	32.49980	0.41115	36.08978
Colon	129.57414	3.88427	**13.92300**	*0.20932*	14.78265	1.99488	29.93745	0.36062	131.95820
CongressEW	15.11425	0.26832	6.55370	0.14640	**6.47100**	0.26917	10.78785	*0.10619*	11.42631
diabetes	15.92825	0.21285	7.23750	0.10011	7.43525	0.23435	12.26440	0.13904	12.36188
Exactly	20.48465	0.30554	10.30655	*0.12570*	10.30675	1.66542	17.66340	0.14973	17.82254
Exactly2	20.43545	0.35203	10.29435	0.16709	**9.22435**	1.49059	17.78690	*0.15130*	17.75914
HeartEW	13.58960	0.15012	5.85805	0.23396	**5.52360**	0.09274	9.15770	0.17716	9.80266
Ionosphere	16.45885	0.22673	6.99065	0.19696	**6.44800**	0.53220	11.61885	0.13817	13.06545
KrvskpEW	174.29655	5.39531	110.73630	5.82115	134.61060	19.15036	204.51765	8.98367	181.60093

(continued)

Table 8 (continued)

Benchmark	BWOA		BFFA		BBAT		BCS		Avg. running time for all wrappers
	Avg_time	Std_time	Avg_time	Std_time	Avg_time	Std_time	Avg_time	Std_time	
Leukemia	448.26620	5.37365	**49.81575**	9.50213	55.93734	5.57145	104.96951	*1.61859*	448.97026
liver	13.91580	0.20839	5.86075	0.08863	5.92375	0.11529	9.95730	0.21707	9.91338
Lymphography	13.36785	0.19344	5.54515	0.13048	**5.42355**	0.12725	9.06855	0.12217	9.80135
M-of-n	20.39685	0.96515	**10.15700**	*0.12309*	11.20680	1.22543	17.57100	0.26213	17.61491
parkinsons2	15.45160	4.22958	**5.70390**	0.17678	5.89475	0.08942	9.69435	0.09975	10.67310
penglung	31.02840	0.17072	**6.88880**	0.14859	7.41525	0.29724	13.01855	0.14372	29.39892
semeion	335.16911	17.16440	241.83800	11.17802	282.62628	52.70882	508.09561	24.70120	391.39309
SonarEW	16.91945	0.21055	6.47705	0.08860	**6.35840**	0.28093	10.70110	0.24568	13.48201
SpectEW	14.37215	0.25106	6.02745	0.11764	**5.92040**	0.13919	9.92685	0.16231	10.73842
Tic-tac-toc	18.13625	0.22674	8.83325	0.12522	10.04535	0.82020	15.28880	0.22887	15.13844
Vertebral	14.49450	0.17849	6.15050	0.09254	6.36935	0.12310	10.70925	0.12557	10.50414
Vote	14.16485	0.14220	7.00885	4.32491	**5.85845**	0.16543	9.86545	0.18568	10.55543
WavefromEW	524.57385	20.98163	355.35800	*7.61042*	530.33177	166.99772	745.13113	8.07353	585.70909
WineEW	13.21600	0.14107	5.45050	3.35143	5.51115	0.14689	9.06595	0.11225	9.47096
Zoo	13.66735	0.14735	**5.30090**	0.08704	5.51100	*0.07340*	9.00810	0.11115	9.66266
orlraws10P	685.27160	12.68253	89.82715	2.08516	**73.53670**	13.28379	169.96240	*2.04080*	693.22410
CLL_SUB_111	749.18055	4.75014	122.02440	8.24413	**122.51595**	35.19364	211.40710	*2.68361*	760.77610
GLIOMA	254.36205	*1.19393*	28.90070	6.09773	**26.01605**	6.23975	54.52070	10.02217	261.01438
lung_discrete	32.65420	0.18930	**7.20610**	0.25924	7.62425	0.41573	12.83800	*0.12458*	28.93498

Table 9 Hybrid chi-square Wrapper results based on error rate evaluation metric

Benchmark	Chi-square BPSO		Chi-square BMVO		Chi-square BMFO		Chi-square BGWO		Avg. error rate
	Avg_error	Std_error	Avg_error	Std_error	Avg_error	Std_error	Avg_error	Std_error	
Breastcancer(*FSR* = 25%)	0.08242	0.00174	0.05195	0.00000	0.05195	0.00000	0.05197	0.00012	0.05660
Breastcancer(*FSR* = 35%)	0.07106	0.00231	0.04742	0.00637	0.04534	0.00080	0.04612	0.00661	0.04964
Breastcancer(*FSR* = 75%)	**0.05770**	0.00331	**0.03988**	0.00295	**0.03897**	0.00000	**0.03766**	0.00829	**0.04167**
penglung(*FSR* = 25%)	0.47644	0.00888	0.45606	0.08614	0.45913	0.00942	0.45625	0.06117	0.46126
penglung(*FSR* = 35%)	0.47294	0.01574	0.43613	0.07244	0.44706	0.00789	0.43125	0.07560	0.44978
penglung(*FSR* = 75%)	**0.45856**	0.00740	**0.40094**	0.06150	**0.41250**	0.00605	**0.41875**	0.06117	**0.41776**
WavefromEW(*FSR* = 25%)	0.34901	0.00988	0.26460	0.00839	0.25269	0.00588	0.26674	0.01707	0.27534
WavefromEW(*FSR* = 35%)	**0.30342**	0.00812	**0.22788**	0.00940	**0.21314**	0.00881	**0.22463**	0.02089	**0.23136**
WavefromEW(*FSR* = 75%)	0.33629	0.00978	0.25331	0.01428	0.22963	0.01172	0.25650	0.02051	0.25406
KrvskpEW(*FSR* = 25%)	0.27280	0.00941	0.06278	0.01624	0.06477	0.00391	0.12282	0.06977	0.09737
KrvskpEW(*FSR* = 35%)	0.27682	0.01353	**0.05675**	0.01152	**0.04658**	0.00087	**0.11681**	0.06402	**0.09546**
KrvskpEW(*FSR* = 75%)	**0.27158**	0.01444	0.07804	0.03211	0.05846	0.01084	0.13051	0.05569	0.10454
orlraws10P(*FSR* = 25%)	0.54109	0.01433	0.52386	0.06762	0.50805	0.02425	0.50909	0.04569	0.51351
orlraws10P(*FSR* = 35%)	**0.51673**	0.01417	**0.51168**	0.04888	**0.45909**	0.02033	**0.45909**	0.00000	**0.47431**
orlraws10P(*FSR* = 75%)	0.57668	0.01079	0.53741	0.04009	0.54532	0.00061	0.54545	0.00000	0.54725
CLL_SUB_111(*FSR* = 25%)	0.66842	0.01978	**0.57600**	0.09861	**0.75458**	0.01664	**0.71923**	0.05732	0.69146
CLL_SUB_111(*FSR* = 35%)	0.66735	0.01751	0.58858	0.08507	0.75662	0.01327	0.74231	0.04516	**0.69084**
CLL_SUB_111(*FSR* = 75%)	**0.66308**	0.02230	0.60362	0.07938	0.76435	0.01044	0.73462	0.04652	0.69440

Table 9 (continued)

Benchmark	Chi-square BWOA		Chi-square BFFA		Chi-square BBAT		Chi-square BCS		Avg. error rate
	Avg_error	Std_error	Avg_error	Std_error	Avg_error	Std_error	Avg_error	Std_error	
Breastcancer(*FSR* = 25%)	0.05195	0.00000	0.05195	0.00000	0.05870	0.00117	0.05195	0.00000	0.05660
Breastcancer(*FSR* = 35%)	0.04810	0.00614	0.04286	0.00611	0.05056	0.00531	0.04564	0.00113	0.04964
Breastcancer(*FSR* = 75%)	**0.03912**	0.00073	**0.03814**	0.00503	**0.04276**	0.00912	**0.03912**	0.00036	**0.04167**
penglung(*FSR* = 25%)	0.45600	0.07280	0.46256	0.05840	0.46825	0.04664	0.45538	0.02032	0.46126
penglung(*FSR* = 35%)	0.45000	0.06283	0.43750	0.09511	0.46875	0.05893	0.45463	0.01157	0.44978
penglung(*FSR* = 75%)	**0.41231**	0.05834	**0.40631**	0.05550	**0.42731**	0.05379	**0.40538**	0.01281	**0.41776**
WavefromEW(*FSR* = 25%)	0.26609	0.01201	0.26527	0.01613	0.27316	0.01276	0.26515	0.01560	0.27534
WavefromEW(*FSR* = 35%)	**0.21704**	0.01319	**0.22085**	0.00862	**0.23046**	0.01865	**0.21347**	0.01623	**0.23136**
WavefromEW(*FSR* = 75%)	0.22996	0.01478	0.23532	0.01164	0.25575	0.01926	0.23576	0.00953	0.25406
KrvskpEW(*FSR* = 25%)	**0.05225**	0.00468	**0.05115**	0.00432	0.10286	0.02105	**0.04951**	0.00401	0.09737
KrvskpEW(*FSR* = 35%)	0.05972	0.00564	0.05843	0.00658	**0.09003**	0.03937	0.05853	0.00639	**0.09546**
KrvskpEW(*FSR* = 75%)	0.06665	0.00889	0.06638	0.01133	0.10171	0.02825	0.06300	0.01100	0.10454
orlraws10P(*FSR* = 25%)	0.49950	0.04234	0.51695	0.03960	0.49368	0.04577	0.51586	0.02639	0.51351
orlraws10P(*FSR* = 35%)	**0.45536**	0.00267	**0.45455**	0.00000	**0.48105**	0.04033	**0.45691**	0.00680	**0.47431**
orlraws10P(*FSR* = 75%)	0.54545	0.00000	0.54545	0.00000	0.53541	0.03661	0.54682	0.00610	0.54725
CLL_SUB_111(*FSR* = 25%)	**0.75435**	0.03687	**0.73331**	0.07558	0.69831	0.07136	0.62750	0.12365	0.69146
CLL_SUB_111(*FSR* = 35%)	0.73612	0.04922	0.74131	0.04338	0.71169	0.07743	0.58277	0.18523	**0.69084**
CLL_SUB_111(*FSR* = 75%)	0.75665	0.02300	0.73877	0.05672	**0.67254**	0.10692	**0.62162**	0.11831	0.69440

Table 10 Hybrid Chi-square Wrapper results based on the number of selected feature metrics

Benchmark	Chi-square BPSO		Chi-square BMVO		Chi-square BMFO		Chi-square BGWO		Wrappers Avg. FS
	Avg_FS	Std_FS	Avg_FS	Std_FS	Avg_FS	Std_FS	Avg_FS	Std_FS	
Breastcancer(*FSR* = 25%)	1.32950	0.03300	2.00000	0.00000	2.00000	0.00000	1.99950	0.00224	1.89756
Breastcancer(*FSR* = 35%)	1.69500	0.06428	2.35000	0.48936	2.50850	0.06184	2.44950	0.50986	2.34944
Breastcancer(*FSR* = 75%)	3.33900	0.17444	4.26650	0.51290	4.45000	0.08285	4.25300	0.96139	4.25244
penglung(*FSR* = 25%)	30.25500	1.51765	30.40050	11.53053	40.78400	1.21833	41.45000	5.25632	37.88431
penglung(*FSR* = 35%)	41.02650	2.86828	44.03200	13.86210	57.35100	0.49538	57.70000	4.34196	51.93744
penglung(*FSR* = 75%)	81.80350	1.97932	111.49300	16.48659	120.90100	0.76639	121.35000	8.11934	112.60731
WavefromEW(*FSR* = 25%)	4.61150	0.17257	7.24550	0.87520	7.98950	0.08198	6.94950	1.31371	7.23863
WavefromEW(*FSR* = 35%)	6.28400	0.25349	9.58650	0.70323	11.25350	0.84386	9.74850	1.44434	9.94175
WavefromEW(*FSR* = 75%)	12.13350	0.51933	15.28000	1.88200	17.56000	1.76422	16.34850	2.23078	16.62688
KrvskpEW(*FSR* = 25%)	4.23500	0.15046	5.67400	0.61813	5.60950	0.19291	6.69450	1.41746	5.78488
KrvskpEW(*FSR* = 35%)	5.86850	0.29498	5.62950	0.99248	8.19600	3.99077	8.44500	1.73115	7.37425
KrvskpEW(*FSR* = 75%)	12.22958	21.87148	9.24500	2.89442	15.53800	2.04896	14.59750	2.10943	14.09201
orlraws10P(*FSR* = 25%)	856.73750	67.48478	155.43000	35.18915	1282.58100	12.78039	1291.25000	20.87778	964.25125
orlraws10P(*FSR* = 35%)	1257.10300	101.52459	253.90600	65.66660	1846.76200	7.45600	1850.15000	27.85352	1469.77081
orlraws10P(*FSR* = 75%)	2496.33700	246.53248	677.10650	244.41077	3860.51750	16.13220	3864.90000	42.55387	2989.22250
CLL_SUB_111(*FSR* = 25%)	1036.96150	110.33180	290.41650	115.72560	1412.43650	16.00500	1424.20000	27.49660	1094.32644
CLL_SUB_111(*FSR* = 35%)	1430.24750	123.50656	456.42250	196.61860	1989.96600	14.50531	1981.20000	31.17455	1543.46106
CLL_SUB_111(*FSR* = 75%)	2163.30946	243.60428	1240.66650	663.52569	4237.86900	22.12811	4258.75000	40.33919	3072.48799

(continued)

Table 10 (continued)

Benchmark	Chi-square BWOA		Chi-square BFFA		Chi-square BBAT		Chi-square BCS		Wrappers Avg. FS
	Avg_FS	Std_FS	Avg_FS	Std_FS	Avg_FS	Std_FS	Avg_FS	Std_FS	
Breastcancer($FSR = 25\%$)	2.00000	0.00000	2.00000	0.00000	1.85150	0.02519	2.00000	0.00000	1.89756
Breastcancer($FSR = 35\%$)	2.29950	0.47050	2.70000	0.47016	2.30750	0.38510	2.48550	0.08727	2.34944
Breastcancer($FSR = 75\%$)	4.41650	0.48652	4.47650	0.72355	4.35150	1.12799	4.46650	0.29828	4.25244
penglung($FSR = 25\%$)	42.48900	4.71903	40.37800	4.84407	36.83500	12.03345	40.48300	1.34402	37.88431
penglung($FSR = 35\%$)	57.05000	4.71811	55.85000	4.86962	45.37150	13.97325	57.11850	1.37767	51.93744
penglung($FSR = 75\%$)	123.03150	5.34828	122.60200	0.17209	100.02100	43.57813	119.65650	1.89439	112.60731
WavefromEW($FSR = 25\%$)	7.76500	0.54715	7.96450	0.53275	7.53150	0.93116	7.85200	0.36929	7.23863
WavefromEW($FSR = 35\%$)	11.08700	0.81488	10.43400	0.74457	10.05850	1.25767	11.08200	1.17076	9.94175
WavefromEW($FSR = 75\%$)	18.42150	49.42853	17.84000	1.90115	17.96900	2.21691	17.46250	1.91832	16.62688
KrvskpEW($FSR = 25\%$)	5.96400	0.52773	5.90700	0.42877	6.41900	0.68024	5.77600	0.48958	5.78488
KrvskpEW($FSR = 35\%$)	7.14850	1.07651	7.26450	1.01818	8.83750	1.34113	7.60450	0.69069	7.37425
KrvskpEW($FSR = 75\%$)	14.83500	2.20712	14.75900	1.54652	16.48650	3.26442	15.04550	1.97153	14.09201
orlraws10P($FSR = 25\%$)	1250.13050	153.31573	1288.71700	18.97134	605.92350	661.28182	983.24050	258.30366	964.25125
orlraws10P($FSR = 35\%$)	1849.47650	29.69710	1848.03550	27.33772	1260.06150	864.00318	1592.67200	304.80679	1469.77081
orlraws10P($FSR = 75\%$)	3872.03950	35.47333	3858.45150	55.59436	2014.88450	1981.31652	3269.54350	740.83741	2989.22250
CLL_SUB_111($FSR = 25\%$)	1412.50150	21.41718	1417.20500	23.72441	1020.90600	515.99777	739.98450	554.05084	1094.32644
CLL_SUB_111($FSR = 35\%$)	1990.66450	20.47012	1979.70250	28.49044	1467.16650	807.87796	1052.31900	729.63050	1543.46106
CLL_SUB_111($FSR = 75\%$)	4253.20800	34.73969	4246.43200	40.32503	2520.92900	1913.70502	1658.74000	1426.47055	3072.48799

Table 11 Hybrid chi-square wrapper results based on running time metric (seconds per run)

Benchmark	Chi-square BPSO		Chi-square BMVO		Chi-square BMFO		Chi-square BGWO		Avg. running time
	Avg_time	Std_time	Avg_time	Std_time	Avg_time	Std_time	Avg_time	Std_time	
Breastcancer(*FSR* = 25%)	6.31932	0.13531	10.31235	0.17761	39.43488	6.11942	6.53133	0.10499	13.14366
Breastcancer(*FSR* = 35%)	6.45106	0.13456	12.07897	2.63979	40.52521	6.16573	6.79135	0.09676	14.06029
Breastcancer(*FSR* = 75%)	8.01747	1.74528	11.98559	2.30956	47.49558	0.71639	7.84572	0.22621	15.7608
penglung(*FSR* = 25%)	11.14784	2.19894	18.24725	0.22842	58.70525	0.61460	23.03650	4.21000	20.41094
penglung(*FSR* = 35%)	13.86715	3.49675	17.52285	3.85795	70.29095	1.21502	29.13640	5.74190	23.09220
penglung(*FSR* = 75%)	30.12681	6.85155	25.38012	4.73786	106.71364	13.60801	52.18999	8.59569	36.33404
WavefromEW(*FSR* = 25%)	92.32275	1.71963	132.27853	10.90714	351.98453	76.41230	67.57456	7.45866	137.4521313
WavefromEW(*FSR* = 35%)	118.34845	19.03172	174.62618	8.96916	733.18845	16.00377	169.88022	15.66579	275.93786
WavefromEW(*FSR* = 75%)	300.08172	16.55671	208.54524	49.19328	2213.57501	37.40672	307.73752	30.08385	615.09845
KrvskpEW(*FSR* = 25%)	33.47878	0.91956	44.59591	1.56646	159.66517	2.39429	38.82683	8.12912	59.73568
KrvskpEW(*FSR* = 35%)	38.22102	0.89047	47.59205	1.70395	187.99319	3.99077	47.25462	9.39387	68.40007
KrvskpEW(*FSR* = 75%)	65.42453	1.75432	67.64383	6.85876	395.46638	6.80144	85.41125	5.59700	137.61856
orlraws10P(*FSR* = 25%)	180.75921	1.87751	102.10586	1.82249	556.97339	4.05744	329.22187	8.30172	179.88697
orlraws10P(*FSR* = 35%)	292.04378	73.61213	187.21770	50.30957	1290.55225	22.94186	796.15400	104.74734	387.12303
orlraws10P(*FSR* = 75%)	548.03549	8.65546	303.23956	5.00935	1689.14421	22.07529	1026.55326	10.54131	547.47362
CLL_SUB_111(*FSR* = 25%)	206.23462	3.97390	116.93642	2.19115	644.48533	11.88602	378.81452	5.77622	209.46526
CLL_SUB_111(*FSR* = 35%)	455.51380	1.93319	214.46910	38.37687	1309.82365	100.47624	830.52305	38.49720	441.10499
CLL_SUB_111(*FSR* = 75%)	601.88747	12.22285	336.04425	6.50716	1877.23617	46.79718	1124.10558	50.37385	616.84725

(continued)

Table 11 (continued)

Benchmark	Chi-square BWOA		Chi-square BFFA		Chi-square BBAT		Chi-square BCS		Avg. running time
	Avg_time	Std_time	Avg_time	Std_time	Avg_time	Std_time	Avg_time	Std_time	
Breastcancer(*FSR* = 25%)	16.74223	3.30299	7.70536	1.72990	6.65900	0.12448	11.44477	1.45442	13.14366
Breastcancer(*FSR* = 35%)	21.69424	3.43820	6.97013	0.15359	6.80362	0.16570	11.16776	0.23459	14.06029
Breastcancer(*FSR* = 75%)	22.61477	3.73363	7.37460	0.12935	7.12737	0.30221	13.62568	2.85897	15.7608
penglung(*FSR* = 25%)	25.29486	4.88036	6.09828	1.02441	6.12873	0.96073	14.62881	0.21120	20.41094
penglung(*FSR* = 35%)	28.31658	5.66189	9.30828	0.16100	6.05665	0.12252	10.23873	1.09455	23.09220
penglung(*FSR* = 75%)	45.59696	0.57878	6.77914	0.17209	10.82026	0.42493	13.06543	2.62156	36.33404
WavefromEW(*FSR* = 25%)	116.32329	5.37845	68.44219	2.75824	152.32804	31.78592	118.36316	2.63216	137.4521313
WavefromEW(*FSR* = 35%)	291.92540	27.47239	174.42415	4.15874	239.34024	32.81508	305.76980	5.71527	275.93786
WavefromEW(*FSR* = 75%)	466.21312	49.42853	301.67441	12.04559	554.25729	71.33589	568.70329	9.67018	615.09845
KrvskpEW(*FSR* = 25%)	57.34217	1.65468	37.90333	7.45907	38.32896	2.69333	67.74427	11.30166	59.73568
KrvskpEW(*FSR* = 35%)	65.60792	2.79430	39.24947	0.95562	46.72643	6.21508	74.55587	1.58847	68.40007
KrvskpEW(*FSR* = 75%)	132.40292	9.06038	82.98403	2.77601	112.55620	26.94556	159.05937	4.13820	137.61856
orlraws10P(*FSR* = 25%)	179.05408	2.66247	23.21224	0.58643	21.68696	6.68020	46.08215	0.62625	179.88697
orlraws10P(*FSR* = 35%)	352.61945	73.48164	44.22480	9.48878	33.46060	9.17106	100.71165	13.63551	387.12303
orlraws10P(*FSR* = 75%)	539.21165	7.38586	71.37295	2.32801	63.67275	22.67464	138.55907	2.67978	547.47362
CLL_SUB_111(*FSR* = 25%)	205.31698	4.47325	31.42611	0.64652	30.87181	7.77725	61.63629	0.84412	209.46526
CLL_SUB_111(*FSR* = 35%)	458.09101	78.98422	67.18835	1.75753	64.77310	16.85528	128.45785	2.30073	441.10499
CLL_SUB_111(*FSR* = 75%)	604.48001	5.48099	100.79421	13.83071	113.97444	38.56610	176.25583	15.66681	616.84725

Table 12 Chi-square results

Benchmark	Error rate	No. of selected features	Time (seconds/run)
Breastcancer(*FSR* = 25%)	0.05195	2	0.03100
Breastcancer(*FSR* = 35%)	0.03896	3	0.03200
Breastcancer(*FSR* = 75%)	0.03896	7	0.04000
penglung(*FSR* = 25%)	0.50000	81	0.13500
penglung(*FSR%*)	0.37500	114	0.14400
penglung(*FSR%*)	0.37500	244	0.15400
WavefromEW(*FSR%*)	0.26055	10	0.45240
WavefromEW(*FSR%*)	0.18349	14	0.49920
WavefromEW(*FSR%*)	0.23853	30	0.53040
KrvskpEW(*FSR* = 25%)	0.07163	9	0.24600
KrvskpEW(*FSR* = 35%)	0.07736	13	0.26100
KrvskpEW(*FSR* = 75%)	0.08883	27	0.27700
orlraws10P(*FSR* = 25%)	0.54545	2576	5.39700
orlraws10P(*FSR* = 35%)	0.45455	3606	5.46200
orlraws10P(*FSR* = 75%)	0.54545	7728	7.63600
CLL_SUB_111(*FSR* = 25%)	0.76923	2835	8.23300
CLL_SUB_111(*FSR* = 35%)	0.76923	3969	8.45100
CLL_SUB_111(*FSR* = 75%)	0.76923	8505	9.81300

- *orlraws*10*P* results: For all optimizers, the error rate decreased for all *FSRs* and the minimum error rates achieved when *FSR* was 35%. For *FSR* of 35%, the error rate decreased by 7.96%, the feature reduction ratio was 85.7%, and the running time reduced by 56% of the time needed for applying the wrapper without chi-square.
- *CLL_SUB*_111 results: Almost for all optimizers, the error rate decreased for all *FSRs* and the minimum error rates achieved when *FSR* was 35%. For *FSR* of 35%, the error rate decreased by 1.67%, the feature reduction ratio was 86.4%, and the running time reduced by 58% of the time needed for applying the wrapper without chi-square.

In general, as noted from *orlraws*10*P* and *CLL_SUB*_111 results, we can say that for the large dimensionality datasets, the integration of chi-square filter with wrapper approach gets benefits by reducing the error rate, reducing the number of selected features, and reducing the running time so it is recommended to apply chi-square wrapper with huge datasets. Also, the best *FSR* that can be assigned to K parameter of chi-square filter is 35% for the large dimensionality and the datasets with large number of instances. On the other hand, for small dimensionality datasets such as *breastcancer*, the best *FSR* is 75%. This can be explained that when the number of features is small, we need to select most of the features to reduce the error of classification. Another thing to note is the *penglung* dataset where making any dimensionality reduction using the chi–square filter increased the error rate. This can

be explained by saying that all features of this dataset are significant and relevant (features are not redundant). The last note regarding the BPSO wrapper which is in most of the cases (4 out of 6 datasets) the integration of chi-square increased the error rate for all values of *FSR*.

The results of the last-stage experiments showed that applying the chi-square without wrapper decreased the error rate on small dimensionality datasets and increased the error rate on large dimensionality datasets when compared with wrapper and chi-square wrapper approaches. Also, the running time is smaller than wrapper and chi-square wrapper.

8 Conclusion and Future Works

Recently, metaheuristic algorithms were utilized for optimizing FS problems in different applications. Developing specialized tools geared toward FS has attracted many researchers for several reasons including the automation and acceleration of the FS process, alleviating the implementation effort, minimizing the coding errors, and facilitating the commoners with their applications. Proceeding from here, we intended to develop EvoloPy-FS as a programming foundation that accepts further modifications, additions, and improvements. From the experiments, we conclude the possibility of applying the wrapper and filter–wrapper approaches for large dimensionality datasets and achieving more trustable results (minimum classification errors) compared with applying filter approach only.

Our goals are ambitious to make EvoloPy-FS a full-scale metaheuristics tool used to address FS by a broader community. For the future directions, we intend to make a thorough revision for the code and make a significant expansion for the framework components, namely: the benchmark functions, the evaluation criteria, the datasets, and the experiments. By the same token, we plan to increase the number of the implemented optimizers and study the impact of transfer functions on the FS process.

References

1. Guyon I, Elisseeff A (2003) An introduction to variable and feature selection. J Mach Learn Res 3(Mar):1157–1182
2. Huan L, Hiroshi M (eds) (2007) Computational methods of feature selection. CRC Press
3. Zhao, Z, Liu H (2007) Spectral feature selection for supervised and unsupervised learning. In: Proceedings of the 24th international conference on Machine learning. ACM
4. Huan L, Yu L (2005) Toward integrating feature selection algorithms for classification and clustering. IEEE Trans Knowl Data Eng 17(4):491–502
5. Manoranjan D, Huan L (1997) Feature selection for classification. Intell Data Anal 1(3):131–156
6. Hou C et al (2014) Joint embedding learning and sparse regression: a framework for unsupervised feature selection. IEEE Trans Cybern 44(6):793–804

7. Celeux G et al (2011) A framework for feature selection in clustering. J Am Stat Assoc 105:713–726. J Am Stat Assoc 106(493)
8. Zhao Z et al (2010) Advancing feature selection research. ASU Featur Sel Repos 2010:1–28
9. Li J et al (2017) Feature selection: a data perspective. ACM Computing Surveys (CSUR) 50(6):94
10. Ramrez-Gallego S et al (2018) An information theory-based feature selection framework for big data under apache spark. IEEE Trans Syst, Man, Cybern: Syst 48(9):1441–1453
11. Verénica B-C, Noelia S-M, Amparo A-B (2015) Recent advances and emerging challenges of feature selection in the context of big data. Knowl-Based Syst 86:33–45
12. Liu, H, Motoda H (2012) Feature selection for knowledge discovery and data mining vol 454. Springer Science and Business Media
13. Kohavi R, John GH (1997) Wrappers for feature subset selection. Artif Intell 97(1–2):273–324
14. Abe S (2010) Feature selection and extraction. Support vector machines for pattern classification. Springer, London, pp 331–341
15. Molina LC, Belanche L, Nebot À (2002) Feature selection algorithms: a survey and experimental evaluation. Data mining, 2002. ICDM 2003. 2002 IEEE international conference on. Proceedings. IEEE
16. Yong L, Feng T, Zhiyong Z (2015) Feature selection based on dependency margin. IEEE Trans Cybern 45(6):1209–1221
17. Ensan F, Bagheri E, Gašević D (2012) Evolutionary search-based test generation for software product line feature models. In: International conference on advanced information systems engineering. Springer, Berlin, Heidelberg
18. Yusta SC (2009) Different metaheuristic strategies to solve the feature selection problem. Pattern Recognit Lett 30(5):525–534
19. Yang X-S (2013) Metaheuristic optimization: nature-inspired algorithms and applications.In: Artificial intelligence, evolutionary computing and metaheuristics. Springer, Berlin, Heidelberg, pp 405–420
20. Holland JH (1992) Genetic algorithms. Sci Am 267(1):66–73
21. Koza JR (1992) Genetic programming II, automatic discovery of reusable subprograms. MIT Press, Cambridge, MA
22. Kennedy J (2006) Swarm intelligence. Handbook of nature-inspired and innovative computing. Springer, Boston, MA, pp 187–219
23. Eberhart R, Kennedy J (2011) 'A new optimizer using particle swarm theory. Micro Machine and Human Science, 1995. MHS'95. In: Proceedings of the sixth international symposium on. IEEE, 1995
24. Dorigo M, Birattari M (2011) Ant colony optimization. Encyclopedia of machine learning. Springer, Boston, MA, pp 36–39
25. Xin-She Y, Suash D (2009) Cuckoo search via Lévy flights. In: World Congress on nature and biologically inspired computing (2009) NaBIC 2009. IEEE, p 2009
26. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69(2014):46–61
27. Mirjalili S, Mirjalili SM, Hatamlou A (2016) Multi-verse optimizer: a nature-inspired algorithm for global optimization. Neural Comput Appl 27(2):495–513
28. Mirjalili S (2015) Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm. Knowl-Based Syst 89:228–249
29. Mirjalili S, Andrew L (2016) The whale optimization algorithm. Adv Eng Softw 95:51–67
30. Xin-She Y (2010) A new metaheuristic bat-inspired algorithm. Nature inspired cooperative strategies for optimization (NICSO) Springer vol. 2010. Berlin, Heidelberg, pp 65–74
31. Xin-She Y (2010) Firefly algorithm, Levy flights and global optimization. Research and development in intelligent systems XXVI. Springer, London, pp 209–218
32. Mafarja M, Aljarah I, Faris H, Hammouri AI, Al-Zoubi AM, Mirjalili S (2019) Binary grasshopper optimisation algorithm approaches for feature selection problems. Expert Syst Appl 117:267–286
33. Ahmed S, Mafarja M, Faris H, Aljarah I (2018) Feature selection using salp swarm algorithm with chaos. In: Proceedings of the 2nd international conference on intelligent systems, metaheuristics, and swarm intelligence ACM, pp 65–69

34. Faris H, Al-Zoubi AM, Heidari AA, Aljarah I, Mafarja M, Hassonah MA, Fujita H (2019) An intelligent system for spam detection and identification of the most relevant features based on evolutionary random weight networks. Inf Fusion 48:67–83
35. Mafarja M, Aljarah I, Heidari AA, Faris H, Fournier-Viger P, Li X, Mirjalili S (2018) Binary dragonfly optimization for feature selection using time-varying transfer functions. Knowl-Based Syst 161:185–204
36. Aljarah I, Mafarja M, Heidari AA, Faris H, Zhang Y, Mirjalili S (2018) Asynchronous accelerating multi-leader salp chains for feature selection. Appl Soft Comput 71:964–979
37. Mafarja M, Aljarah I, Heidari AA, Hammouri AI, Faris H, Al-Zoubi AM, Mirjalili S (2018) Evolutionary population dynamics and grasshopper optimization approaches for feature selection problems. Knowl-Based Syst 145:25–45
38. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82
39. Wall M (1996) GAlib: A C++ library of genetic algorithm components. Mech Eng Dep Mass Inst Technol 87:54
40. Keijzer M et al (2001) Evolving objects: A general purpose evolutionary computation library. in: International conference on artificial evolution (Evolution Artificielle). Springer, Berlin, Heidelberg
41. Emmerich M, Hosenberg R (2001) TEA-a C++ library for the design of evolutionary algorithms. Universitätsbibliothek Dortmund
42. Harder R (2001) OpenTS: an open source java tabu search framework. INFORMS Annual Meeting, Miami
43. Bleuler S et al (2003) PISA-a platform and programming language independent interface for search algorithms. iN: International conference on evolutionary multi-criterion optimization. Springer, Berlin, Heidelberg
44. Cahon S, Melab N, Talbi E-G (2004) Paradiseo: a framework for the reusable design of parallel and distributed metaheuristics. J Heuristics 10(3):357–380
45. Wagner S, Affenzeller M (2005) Heuristiclab: a generic and extensible optimization environment. Adaptive and natural computing algorithms. Springer, Vienna, pp 538–541
46. Streichert F, Ulmer H (2005) JavaEvA-a java framework for evolutionary algorithms. In: Center for Bioinformatics Tübingen, University of Tübingen, Technical Report WSI-2005-06
47. Li Y, Yu S-M (2006) A unified optimization framework for real world problems. Lect Ser Comput Comput Sci 7:816–819
48. Pohlheim H (2007) Geatbx: genetic and evolutionary algorithm toolbox for use with matlab. H. Pohlheim, Berlin http://www.geatbx.com
49. Pampara, G, Engelbrecht AP, Cloete T (2008) Cilib: a collaborative framework for computational intelligence algorithms-part I. In: IEEE International joint conference on neural networks, 2008. IJCNN 2008. (IEEE World Congress on Computational Intelligence). IEEE
50. Ventura S et al (2008) JCLEC: a Java framework for evolutionary computation. Soft Comput 12(4):381–392
51. Perone CS (2009) Pyevolve: a Python open-source framework for genetic algorithms. Acm Sigevolution 4(1):12–20
52. Kronfeld M, Planatscher H, Zell A (2010) The EvA2 optimization framework. In: International Conference on Learning and Intelligent Optimization. Springer, Berlin, Heidelberg
53. Durillo JJ, Nebro AJ (2011) JMetal: a Java framework for multi-objective optimization. Adv Eng Softw 42(10):760–771
54. Weppenaar DVI, Vermaak HJ (2011) Solving planning problems with drools planner a tutorial. Interim: Interdiscip J 10(1):91–109
55. Lukasiewycz M et al (2011) Opt4J: a modular framework for meta-heuristic optimization. In: Proceedings of the 13th annual conference on Genetic and evolutionary computation. ACM
56. Fortin F-A et al (2012) DEAP: evolutionary algorithms made easy. J Mach Learn Res 13(Jul):2171–2175
57. Izzo, D (2012) Pygmo and pykep: open source tools for massively parallel optimization in astrodynamics (the case of interplanetary trajectory optimization). In: Proceedings of the fifth international conference on astrodynamics tools and techniques, ICATT

58. Luke, S (2017) 'ECJ then and now. In: Proceedings of the genetic and evolutionary computation conference companion. ACM
59. Tian Y et al (2017) PlatEMO: a MATLAB platform for evolutionary multi-objective optimization [educational forum]. IEEE Comput Intell Mag 12(4):73–87
60. Kohavi R et al (1994) MLC++: a machine learning library in C++. In: Proceedings sixth international conference on tools with artificial intelligence. TAI 94. IEEE
61. Witten IH et al (1999) Weka: practical machine learning tools and techniques with Java implementations
62. Hanke M et al (2009) PyMVPA: a python toolbox for multivariate pattern analysis of fMRI data. Neuroinformatics 7(1):37–53
63. Kachel A et al (2010) Infosel++: information based feature selection c++ library. In: International conference on artificial intelligence and soft computing. Springer, Berlin, Heidelberg
64. Schaul T et al (2010) PyBrain. J Mach Learn Res 11(Feb):743–746
65. Alcalá-Fdez J et al (2011) Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. J Mult-Valued Log Soft Comput 17
66. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O ... Vanderplas J (2011) Scikit-learn: machine learning in Python. J Mach Learn Res 12(Oct):2825–2830
67. Albanese D et al (2012) mlpy: Machine learning python. arXiv preprint arXiv:1202.6548 (2012)
68. Curtin RR et al (2013) MLPACK: a scalable C++ machine learning library. J Mach Learn Res 14(Mar): 801–805
69. Demar J et al (2013) Orange: data mining toolbox in Python. J Mach Learn Res 14(1):2349–2353
70. Thüm T et al (2014) FeatureIDE: an extensible framework for feature-oriented software development. Sci Comput Program 79:70–85
71. Soufan O et al (2015) DWFS: a wrapper feature selection tool based on a parallel genetic algorithm. PloS one 10(2):e0117988
72. Roffo G (2016) Feature selection library (MATLAB toolbox). arXiv preprint arXiv:1607.01327 (2016)
73. van Rossum G (1990–2004) Python programming language
74. Faris H, Aljarah I, Mirjalili S, Castillo PA, Guervés JJM (2016) EvoloPy: an open-source nature-inspired optimization framework in python. In: IJCCI (ECTA), pp 171–177
75. Faris H, Aljarah I, Al-Madi N, Mirjalili S (2016) Optimizing the learning process of feedforward neural networks using lightning search algorithm. Int J Artif Intell Tools 25(06):1650033
76. Faris H, Aljarah I, Mirjalili S (2016) Training feedforward neural networks using multi-verse optimizer for binary classification problems. Appl Intell 45(2):322–332
77. Aljarah I, Faris H, Mirjalili S (2018) Optimizing connection weights in neural networks using the whale optimization algorithm. Soft Comput 22(1):1–15
78. Emary E, Zawbaa HM, Hassanien AE (2016) Binary grey wolf optimization approaches for feature selection. Neurocomputing 172(2016):371–381
79. Xue B, Zhang M, Browne WN (2013) Novel initialisation and updating mechanisms in PSO for feature selection in classification. In: European conference on the applications of evolutionary computation. Springer, Berlin, Heidelberg
80. Chuang L-Y et al (2008) Improved binary PSO for feature selection using gene expression data. Comput Biol Chem 32(1):29–38
81. Huang CL, Dun JF (2008) A distributed PSO-SVM hybrid system with feature selection and parameter optimization. Appl Soft Comput 8:1381–1391
82. Xue B, Zhang M, Browne WN (2012) New fitness functions in binary particle swarm optimisation for feature selection. In: IEEE congress on evolutionary computation (CEC2012) pp 2145–2152
83. Lin W et al (2016) 'An empirical study on the characteristics of Python fine-grained source code change types. In: 2016 IEEE international conference on software maintenance and evolution (ICSME). IEEE

Multi-objective Particle Swarm Optimization: Theory, Literature Review, and Application in Feature Selection for Medical Diagnosis

Maria Habib, Ibrahim Aljarah, Hossam Faris and Seyedali Mirjalili

Abstract Disease prediction has a vital role in health informatics. The early detection of diseases assists in taking preventive steps and more functional treatment. Incorporating intelligent classification models and data analysis methods has intrinsic impact on converting such trivial, row data into worthy useful knowledge. Due to the explosion in computational and medical technologies, we observe an explosion in the volume of health- and medical-related data. Medical datasets are high-dimensional datasets, which make the process of building a classification model that searches for optimal set of features a hard, yet more challenging task. Hence, this chapter introduces a fundamental class of optimization known as the multi-objective evolutionary algorithms (MOEA) for optimization, which handles the feature selection for classification in medical applications. The chapter presents an introduction to multi-objective optimization and their related mathematical models. Furthermore, this chapter investigates the utilization of a well-regarded multi-objective particle swarm optimization (MOPSO) as wrapper-based feature selection method, in order to detect the presence or absence of different types of diseases. Therefore, the performance of MOPSO and its behavior are examined by comparing it with other well-regarded MOEAs on several medical datasets. The experimental results on most of the medical datasets show that the MOPSO algorithm outperforms other algorithms such as non-dominated sorting genetic algorithm (NSGA-II) and multi-objective evolutionary algorithm based on decomposition (MOEA/D) in terms of classification accuracy and minimum number of features.

M. Habib · I. Aljarah · H. Faris
King Abdullah II School for Information Technology, The University of Jordan, Amman, Jordan
e-mail: i.aljarah@ju.edu.jo

H. Faris
e-mail: hossam.faris@ju.edu.jo

S. Mirjalili (✉)
Torrens University Australia, Brisbane, QLD 4006, Australia
e-mail: ali.mirjalili@gmail.com

Griffith University, Brisbane, QLD 4111, Australia

S. Mirjalili et al. (eds.), *Evolutionary Machine Learning Techniques*,
Algorithms for Intelligent Systems, https://doi.org/10.1007/978-981-32-9990-0_9

Keywords Multi-objective optimization · Feature selection · Machine learning · Evolutionary algorithms · Medical applications

1 Introduction

Integrating machine learning techniques into clinical decision support systems (CDSS) for diagnosis considers a fundamental improvement in healthcare and medical informatics [7, 9]. Decision support systems (DSS) depend on computer-based systems that enhance the process of decision making [9]. If we are talking about CDSS or as a sub-term the diagnosis decision support system (DDSS), then we are talking about computational methods that analyze medical data and help in different situations, as in disease detection and treatment [9]. In most cases, medical datasets are very high-dimensional datasets; therefore, developing classification models and feature selection methods is relatively hard problem. Having n number of features means to search for the optimal set of features in a search space of size 2^n, which is NP-hard search problem. NP-hard problems cannot be addressed using the known algorithms in polynomial time. Therefore, the time required for solving them increases exponentially with the growing size of the problem or the number of objectives. Scientists usually use heuristic algorithms in order to solve such hard optimization problems [66]. Metaheuristic algorithm is a sub-category of heuristic algorithms, which are stochastic algorithms that compromise between randomization and local search [66]. However, metaheuristics are well-regarded choice of methods for solving most of hard optimization problems [62].

Computational intelligence (CI) is a sub-field of artificial intelligence. CI covers artificial neural networks, evolutionary computation, and swarm intelligence [24, 28]. Evolutionary computation is a kind of stochastic and metaheuristic optimization techniques that mimic the natural evolution process, inspired by the Darwinian principles in nature. Nature-inspired algorithms could be categorized based on the root of inspiration into evolutionary-based, swarm-based, and physical- or chemical-based [66]. Evolutionary-based algorithms use mechanisms inspired by the biological evolution like reproduction, mutation, and selection. Swarm intelligence algorithms study the collective behavior of interacting agents that follow few simple rules, inspired by the social behavior of animals or insects, such as the ants' foraging behaviors, animal herding, and bird flocking [4, 5, 25, 66].

Feature selection (FS) is an imperative reprocessing step with conflicting objectives. The most common objective is to minimize the number of features, which is often called dimensionality reduction. The second objective is often to minimize the classification or prediction rate for a given dataset. Therefore, we can formulate and treat feature selection problems as multi-objective problems.

Two common methods to optimize problems with multiple objectives are discussed in this paragraph. The first method is called aggregation-based multi-objective optimization, in which we combine multiple objectives in to a single objective using a set of weights. The second method is ε-constraint. The former method suffers from

the difficulty in finding uniformly distributed Pareto optimal solutions and failure in finding non-cover-shaped Pareto optimal fronts as also discussed in [16]. In contrast, the latter solves the problem of non-convex objective space by preserving one objective and restricting the other objectives using user-defined constraints. However, finding a feasible set of solutions depends highly on the initial ε-vector value. Hence, the evolutionary search is used to find a set of optimal solutions simultaneously, and synergistically with different operators, such as the non-domination, the niching, and the elitism [16].

Mainly, MOEAs have two main branches—the dominance-based and thebreak decomposition-based. The dominance-based MOEA utilizes a selection method based on Pareto domination. A popular example is the NSGA-II algorithm [15], which uses non-dominated sorting based on elitism-preserving approach. Thebreak decomposition-based MOEA decomposes multiple objectives into a number of single-objective problems, such as the MOEA/D [67].

This chapter presents an interpretation for using multi-objective particle swarm optimization as wrapper-based feature selection for medical diagnosis and also presents a comparison with other well-regarded multi-objective evolutionarybreak algorithms—the NSGA-II and MOEA/D. MOPSO showed promising results for feature selection for medical data and superior performance over other multi-objective evolutionary algorithms.

The rest of the chapter is organized as follows: Sect. 3 presents the basic principles and formulation for multi-objective problems, and Sect. 4 describes the framework of multi-objective evolutionary optimization and in particular the MOPSO. Section 5 illustrates the formulation of feature selection as multi-objective problem, besides recent works of multi-objective feature selection in medical applications. Section 6 shows the conducted experiments and obtained results. Finally, Sect. 7 is a summary and conclusion of the chapter.

2 Literature Review

The importance of feature selection is continuously growing as the number of large datasets is continuously increasing. Developing efficient computational techniques to distinguish relevant features is at significant level, yet challenging. However, an increasing number of research studies have been conducted for approaching the feature selection problem. Broadly speaking, various metaheuristic algorithms have been utilized for searching for the optimal subset of features, such as the adoption of salp swarm optimization [2, 6, 27], dragonfly optimization [45, 47], grasshopper optimization [3, 44, 46], multi-verse optimizer [26], and ant lion optimizer [48].

The medical datasets, normally, contain hundreds or thousands of features; hence, several studies have been introduced the use of multi-objective evolutionary algorithms for feature selection in medical frameworks. For instance, in [1] the authors

present the utilization of evolutionary artificial neural networks based on the Pareto differential evolution for the prediction of breast cancer. Other studies devoted for the prediction of breast cancer, as in [18] where they implemented multi-objective evolutionary approach based on multi-expression programming. In [34], the authors have implemented multi-objective classifier based on multilayer perceptron (MLP) and differential evolution, in which the multi-objective differential evolution is used for optimizing the MLP.

Further, in [55], a multi-objective genetic programming has been designed to optimize the decision trees and find the optimal decision tree models. The aim of decision trees is predicting if the status of a diabetic patient deteriorates. In [53], a multi-objective evolutionary algorithm based on NSGA-II is used for biclustering gene expression data. Therefore, gene expression has a substantial relation between genotypes and phenotypes. Further, in [63], the authors utilized both MOPSO and NSGA-II for identifying the impressive features for Warfarin dose prediction, where MOPSO achieved superior performance. Nonetheless, in [37] a multi-objective neural network is used for feature selection in the diagnosis of prostatic cancer.

In [61], an adaptive fuzzy and chaotic multi-objective PSO has been implemented for selecting the optimal set of features (genes), form microarray and RNA-sequencing gene expression datasets, in which the proposed algorithm achieved remarkable results in terms of accuracy, sensitivity, and specificity. Interestingly, in [42], Li and Yin implemented a multi-objective binary biogeography-based optimization (MOBBBO) algorithm for feature selection of gene expression data. The MOBBBO as wrapper-based FS was based on support vector machine and leave-one-out cross-validation. They examined the performance of MOBBBO on ten gene expression datasets, covering various diseases, as lung tumor, brain tumor, and prostate tumor, in which MOBBBO outperformed NSGA-II algorithm.

Furthermore, in [58] two multi-objective gray wolf optimizer (MOGWO) approaches have been proposed for cervical cancer detection. One of the methods is based on a scalarized technique, and the other is based on non-dominated sorting (NSGWO). NSGWO showed superior results as wrapper-based feature selection in comparison with NSGA-II, multi-objective firefly algorithm, and other MOEAs. As microarray is an important technology for the diagnosis of human diseases, in [22] the authors conducted comparison study among several MOEAs for feature selection of cancer microarray datasets, where they used the Leukemia, Colon, and Lymphoma datasets to investigate the performance of NSGA-II, SPEA2, multi-objective cross-generational elitist selection (MOCell), and heterogeneous recombination and cataclysmic mutation (MOCHC), in which MOCHC has advantages over the other MOEAs.

Generally, there are several research areas in medical informatics, where researchers applied evolutionary multi-objective algorithms, not just for disease diagnosis, but, moreover, for drug and pharmacological development [19].

3 Multi-objective Optimization

Multi-objective problems (MOPs) are type of optimization problems that at least two objective functions to be optimized, simultaneously [16]. As the objectives are in conflict to each other, there is no optimal solution for MOPs, but a set of trade-off solutions that called Pareto optimal set. MOPs, in literature, are also known as multi-criteria or multi-attribute optimization [66]. Mathematically, any optimization problem with constraints can be defined as follows in (Eqs. 1, 2, and 3) [16]:

$$Optimize \;\; F(X) = [f_1(X), \;\; f_2(X), \ldots, f_k(X)] \tag{1}$$

where X represents an n-dimensional vector of decision variables that belongs to the feasible solution space Ω, and defined as $X = x_1, \ldots, x_n$; k is the number of objective functions, if $k = 1$; then it is a single optimization problem, whereas having $k > 3$ objectives; then it is a many-objective optimization problem. An extreme case, when $k = 0$ that is called a feasible problem, since any feasible solution in the solution space is an optimal solution. Indeed, by supposing that $F(X)$ is a minimization problem, then the constraints are subject to:

$$g_m(x) \leq 0, \;\; m = 1, \;\; 2, \ldots M \tag{2}$$

$$h_j(x) = 0, \;\; j = 1, \;\; 2, \ldots J \tag{3}$$

where $g_m(x)$ is the mth inequality constraint and $h_j(x)$ is the jth equality constraint. The above equations allow having problems with zero or more constraints. In case of no constraints, the problem is called unconstrained optimization problem. In case of at least one constraint including equality or non-equality, we deal with constraint optimization problem. Depending on the equations used for objectives and constraints, the problem is linear or nonlinear [66].

Essentially, the main idea is how to find the optimal solution among a set of potential optimal solutions. In other words, consider a single-objective optimization problem that has two solutions x_1 and x_2. We can find the optimal solution by a simple comparison. Hence, x_1 is better than x_2, if and only if $f(x_1) > f(x_2)$. On the other hand, in the case of MOPs, the comparison among solutions is more complex, since we are dealing with vector-valued solutions. Hence, the adoption of *Pareto Dominance* relationship is essential for comparing the vectors. Suppose that we have a minimization problem and two vector solutions; $\mathbf{x_1}$ and $\mathbf{x_2}$. we say that $\mathbf{x_1}$ dominates $\mathbf{x_2}$ denoted by ($\mathbf{x_1} \preceq \mathbf{x_2}$) if f($\mathbf{x_1}$) dominates f($\mathbf{x_2}$), which means all the components of f($\mathbf{x_1}$) are not worse than the corresponding elements of f($\mathbf{x_2}$) and is better in at least one. *Pareto Dominance* relationship is represented by the following equation:

$$\forall i : f_i(\mathbf{x_1}) \leq f_i(\mathbf{x_2}) \quad and \quad \exists j : f_j(\mathbf{x_1}) < f_j(\mathbf{x_2})$$
$$where \ i \ and \ j \in \{1, 2, 3, \ldots, k\}$$

In other words, if $\mathbf{x_3}$ is better in $objective_1$ and worse in $objective_2$, and $\mathbf{x_4}$ is better in $objective_2$ and worse in $objective_1$, then both $\mathbf{x_3}$ and $\mathbf{x_4}$ are non-dominated solutions and denoted by the Pareto optimal set as $P^* = \{\mathbf{x^*} \in \Omega\}$. The image of the optimal Pareto set in the objective space called Pareto front, denoted as $PF^* = \{f(\mathbf{x^*}) \mid \mathbf{x^*} \in P\}$ [16]. Figure 1 illustrates the presentation of MOP and clarifies the dominance concept. The figure shows an optimization function F maps feasible solutions from the decision space into the objective space; in addition, it shows how those feasible solutions might be non-dominated solutions, dominated solutions, or optimal solutions on the Pareto front. In the this figure, neither *Solution a* nor *Solution b* is a better solution, even so, both are better than *Solution c*.

Particularly, solving MOPs requires finding the set of Pareto optimal solutions. But, there is the next step of applying the multi-criterion decision making for selecting the preferred solution among the optimal trade-off solutions. Therefore, choosing a single preferred solution is important as well as the importance of finding a representative Pareto optimal set of solutions.

There are three main techniques for a decision-maker to select a solution. The a priori, a posteriori, and the interactive approach (progressive). In the a priori approach, the preference information is predefined and is used to direct the search into one

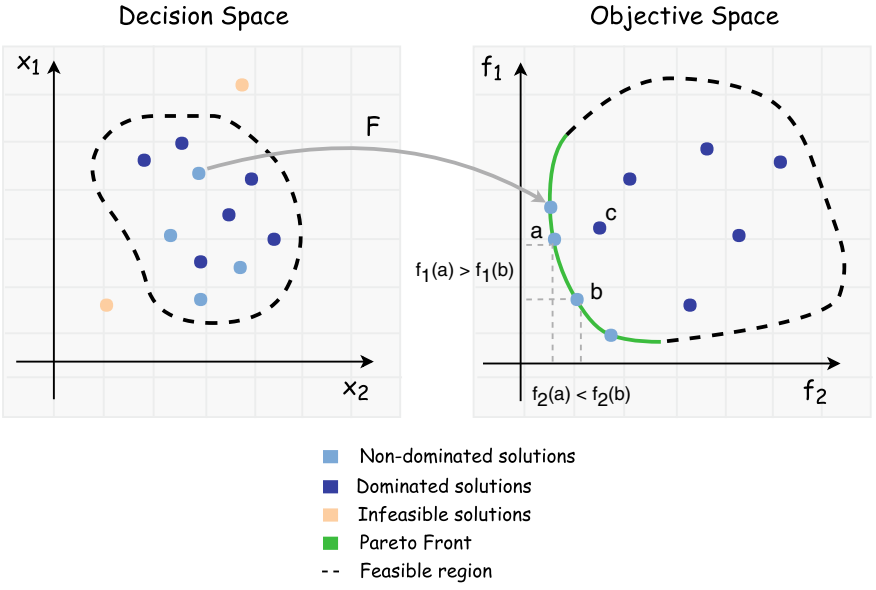


Fig. 1 Representation of multi-objective optimization problem and dominance relation, where both f_1 and f_2 are minimization problems

part of the Pareto optimal front based on two different criteria, the aggregation-based ordering and the aggregation-based scalarization, while the a posteriori approach uses the preference information after the set of Pareto optimal solutions is found, relying on one of three criteria, either the independent sampling, the cooperative search, or the hybrid selection. This approach is computationally expensive and suffers from difficulties in converging to the Pareto optimal front as the number of objectives increases. The interactive approach integrates the preference information with the multi-objective optimization algorithm during the optimization run, in which, after a number of generations, and iteratively, the decision-makers should rank the best-found non-dominated solutions according to their preference [16].

4 Multi-objective Evolutionary Optimization

Multi-objective evolutionary optimization is a combination of evolutionary computation and multi-criteria decision making. Evolutionary algorithms (EAs) or swarm-based algorithms are metaheuristic or heuristic algorithms used to solve optimization problems. Computationally, intelligent algorithms proved successful abilities in finding global optimal solutions in case of high-dimensional search spaces [66], as well as in non-differentiable functions and multi-objective problems [13].

Basically, EAs during the search for optimal solutions depend on a population of solutions instead of searching using single solution at a time. The roots of evolutionary algorithms return back to 1960s and 1970s, when the basic EAs' principles and concepts were established [33]. JH Holland created the basic framework for the evolutionary algorithms which based on Darwin's theory of evolution and the survival for the fittest. EAs are stochastic algorithms which rely on initial population of solutions and evolve to the best solutions over generations.

Mainly, EAs contain three basic operations: First is the design of initial population and control parameters, second is the design of a fitness function that assesses the quality of potential solutions, and third is the design of genetic operators. The use of genetic operators (as crossover and mutation) enhances the process of information exchange among solutions and guides the algorithm to converge toward the optimal solutions [33]. Thereafter, the EA algorithm continues iterating over the aforementioned operations until a stopping criterion is satisfied. Similarly, the swarm-based algorithms start with initial population, involve a fitness function, a stopping criteria, and instead of the genetic operators, the swarm-based algorithms use update and move agents strategies by means of global and local best solutions in order to guide the population toward the global optimum [66].

Conventionally, EAs have been applied to solve single-objective optimization problems. However, practically, several real-world problems are more complex and can be modeled as problems with multiple optimizing objectives. Let us take an example from the medical field, the problem of medical images reconstruction, in which the aim is to minimize the error between the original image and the reconstructed image, and at the same time, the need of the reconstructed image to be smooth and

noiseless. It is difficult using single-objective optimization algorithms to obtain a solution that satisfies both conflicting objectives simultaneously. Accordingly, since EAs are population-based algorithms, they can handle more properly the nature of multi-objective optimization problems [13].

Basically, different attempts have been initiated to solve multi-objective problems, such as optimizing the highest priority objective or converting the MOPs into single-objective problems using weighted-sum techniques [13]. Primarily, the first attempt to solve MOPs using evolutionary algorithms was by David Schaffer [60], in mid-1980s, when he designed a vector-valued genetic algorithm (VEGA) that divides the population into a number of sub-populations equal to the number of objectives, each sub-population assigned a different fitness function relating to the objective function, and then the sub-populations shuffled together for evolving the next generation [60]. In spite of that VEGA is simple to implement, but it was lacking the diversity of solutions. Afterward, Goldberg suggested the use of Pareto dominance to solve MOPs [57]. Until nearly 1995, different MOEAs have been proposed, such as multi-objective genetic algorithm (MOGA) [56], a niched Pareto genetic algorithm for multi-objective optimization [57], vector-optimized evolution strategy (VOES) [41], and other variants, which all form the first generation of MOEAs.

Since then, new MOEAs started to adopt elitism techniques, forming a second generation of evolutionary multi-objective optimization. A salient exemplar is the NSGA algorithm [64], which proposed by Srinivas and Deb, in which NSGA utilizes ranking, niching, and fitness sharing techniques to maintain the diversity of solutions. Other examples are as Pareto archived evolution strategy (PAES) [36] and strength Pareto evolutionary algorithm (SPEA) [14]. Afterward, the decomposition approach has been introduced with the development of multi-objective evolutionary algorithm based on decomposition (MOEA/D) [67], which stands mainly on decomposing the problem into a number of single-objective sub-problems.

The developments in MOEA continue; hence, different algorithms have been generated, such as MOPSO [23], multi-objective gray wolf optimizer [51], grasshopper algorithm for multi-objective optimization [52], and the multi-objective ant lion optimizer [49].

Generally, as the field of MOEAs has been matured, indeed, for any algorithm approaching MOPs requires solving several crucial issues [13];

- How to obtain and store the non-dominated solutions during algorithm iterations.
- How to guide the current found Pareto front ($PF_{current}$) into the true Pareto front (PF_{true}). As illustrated in Fig. 2a, in which the algorithm guides the particles to converge toward the true Pareto front.
- How to preserve diversity of solutions on the Pareto front, as well as not losing the good non-dominated solutions during archiving. Sub-figures (b) and (c) show how an algorithm should result in well-diversified solutions.
- Selecting limited number of non-dominated Pareto front solutions and delivering them for the decision-maker.

Accordingly, a model MOEA makes use of Pareto domination approaches to find a set of non-dominated solutions, use external archive or repository to store non-

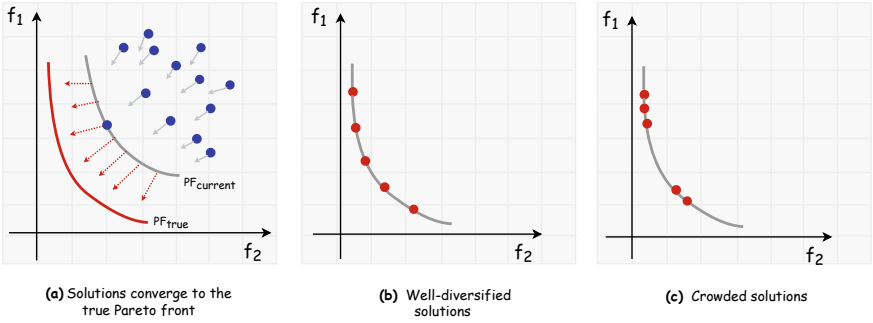


Fig. 2 A description of typical MOEA converges to the optimal true Pareto front. f_1 and f_2 are minimization objectives. In **a** during the search process solutions are guided into optimal Pareto front, however, it may not be the optimal true Pareto front. **b** illustrates a well distributed solutions, whereas **c** shows how the found solutions are crowded at certain region of the Pareto front

dominated solutions, utilize different techniques as niching for maintaining diversity, in addition to apply archiving and elitism strategies for speeding up convergence. This chapter will present an example of multi-objective evolutionary algorithms which is the multi-objective particle swarm optimization.

4.1 Multi-objective Particle Swarm Optimization (MOPSO)

Particle swarm optimization (PSO) was proposed by Kennedy and Eberhart in 1995 [23]. PSO imitates the swarm social behavior of flocks of birds of school of fishes. The objective of PSO is to find the optimal solution within the search space of an objective function, likewise a swarm of birds that search for the best source of food. In PSO, a set of randomly generated particles search for the best solutions. The particles search by adjusting their flight directions and velocities [23], using Eqs. 4 and 5, respectively.

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \tag{4}$$

$$\begin{aligned} v_{id}(t+1) = w * v_{id}(t) + r_1 * c_1 * [p_{id}(t) - x_{id}(t)] \\ + r_2 * c_2 * [g_d(t) - x_{id}(t)] \end{aligned} \tag{5}$$

where d is the number of dimensions. w is the inertia weight that controls the exploration of a particle, r_1 *and* r_2 are random numbers $\in [0, 1]$, c_1 *and* c_2 are acceleration constants that used to control the effect of personal and global best particles; p_{id} shows the personal best position for a particle (*pbest*), and g_d indicates the global best position found by neighbors (*gbest*).

Certainly, PSO shows high-speed convergence ability in single-objective problems, that is made it desirable choice for MOPs. The design of MOPSO utilizes the Pareto dominance to generate a set of leaders that control the flight direction of particles and guide the search process for optimality. Furthermore, it stores the founded non-dominated solutions in external global memory (called repository), which used later by particles as global leaders. The global guide is chosen using roulette wheel selection based on hypercubes' score. Further, MOPSO adopts a geographically based strategy in order to maintain diversity of solutions.

Primarily, the external repository "archive" contains two parts—the controller and the grid. The purpose of the controller is to make decision about new solution if it should be added to the archive or not; updating or pruning the archive relies on the dominance relation. However, whenever the archive is full, an adaptive grid procedure is invoked. In contrast, the grid is used to promote the diversity among solutions. Essentially, the objective space is divided into regions called hypercubes. The hypercubes are geographical regions which consist of a number of solutions, which is created in regard to the objective functions. Each hypercube assigned a fitness value based on the number of particles it contains. Thus, the hypercubes with high number of particles have low fitness value. Figure 3 shows an example of hypercube with crowded number of solutions. The roulette wheel selection technique is used to select a hypercube; upon selecting a hypercube, a random particle is chosen from it. Hence, the grid facilitates the process of selecting the solutions that are located in less populated regions in the objective space than the ones in the crowded areas.

The following steps summarize the process of MOPSO algorithm [12].

1. Initialize the population POP_i, where i =[1, 2, …,N]; N is the population size.
2. Initialize the velocity of each particle VEL_i.
3. Evaluate each particle and assign it with a fitness value.
4. Store the particles' positions that represent non-dominated solutions in external archive (REP).
5. Create hypercubes, and set the particles using the hypercubes as coordinate system.

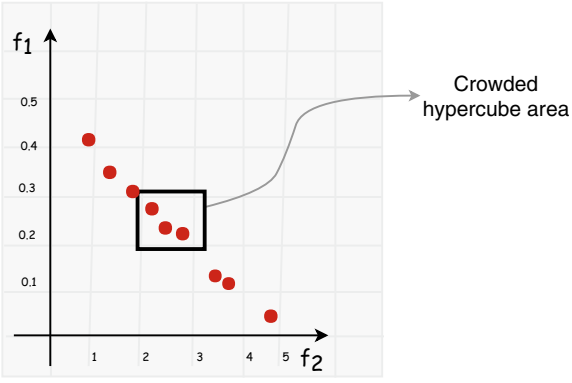


Fig. 3 MOPSO hypercubes' representation, where f_1 corresponds to the error rate and f_2 is the number of features

6. Initialize the memory of each particle, and store the initial positions as the best particles' positions found.
7. Calculate the speed of each particle using Eq. 6.
8. Calculate the new positions of each particle using Eq. 4.
9. Preserve the particles within the search space constraints' boundaries.
10. Evaluate the particles.
11. Update the REP and hypercubes by inserting the new non-dominated solutions, where the dominated solutions removed. However, when the REP is full, the particles on less crowded region assigned a higher priority.
12. Update the memory of personal best positions of each particle if the current position is better than the position in its memory using Pareto dominance.
13. If the stopping condition is met, then stop, otherwise go back to step 7.

$$\begin{aligned} v_{id}(t+1) = w * v_{id}(t) + r_1 * c_1 * (p_{id}(t) - x_{id}(t)) \\ + r_2 * c_2 * (REP(h) - x_{id}(t)) \end{aligned} \tag{6}$$

In Eq. 6, the $REP(h)$ is a non-dominated solution to be selected from the repository, in which the index h is chosen based on hypercubes' fitness value.

4.2 Binary MOPSO

Originally, PSO has been designed to deal with continuous variables and real-number spaces [12, 35]. Handling problems with discrete search space requires transforming the continuous search space into binary search space, such examples with discrete search spaces are the scheduling, routing, feature selection, or dimensionality reduction [35]. Particle swarm optimization modifies particles' trajectories based on changes of previous velocities and positions that influenced by the best global and local performances. In discrete PSO problems, velocities are used to indicate the changing of probability that a bit is holding value 0 or 1. In other words, v_{id} represents the probability that x_{id} is equal to 1.

The binary PSO developed by Kenndy and Eberhart in 1997 [35], which mainly differs than the classical PSO by using a transfer function (TF) with a different strategy for updating particles positions. In the binary version, the particles move in binary search space; therefore, the particle position vector x_{id} is an integer value in [0, 1], and the particle velocity vector v_{id} is a probability in the interval [0, 1]. Transfer function (TF) is a mathematical model used to convert continuous search spaces into discrete search spaces [50]. In the original version of binary PSO [35], the authors used the Sigmoid function as transfer function; however, in [50], the author suggested other types of transfer functions such as V-shaped and S-shaped transfer functions for formulating the binary PSO. Taking as an example, the Sigmoid transfer

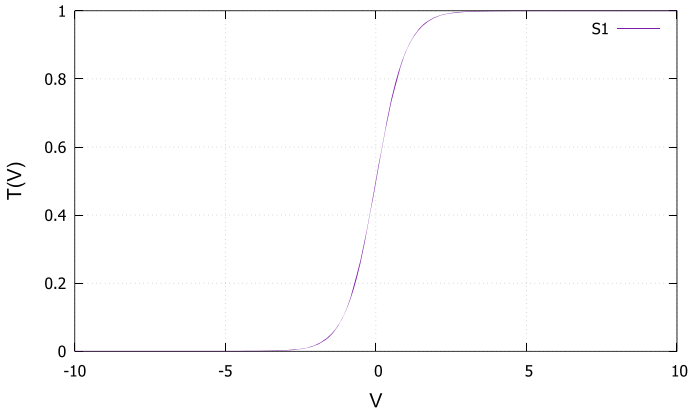


Fig. 4 S1 transfer function

function (S1), which is shown in Eq. 7. The function is sharply increased and reached saturation as the velocity increases, as in Fig. 4.

$$T(v_{id}(t)) = \frac{1}{1 + e^{-2 * v_{id}(t)}} \tag{7}$$

where $v_{id}(t)$ is subject to the $particle_i$ velocity at iteration t and dimension d. Using the probability value produced from the TF of Eq. 7, the position equation will be updated using Eq. 8

$$x_{id}(t+1) = \begin{cases} 1 & \text{if } rand \geq T(v_{id}(t)) \\ 0 & \text{if } rand < T(v_{id}(t)) \end{cases} \tag{8}$$

5 Application of MOPSO in Feature Selection for Medical Diagnosis

Broadly speaking, having irrelevant and redundant features—considering solving a classification problem—deteriorates the classification performance, especially with high-dimensional datasets. Further, having large number of features leads to overfitting and results in weak generalization abilities [10]. For example, the analysis of gene expression data that results from microarray experiments consists of hundreds or thousands of genes, which makes the classification task based on the significantly expressed genes a difficult task.

The objective of feature selection is to address this problem by choosing the relevant features. In single-objective feature selection tasks, the FS has one objective to be optimized. Single-objective FS aims to find the best classification performance regardless the training cost or the number of features. Multi-objective

feature selection (MOFS) handles the FS task by converting it to multi-objective optimization problem, in which the goal is to deal with the optimization of two objectives. The objectives are the number of features and the classification performance. As a result, the solution for multi-objective feature selection optimization problem is a set of non-dominated solutions, where each solution is a vector of two components, the number of features and the classification error rate [10]. Handling the FS problem as a minimization problem, then, the objective is to minimize the number of features and to minimize the classification error rate $(1 - classification\ performance\ (Accuracy))$.

5.1 Problem Formulation

Feature selection problem has been handled as multi-objective problem by using multi-objective particle swarm optimizer. The objectives are to minimize both the number of features and the classification error rate. Accordingly, the number of features of each dataset, independently, represents the problem dimension, with binary search space in the interval [0, 1]. Since MOPSO is population-based metaheuristic, the initial population of the swarm corresponds to the potential subsets of features. In Fig. 1, over the decision space, x_1 is a positive real number which represents the error rate and x_2 is also a positive real number which represents the number of features, and the function F results in a set of trade-off decision vectors represented by $[F(x_1), F(x_2)]$. For instance, one decision vector might be $v_1 = [0.001, 12]$, and another vector is $v_2 = [0.05, 5]$. v_1 is a satisfying solution if the decision-maker prefers having the lowest error rate regardless the number of features. In contrast, v_2 is a satisfying solution for a decision-maker concerns on having the minimum number of features. Hence, both v_1 and v_2 are acceptable solutions, but, which objective to sacrifice returns to the decision-maker preference.

5.1.1 Solution Encoding

Simply, each particle in the swarm represents the selected features by value 1; thus, each particle is a binary-valued vector, in which each element refers to a feature in the dataset. Hence, the length of a particle equals the number of features in the corresponding dataset. Figure 5 shows an example of the representation of a particle, which represents the number of features in the dataset, where the number of features equals 30. If a feature is selected, then value 1 in the position i denotes that the feature i is selected, while value 0 denotes that feature i is not selected. Consequently, from Fig. 5 we notice that the selected subset of features contains the following set of features $\{f_2, f_3, f_6, f_{30}\}$.

The Sigmoid (S1) transfer function is utilized in order to define the probability for choosing a feature or not choosing it. If the random probability is less than the value of the transfer function, then the feature is assigned a value zero, otherwise it assigned one, as illustrated in Eq. 8.

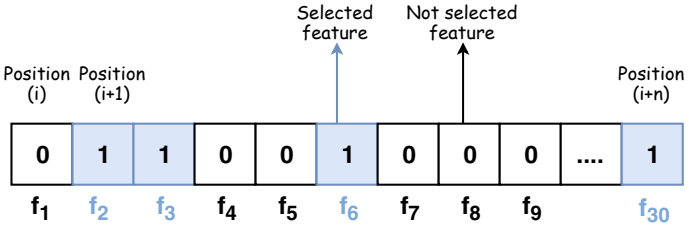


Fig. 5 A particle representation of a sample dataset, where the number of features (n) is equal to 30

5.1.2 Fitness Formulation and Evaluation

The evaluation of the selected subsets of features for MOPSO is carried out using two main objectives—the classification error rate and the number of features. The fitness function is formulated as in Eq. 9:

$$Minimize\ F(x) = \begin{cases} f_1(x) = \frac{l}{A} & , \ l \in A,\ A \in \mathbb{R}^+ \\ f_2(x) = \frac{FP+FN}{P+N} * 100\% & , \ (P+N) \in \mathbb{R}^+ \end{cases} \tag{9}$$

l is the selected number of features. A is the overall number of features. P is equal to TP+FN, and N is equal to FP+TN. The first objective function $f_1(x)$ is subject to the selected features ratio, while the second objective $f_2(x)$ corresponds to classification error rate. FP, FN, TP, and TN are corresponding to false positives, false negatives, true positives, and true negatives, respectively [31].

The well-known K-nearest neighbor (K-NN) is used to evaluate the performance of the optimizers with k = 5. Choosing the value of k to be 5 is based on a previous research study in [54], where the authors compared the performance of particle swarm optimizer with different k values for K-NN algorithm. The classification error of 5-NN is determined using fivefold cross-validation, in which the classification errors were averaged to get the final classification error of each candidate solution. Certainly, in K-fold cross-validation, the choice of k value is usually 5 or 10; there is no conventional rule [38]. The datasets are divided into 30% for testing and 70% for training [11].

The selected subset of features is evaluated on the testing set (30%), using 5-NN, in order to have the final averaged testing classification error rate. Figure 6 illustrates the overall methodology.

5.2 *Datasets' Description*

The used datasets are benchmark datasets available at University of California at Irvine (UCI) machine learning repository [17]. Two of those datasets are gene expres-

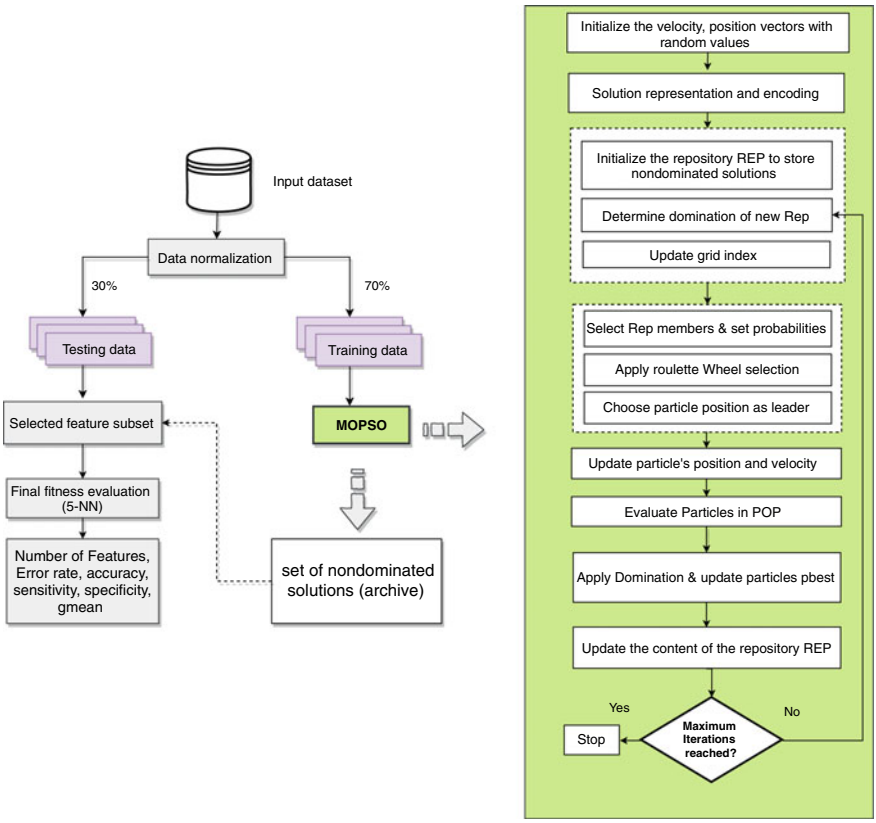


Fig. 6 Methodology design

sion datasets (the Leukemia and Colon cancer datasets) which were taken from two published papers [8, 30]. All datasets were normalized by using Min-Max method in the range from 0 to 1 [31]. Table 1 presents a summary of the datasets, the number of instances, features, classes, and class ratios. Furthermore, a brief description about the datasets will be presented in the following paragraphs:

Wisconsin breast cancer database (WBCD): WBCD dataset was obtained by Dr. Wolberg and Mangasarian from the University of Wisconsin Hospital. WBCD dataset contains 699 breast cancer patient records with nine collected features. It contains mainly two classes—the benign and malignant. The features are like the cell's shape, size, uniformity, and other features as described in [17, 20].

Colon dataset: The Colon cancer dataset is a well-known dataset in the literature [21, 29]. The deoxyribonucleic acid (DNA) microarray data aids scientists in identifying functional roles of genes and thus helps in disease characterization and diagnosis. Cancer tissues have distinctive characteristics of gene expression in compared to normal tissues. The Colon dataset is gene expression dataset for colon

Table 1 Summary of used datasets

#	Dataset	No. of features	No. of instances	No. Classes	Classes ratios (%)
1	WBCD	30	569	2	[62.7, 37.3]
2	Colon cancer	2000	62	2	[64.5, 35.5]
3	Heart	13	270	2	[55.5, 44.4]
4	Leukemia	7129	72	2	[65.3, 34.7]
5	Diabetes	8	768	2	[65.1, 34.9]
6	Acute inflammation	6	120	2	[49.2, 50.8]
7	Hepatitis	10	155	2	[79.4, 20.6]
8	Liver disorders	6	345	2	[58, 42]
9	Parkinson	22	195	2	[24.6, 75.4]
10	Planning- relax	12	182	2	[28.6, 71.4]
11	SPECTF heart	44	267	2	[79.4, 20.6]
12	Vertebral	6	310	2	[32.3, 67.7]

tumor, which contains 40 colon tumor samples and 22 normal colon tissues. By processing that data, it resulted into two classes; cancerous or noncancerous, 6200 gene expression values, and 62 records of data [8].

Heart disease: The dataset is for heart disease diagnoses, collected from Cleveland Clinic Foundation, Hungarian Institute of Cardiology, University Hospital of both Zurich and Basel in Switzerland. The whole database created in two versions—one with 75 features and the other one is with 14 features. The aim is to predict the presence or absence of heart disease. The dataset contains 270 patient records comprising 14 features. The features are mostly related characteristics for blood and heart, as were described in [17].

Leukemia dataset: The aim of this dataset is leukemia classification using gene expression data based on DNA microarray. Leukemia is type of cancer that starts in blood-forming tissues. The dataset contains two output classes—the acute myeloid leukemia and acute lymphoblastic leukemia. The dataset created from 38 bone marrow samples is collected from acute leukemia patients. The ribonucleic acid (RNA) is extracted from the bone marrow samples and processed through several steps resulting in 7129 gene expression values. Hence, the features of the dataset correspond to the normalized log for (expression level) the all 7129 genes besides 72 records of data [30].

Diabetes: Diabetes dataset is created by Michael Kahn at Washington University as participation in a symposium on artificial intelligence in medicine [65]. The data is collected by automatic electronic recording and paper-based recording, resulting in 70 set of records and 20 features. A detailed description about the features of the dataset and its creation is found in [17].

Acute inflammation: The dataset aims for the diagnoses of two acute inflammations—one in the urinary bladder and the other in nephritis. The dataset contains six features and 120 patient records [17].

Hepatitis: Hepatitis is an inflammation of cells in the liver tissues. The dataset concerns on predicting the survival of a hepatitis patient. The collected data has 19 features and 155 patient records. The features were illustrated in [17].

Liver disorders: BUPA Liver disorders dataset has been created by BUPA medical research center in 1990. Liver dataset has been employed with diverse machine learning algorithms for disease diagnosis [32, 39]. It contains 345 patient records and 7 attributes. The first five attributes related to different blood tests results. The other features: One is related to the number of alcoholic drinks taken by the patient per day, and the other feature is the class label that indicates the existence or absence of liver disorder [17].

Parkinson's disease: The objective of this dataset is to identify the healthy individuals from the individuals with Parkinson's disease, which is done by the detection of dysphonia (voice disorders). The authors of the dataset have used pitch period entropy (PPE) as a measurement for dysphonia. The dataset created at the University of Oxford in cooperation with the National Center for Voice and Speech. The collected voice measurements were from 31 persons, where 23 of them were having Parkinson's disease. In addition, the total dataset contains 195 phonations. The phonations were recorded in an Industrial Acoustics Company by using a microphone located at 8 cm from the lips. Each attribute in the dataset corresponds to different voice measurement as illustrated in [17, 43].

Planning relax: The proposal of this dataset is to understand brain signals, which further facilitate in controlling prosthetic organs for disabled people. One of the common tools to monitor brain activity is electroencephalogram (EEG). Research studies that concerned on brain–computer interfaces proofed that the variations of power spectra in EEG signal correspond to the detection of an imagination of motor acts [59]. The authors of this dataset concern on classifying two types of mental states recorded using EEG, where the states are planning and relax. The planning state results from an imagination of a motor act. By analyzing the produced patterns of EEG waves, having a signal with 7–13 hertz, means that the tested person is in rest state, whereas the signals with frequency higher than 13 Hz reflect a planning of movement state. The dataset collected from healthy, right-handed subjects, in which EEG data is recorded for five trials of five seconds epoch for each state of relax and imagining of movement. The features are extracted by applying wavelet packet analysis on the recorded EEG signals, resulting in 12 features and 182 total number of records [17].

SPECTF heart: The dataset was gathered at the medical college of Ohio, which illustrates the diagnosing of cardiac Single-Proton Emission Computed Tomography (SPECT) images. SPECT imaging is a tool for diagnosing myocardial perfusion, where the cardiology injects the patient with radioactive tracer and takes two images—one after 15 min (called the stress image) and another one after 5 h of injection and referred to the rest image. The images are two-dimensional (black and white) that represent the left ventricle (LV) muscle perfusion, which is relative to the disper-

sion of radioactive counts into the myocardium. The cardiology compares the stress and rest images in order to identify any abnormalities in the LV muscle perfusion [40]. The authors applied image analysis and processing techniques to extract key features, resulting in 44 features and two output classes—normal and abnormal. The dataset contains 267 SPECT images and clinical patients' records, such as weight, height, sex, and the diagnosis [40].

Vertebral: The dataset is collected by Dr. Henrique, with the group of applied research in orthopedics (GARO) in France. The dataset aims to categorize orthopedic patients into normal or abnormal (which refer to either having disk hernia or spondylolisthesis). The dataset extracted six biomechanical features from the shape and direction of the pelvis and lumbar spine [17].

6 Experiments and Results

This section explains the experimental setup, the parameter settings, and the obtained results.

6.1 Experimental Setup

Initially, the methodology starts by dividing the datasets into 70% for training and 30% for testing. Afterward, MOPSO algorithm produces initial vectors of solutions; thus, the K-NN starts its training process as explained in Sect. 5.1, which utilizes both the potential solutions vectors from MOPSO, and an inner cross-validation criterion. The best optimal solutions obtained based on K-NN are evaluated on the testing set. The process repeated iteratively, until the maximum number of iterations of MOPSO is met.

The investigation of MOPSO as wrapper-based feature selection optimizer involves the comparison with other multi-objective optimizers—the NSGA-II and MOEA/D. The evaluation indicators are obtained by running each algorithm 30 times. Each run includes 100 iterations, the population size (nPop) is 30, and the maximum archive size is 100. The experiments are conducted by implementing the algorithms on MATLAB R2010b and running them on 12 benchmark medical datasets that explained in the previous section. The parameter settings for MOPSO, MOEA/D, and NSGA-II are as shown in Table 2.

6.2 Evaluation Measures

Evaluating the performance of the experimented algorithms depends on different performance metrics, in which they are accuracy, sensitivity, specificity, and g-mean.

Table 2 Initial settings of the MOEAs

Algorithm	Parameter	Value
MOPSO	Acceleration constants	[1.49618, 1.49618]
	Inertia weight, inertia damping rate	[0.7298, 0.99]
	No. of grids per dimension	7
	Inflation rate	0.1
	Leader selection pressure	2
	Deletion selection pressure	2
NSGA-II	Crossover probability (CP)	0.9
	No. of parents	2*round (CP*nPop/2)
	Mutation percentage (MP)	0.4
	Mutation rate	1/dimension
	No. of mutants	round (MP*nPop)
2cmMOEA/D	No. of sub-problems	nPop
	Crossover probability	0.5
	Neighbors ratio (T)	max (ceil (0.15*nPop), 2)

Accuracy is the fraction of the test set that is correctly classified by the model classifier [31], as shown in Eq. 10.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{10}$$

Sensitivity measures the percentage of true positives (TP) that are correctly identified as positives illustrated in Eq. 11, [31].

$$Sensitivity = \frac{TP}{TP + FN} \tag{11}$$

Specificity measures the percentage of true negatives (TN) that are correctly identified as negatives, (Eq. 12) [31].

$$Specificity = \frac{TN}{TN + FP} \tag{12}$$

G-mean is the geometric mean (Eq. 13) that indicates the classification consistency or central tendency for both normal and abnormal classes, which are defined by using the square root of the product of both the specificity and the sensitivity [31].

$$G - mean = \sqrt{TPR \times TNR} \tag{13}$$

6.3 Results

Table 3 compares the performance of MOPSO, NSGA-II, and MOEA/D over 12 datasets, using accuracy, sensitivity, specificity, and g-mean. Note that the boldface numbers show the best results. The results show a significant superiority of MOPSO over both NSGA-II and MOEA/D in most of the datasets. For instance, regarding to different measures, in terms of accuracy, MOPSO is better in eight datasets, which are the WBCD, Colon cancer, Heart, Acute inflammation, Hepatitis, Liver disorders, Parkinson, and Vertebral, in which MOPSO achieved the highest accuracy 0.919 in WBCD and Acute inflammation datasets, whereas in terms of g-mean, it is better in ten datasets, where NSGA-II obtained superior results for Leukemia and Parkinson datasets achieving (0.835, 0.824), respectively. While in terms of sensitivity, MOPSO is better in six datasets, MOEA/D is better in 4, and NSGA-II is better in 2. In terms of sensitivity, MOPSO achieved the best sensitivity value which is 1.000 for Leukemia dataset; while in contrast, MOEA/D achieved 100% sensitivity for Colon cancer dataset. Inspecting the specificity results, MOPSO achieved better results in seven datasets, obtaining 0.987 for Parkinson dataset, whereas NSGA-II performed well in five of the datasets. Noticeably, the three algorithms have competitive results; however, MOPSO tends to be superior.

Nonetheless, Fig. 7 shows the average Pareto front of MOPSO, NSGA-II, and MOEA/D. Each sub-figure describes the set of optimal non-dominated solutions that found for each dataset, in which the x-axis represents the number of features and the y-axis represents the error rate. Evidently, NSGA-II achieved better approximate PF in most of the datasets, holding fewer numbers of features and often lower error rate values. Yet, MOPSO had better approximate PF in five datasets and retained competitive number of features and lower error rates values; the datasets are Heart, Acute inflammation, Hepatitis, Liver disorders, and Vertebral. We notice fluctuations of the error rate over the number of features for Colon, Leukemia, and Diabetes, where Colon and Leukemia datasets are the gene expression datasets. However, for Colon dataset, MOPSO obtains lower values of error rate than NSGA-II and MOEA/D. In Colon and Heart datasets, MOEA/D performs slightly better than NSGA-II at some points, yet MOPSO performs better. In sub-figures WBCD and Parkinson, at some points MOPSO achieved slightly better error rate values, but also higher number of features in compared with both NSGA-II and MOEA/D.

7 Concluding Remarks

This chapter presents an introduction for multi-objective optimization problems and multi-objective evolutionary optimization, as well as it introduces the use of multi-objective particle swarm optimization as wrapper-based feature selection method, for disease detection. A comparison among MOPSO versus NSGA-II and MOEA/D is performed for examining their performance on 12 medical (disease-related) datasets.

Table 3 Comparisons of accuracy, sensitivity, specificity, and g-mean among MOPSO, MOEA/D, and NSGA-II for all datasets. *P*-values of Wilcoxon test of MOPSO against NSGA-II and MOEA/D are separated by commas. ($P \geq 0.05$ are written in italic typeface)

Dataset	Optimizer	Accuracy	Sensitivity	Specificity	G-mean
WBCD	MOPSO	**0.919**	**0.958**	**0.863**	**0.909**
	NSGA-II	0.904, 1.92E−08	0.931, 6.08E−10	0.852, 7.90E−03	0.887, 1.43E−08
	MOEA/D	0.902, 1.20E−05	0.945, 6.33E−05	0.842, 3.03E−02	0.891, 2.39E−04
Colon cancer	MOPSO	**0.811**	0.815	**0.798**	**0.806**
	NSGA-II	0.558, 2.86E−11	0.546, 1.71E−11	0.594, 3.99E−11	0.565, 2.98E−11
	MOEA/D	0.796, 2.51E−02	**1.000**, 1.16E−12	0.569, 1.20E−12	0.753, 2.84E−06
Heart	MOPSO	**0.795**	**0.915**	0.665	**0.771**
	NSGA-II	0.757, 7.29E−07	0.745, 8.85E−11	**0.785**, 6.61E−11	0.763, *6.35E−02*
	MOEA/D	0.760, 2.83E−04	0.798, 1.82E−09	0.705, 3.80E−03	0.730, 4.60E−03
Leukemia	MOPSO	0.825	**1.000**	0.649	0.805
	NSGA-II	**0.892**, 2.58E−11	0.975, 3.41E−07	**0.714**,1.14E−12	**0.835**, 1.03E−09
	MOEA/D	0.844, 4.89E−02	0.945, 1.76E−11	0.626, *7.23E−02*	0.768, 1.03E−02
Diabetes	MOPSO	0.708	0.820	**0.503**	**0.642**
	NSGA-II	**0.719**, 7.51E−06	**0.840**, 1.77E−11	0.471, 4.26E−08	0.627, 9.46E−06
	MOEA/D	0.672, 2.43E−09	0.802, 4.64E−04	0.430, 8.32E−10	0.584, 2.66E−10
Acute inflammation	MOPSO	**0.919**	0.954	0.892	**0.917**
	NSGA-II	0.880, 6.12E−14	0.815, 6.12E−14	**0.944**, 1.22E−12	0.875, 6.12E−14
	MOEA/D	0.818, 3.01E−12	**0.984**, 6.11E−07	0.712, 2.02E−07	0.799, 3.02E−12

(continued)

Table 3 (continued)

Dataset	Optimizer	Accuracy	Sensitivity	Specificity	G-mean
Hepatitis	MOPSO	**0.863**	0.929	**0.416**	**0.609**
	NSGA-II	0.796, 2.24E−11	**0.954**, 7.23E−11	0.282, 1.05E−11	0.518, 9.41E−11
	MOEA/D	0.789, 5.57E−11	0.926, *6.62E−01*	0.342, 2.01E−04	0.534, 1.30E−03
Liver disorders	MOPSO	**0.620**	**0.671**	**0.527**	**0.593**
	NSGA-II	0.534, 2.18E−11	0.649, 1.47E−04	0.390, 2.39E−11	0.497, 2.75E−11
	MOEA/D	0.571, 2.15E−08	0.623, 2.08E−05	0.480, 2.65E−05	0.545, 1.51E−08
Parkinson	MOPSO	**0.890**	0.577	**0.987**	0.752
	NSGA-II	0.881, 1.68E−04	0.714, 3.92E−11	0.954, 2.47E−11	**0.824**, 3.57E−11
	MOEA/D	0.879, 4.34E−02	**0.715**, 2.25E−10	0.908, 2.78E−11	0.801, 3.09E−06
Planning relax	MOPSO	0.634	**0.235**	0.758	**0.407**
	NSGA-II	**0.652**, 1.39E−02	0.058, 3.75E−11	**0.896**, 4.33E−09	0.140, 5.91E−11
	MOEA/D	0.596, 8.40E−07	0.073, 1.52E−11	0.894, 3.01E−11	0.222, 1.41E−10
SPECTF heart	MOPSO	0.775	**0.406**	0.839	**0.573**
	NSGA-II	**0.796**, 1.27E−04	0.205, 4.77E−11	**0.919**, 3.63E−11	0.412, 8.15E−11
	MOEA/D	0.728, 1.25E−08	0.223, 5.33E−10	0.905, 4.46E−09	0.431, 2.02E−08
Vertebral	MOPSO	**0.817**	0.726	**0.863**	**0.791**
	NSGA-II	0.796, 1.69E−14	0.699, 2.71E−14	0.844, 2.71E−14	0.768, 2.71E−14
	MOEA/D	0.770, 1.00E−12	**0.800**, 2.09E−07	0.759, 1.49E−12	0.776, *3.50E−01*

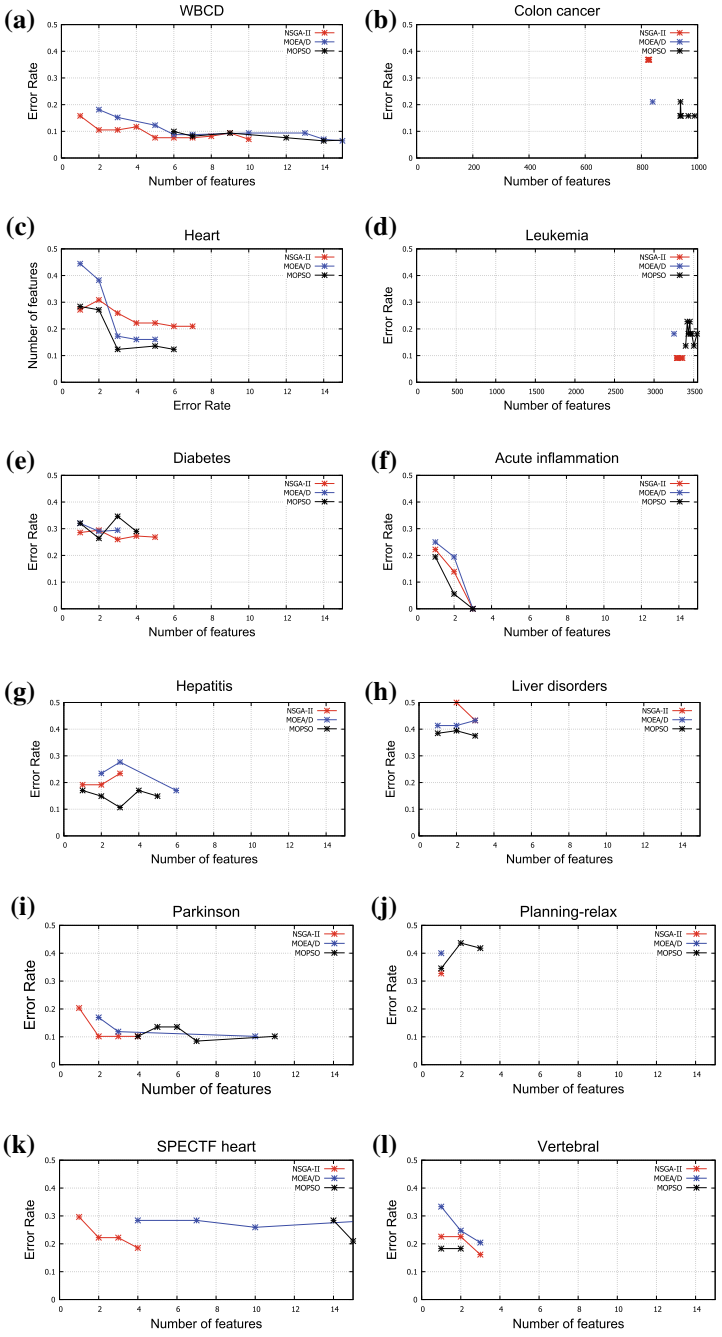


Fig. 7 Comparison of MOPSO, NSGA-II, and MOEA/D over all datasets

MOPSO as wrapper-based feature selection approach showed often superior performance in compared to other MOEAs. Hence, MOPSO has merits to be considered for feature selection techniques in medical applications. Since we maintained the multi-objective formulation, further there should be another step for decision making. However, the aim of developing smart techniques that aid in the diagnosis or in the clinical decision support systems is not to replace the clinician or the therapist, but to assist them in taking more efficient and reliable decisions.

References

1. Abbass HA (2002) An evolutionary artificial neural networks approach for breast cancer diagnosis. Artif Intell Med 25(3):265–281
2. Ahmed S, Mafarja M, Faris H, Aljarah I (2018) Feature selection using salp swarm algorithm with chaos. In: Proceedings of the 2nd international conference on intelligent systems, metaheuristics & swarm intelligence. ACM, pp 65–69
3. Aljarah I, Al-Zoubi AM, Faris H, Hassonah MA, Mirjalili S, Saadeh H (2018) Simultaneous feature selection and support vector machine optimization using the grasshopper optimization algorithm. Cogn Comput 1–18
4. Aljarah I, Faris H, Mirjalili S, Al-Madi N, Sheta A, Mafarja M (2019) Evolving neural networks using bird swarm algorithm for data classification and regression applications. Clust Comput 1–29
5. Aljarah I, Ludwig SA (2013) Towards a scalable intrusion detection system based on parallel pso clustering using mapreduce. In: Proceedings of the 15th annual conference companion on Genetic and evolutionary computation. ACM, pp 169–170
6. Aljarah I, Mafarja M, Heidari AA, Faris H, Zhang Y, Mirjalili S (2018) Asynchronous accelerating multi-leader salp chains for feature selection. Appl Soft Comput 71:964–979
7. Alnemer LM, Rajab L, Aljarah I (2016) Conformal prediction technique to predict breast cancer survivability. Int J Adv Sci Technol 96:1–10
8. Alon U, Barkai N, Notterman DA, Gish K, Ybarra S, Mack D, Levine AJ (1999) Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. Proc Natl Acad Sci 96(12):6745–6750
9. Berner ES (2007) Clinical decision support systems, vol 233. Springer
10. Blum AL, Langley P (1997) Selection of relevant features and examples in machine learning. Artif Intell 97(1–2):245–271
11. Bramer M (2007) Principles of data mining, vol 180. Springer
12. Coello CAC, Lechuga MS (2002) Mopso: a proposal for multiple objective particle swarm optimization. In: Proceedings of the 2002 congress on evolutionary computation. CEC'02 (Cat. No. 02TH8600), vol 2. pp 1051–1056, IEEE
13. Coello CA, Lamont GB, Van Veldhuizen DA et al (2007) Evolutionary algorithms for solving multi-objective problems, vol 5. Springer
14. Corne DW, Knowles JD, Oates MJ (2000) The pareto envelope-based selection algorithm for multiobjective optimization. In: International conference on parallel problem solving from nature. Springer, pp 839–848
15. Deb K, Agrawal S, Pratap A, Meyarivan T (2000) A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii. In: International conference on parallel problem solving from nature. Springer, pp 849–858
16. Deb K, Kalyanmoy D (2001) Multi-objective optimization using evolutionary algorithms. Wiley, New York, NY, USA
17. Dua D, Efi KT (2017) UCI machine learning repository

18. Dioşan L, Andreica A (2015) Multi-objective breast cancer classification by using multi-expression programming. Appl Intell 43(3):499–511
19. Dos Santos BC, Nobre CN, Zárate LE (2018) Multi-objective genetic algorithm for feature selection in a protein function prediction context. In: 2018 IEEE congress on evolutionary computation (CEC). IEEE, pp 1–6
20. Dubey AK, Gupta U, Jain S (2016) Analysis of k-means clustering approach on the breast cancer wisconsin dataset. Int J Comput Assist Radiol Surg 11(11):2033–2047
21. Dudoit S, Fridlyand J, Speed TP (2002) Comparison of discrimination methods for the classification of tumors using gene expression data. J Am Stat Assoc 97(457):77–87
22. Dussaut JS, Vidal PJ, Ponzoni I, Olivera AC (2018) Comparing multiobjective evolutionary algorithms for cancer data microarray feature selection. In: 2018 IEEE congress on evolutionary computation (CEC). IEEE, pp 1–8
23. Eberhart R, Kennedy J (1995) A new optimizer using particle swarm theory. In: Micro machine and human science, 1995. MHS'95. Proceedings of the Sixth International Symposium on. IEEE, pp 39–43
24. Faris H, Aljarah I, Al-Betar MA, Mirjalili S (2018) Grey wolf optimizer: a review of recent variants and applications. Neural Comput Appl, pp 1–23
25. Faris H, Aljarah I, Al-Shboul B (2016) A hybrid approach based on particle swarm optimization and random forests for e-mail spam filtering. In: International Conference on Computational Collective Intelligence. Springer, pp 498–508
26. Faris H, Hassonah MA, Al-Zoubi AM, Mirjalili S, Aljarah I (2018) A multi-verse optimizer approach for feature selection and optimizing svm parameters based on a robust system architecture. Neural Comput Appl 30(8):2355–2369
27. Faris H, Mafarja MM, Heidari AA, Aljarah I, Al-Zoubi AM, Mirjalili S, Fujita H (2018) An efficient binary salp swarm algorithm with crossover scheme for feature selection problems. Knowl-Based Syst 154:43–67
28. Faris H, Mirjalili S, Aljarah I (2019) Automatic selection of hidden neurons and weights in neural networks using grey wolf optimizer based on a hybrid encoding scheme. Int J Mach Learn Cybern 1–20
29. Friedman N, Linial M, Nachman I, Pe'Er D (2000) Using bayesian networks to analyze expression data. J Comput Biol 7(3–4):601–620
30. Golub TR, Slonim DK, Tamayo P, Huard C, Gaasenbeek M, Mesirov JP, Coller H, Loh ML, Downing JR, Caligiuri MA et al (1999) Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. Science 286(5439):531–537
31. Han J, Pei J, Kamber M (2011) Data mining: concepts and techniques. Elsevier
32. Haque MR, Islam MM, Iqbal H, Reza MS, Hasan MK (2018) Performance evaluation of random forests and artificial neural networks for the classification of liver disorder. In: 2018 international conference on computer, communication, chemical, material and electronic engineering (IC4ME2). IEEE pp 1–5
33. Holland JH et al (1992) Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT press
34. Ibrahim AO, Shamsuddin SM, Saleh AY, Abdelmaboud A, Ali A (2015) Intelligent multi-objective classifier for breast cancer diagnosis based on multilayer perceptron neural network and differential evolution. In: 2015 international conference on computing, control, networking, electronics and embedded systems engineering (ICCNEEE). IEEE pp 422–427
35. Kennedy J, Eberhart RC (1997) A discrete binary version of the particle swarm algorithm. In: 1997 IEEE international conference on systems, man, and cybernetics. Computational cybernetics and simulation, vol 5. IEEE pp 4104–4108
36. Knowles J, Corne D (1999) The pareto archived evolution strategy: a new baseline algorithm for pareto multiobjective optimisation. In: Congress on Evolutionary Computation (CEC99), vol 1, pp 98–105
37. Kong Q, Wang D, Wang Y, Jin Y, Jiang B (2018) Multi-objective neural network-based diagnostic model of prostatic cancer. Xitong Gongcheng Lilun Yu Shijian/Syst Eng Theory Pract 38(2):532–544. cited By 0

38. Kuhn M, Johnson K (2013) Applied predictive modeling, vol 26. Springer
39. Kumar S, Katyal S (2018) Effective analysis and diagnosis of liver disorder by data mining. In: 2018 international conference on inventive research in computing applications (ICIRCA). IEEE pp 1047–1051
40. Kurgan LA, Cios KJ, Tadeusiewicz R, Ogiela M, Goodenday LS (2001) Knowledge discovery approach to automated cardiac spect diagnosis. Artif Intell Med 23(2):149–169
41. Kursawe F (1990) A variant of evolution strategies for vector optimization. In: International conference on parallel problem solving from nature. Springer, pp 193–197
42. Li X, Yin M (2013) Multiobjective binary biogeography based optimization for feature selection using gene expression data. IEEE Trans NanoBioscience 12(4):343–353
43. Little MA, McSharry PE, Hunter EJ, Spielman J, Ramig LO (2009) Suitability of dysphonia measurements for telemonitoring of parkinson's disease. IEEE Trans Bio-Med Eng 56(4):1015
44. Mafarja M, Aljarah I, Faris H, Hammouri AI, Al-Zoubi AM, Mirjalili S (2019) Binary grasshopper optimisation algorithm approaches for feature selection problems. Expert Syst Appl 117:267–286
45. Mafarja M, Aljarah I, Heidari AA, Faris H, Fournier-Viger P, Li X, Mirjalili S (2018) Binary dragonfly optimization for feature selection using time-varying transfer functions. Knowl-Based Syst 161:185–204
46. Mafarja M, Aljarah I, Heidari AA, Hammouri AI, Faris H, A-Zoubi AM, Mirjalili S (2018) Evolutionary population dynamics and grasshopper optimization approaches for feature selection problems. Knowl-Based Syst 145:25–45
47. Mafarja M, Heidari AA, Faris H, Mirjalili S, Aljarah I (2020) Dragonfly algorithm: theory, literature review, and application in feature selection. In: Nature-inspired optimizers. Springer, pp 47–67
48. Mafarja MM, Mirjalili S (2018) Hybrid binary ant lion optimizer with rough set and approximate entropy reducts for feature selection. Soft Comput 1–17
49. Mirjalili S, Jangir P, Saremi S (2017) Multi-objective ant lion optimizer: a multi-objective optimization algorithm for solving engineering problems. Appl Intell 46(1):79–95
50. Mirjalili S, Lewis A (2013) S-shaped versus v-shaped transfer functions for binary particle swarm optimization. Swarm Evol Comput 9:1–14
51. Mirjalili S, Saremi S, Mirjalili SM, Coelho LDS (2016) Multi-objective grey wolf optimizer: a novel algorithm for multi-criterion optimization. Expert Syst Appl 47:106–119
52. Mirjalili SZ, Mirjalili S, Saremi S, Faris H, Aljarah I (2018) Grasshopper optimization algorithm for multi-objective optimization problems. Appl Intell 48(4):805–820
53. Mitra S, Banka H (2006) Multi-objective evolutionary biclustering of gene expression data. Pattern Recognit 39(12):2464–2477
54. Mohemmed AW, Zhang M (2008) Evaluation of particle swarm optimization based centroid classifier with different distance metrics. In: 2008 IEEE congress on evolutionary computation (IEEE world congress on computational intelligence). IEEE, pp 2929–2932
55. Mugambi EM, Hunter A (2003) Multi-objective genetic programming optimization of decision trees for classifying medical data. In: International conference on knowledge-based and intelligent information and engineering systems. Springer, pp 293–299
56. Murata T, Ishibuchi H (1995) Moga: multi-objective genetic algorithms. IEEE Int Conf Evol Comput 1:289–294
57. rey Horn J, Nafpliotis N, Goldberg DE (1994) A niched pareto genetic algorithm for multiobjective optimization. In: Proceedings of the first IEEE conference on evolutionary computation, IEEE world congress on computational intelligence, vol 1. Citeseer, pp 82–87
58. Sahoo A, Chandra S (2017) Multi-objective grey wolf optimizer for improved cervix lesion classification. Appl Soft Comput 52:64–80
59. Santhosh J, Bhatia M, Sahu S, Anand S (2004) Quantitative eeg analysis for assessment to plana task in amyotrophic lateral sclerosis patients: a study of executive functions (planning) in als patients. Cogn Brain Res 22(1):59–66
60. Schaffer JD (1985) Multiple objective optimization with vector evaluated genetic algorithms. In: Proceedings of the first international conference on genetic algorithms and their applications (1985) Lawrence Erlbaum Associates. Publishers, Inc., p 1985

61. Shahbeig S, Rahideh A, Helfroush MS, Kazemi K (2018) Gene selection from large-scale gene expression data based on fuzzy interactive multi-objective binary optimization for medical diagnosis. Biocybern Biomed Eng 38(2):313–328
62. Sarah S, Hossam F, Ibrahim A, Seyedali M, Ajith A (2018) Evolutionary static and dynamic clustering algorithms based on multi-verse optimizer. Eng Appl Artif Intell 72:54–66
63. Sohrabi MK, Tajik A (2017) Multi-objective feature selection for warfarin dose prediction. Comput Biol Chem 69:126–133
64. Srinivas N, Deb K (1994) Muiltiobjective optimization using nondominated sorting in genetic algorithms. Evol Comput 2(3):221–248
65. Turing AM, Lerner A, (1987) Aaai 1991 spring symposium series reports. 12(4): Winter 1991, 31–37 aaai 1993 fall symposium reports. 15(1): Spring, (1994) 14–17 aaai 1994 spring symposium series. Intelligence 1(49):8
66. Yang X-S (2010) Nature-inspired metaheuristic algorithms. Luniver press
67. Zhang Q, Li H (2007) Moea/d: a multiobjective evolutionary algorithm based on decomposition. IEEE Trans Evol Comput 11(6):712–731

Multi-objective Particle Swarm Optimization for Botnet Detection in Internet of Things

Maria Habib, Ibrahim Aljarah, Hossam Faris and Seyedali Mirjalili

Abstract Nowadays, the world witnesses an immense growth in Internet of things devices. Such devices are found in smart homes, wearable devices, retail, health care, industry, and transportation. As we are entering Internet of things (IoT) digital era, IoT devices not only hack our world, but also start to hack our personal life. The widespread IoT has created a rich platform for potential IoT cyberattacks. Data mining and machine learning techniques have significant roles in the field of IoT botnet detection. The aim of this chapter is to develop detection model based on multi-objective particle swarm optimization (MOPSO) for identifying the malicious behaviors in IoT network traffic. The performance of MOPSO is verified against multi-objective non-dominating sorting genetic algorithm (NSGA-II), common traditional machine learning algorithms, and some conventional filter-based feature selection methods. As per the obtained results, MOPSO is competitive and outperforms NSGA-II, traditional machine learning methods, and filter-based methods in most of the studied datasets.

Keywords Internet of things · Classification · Multi-objective particle swarm optimization · Non-dominating sorting genetic algorithm · Multi-objective feature selection · Botnets

M. Habib · I. Aljarah · H. Faris
King Abdullah II School for Information Technology,
The University of Jordan, Amman, Jordan

I. Aljarah
e-mail: i.aljarah@ju.edu.jo

H. Faris
e-mail: hossam.faris@ju.edu.jo

S. Mirjalili (✉)
Torrens University Australia,
Brisbane, QLD 4006, Australia
e-mail: ali.mirjalili@gmail.com

Griffith University, Brisbane, QLD 4111, Australia

S. Mirjalili et al. (eds.), *Evolutionary Machine Learning Techniques*,
Algorithms for Intelligent Systems, https://doi.org/10.1007/978-981-32-9990-0_10

1 Introduction and Related Works

In recent years, there has been a rapid increase in IoT devices. These devices are physical, electronic objects, or even embedded systems that are connected to Internet and can be connected together without human intervention [55]. IoT networks have a strong connection with individuals' daily life in organizations, governmental agencies, or homes. Massachusetts Institute of Technology defines the terminology of IoT as a network of all connected devices (or things) in order to share information and enhance the sensing, communication, and computational abilities [55]. From another perspective, IoT can be defined as a network of electronic devices communicates together via wireless technologies, so they exchange data, process it, and store it [14]. Despite the high proliferation of IoT, like any other Internet-connected devices, IoT is susceptible to wide range of network and software security breaches [14]. Since IoT is equipped with low computational power in memory, battery, processing unit, and streaming bandwidth, protecting IoT systems is considered a challenging task. IoT is not just vulnerable to security issues as other Internet networks, mobile networks, or wireless sensor networks. But it is more exposed to several security issues since it is highly heterogeneous within interconnected networks as well as highly scalable [29].

Privacy and security are two major concerns when using IoT devices these days. IoT devices are plug-and-play devices, which means that they are highly susceptible to brute-force attacks and denial-of-service (DoS) attacks [14]. As many IoT devices have default passwords by their manufacturing settings, this makes them an attractable target for botnets or IoT malware [14].

IoT encompasses three main layers—the perception layer, the transportation layer, and the application layer. As each layer has different protocols, then each is exposed to diverse security issues [29]. Taking in particular the transportation layer in which IoT communicates over various technologies, it could be Bluetooth, Wi-Fi, Zigbee, and 3G networks. Thus, the communication layer is vulnerable to denial-of-service (DoS) or distributed denial-of-service (DDoS) attacks, middle and forgery attacks, and the risk of IPv6 and 6LoWPAN application protocols [29]. It has been observed in 2016 that a large-scale DDoS attack using Mirai botnet targeted the domain name service provider Dyn, which resulted in disrupting large number of services like CNN, PayPal, and Netflix.

Intrusion detection systems (IDSs) are one of the solutions proposed to protect networks from potential threats and intrusions. IDSs are software or hardware devices that continuously monitor the network in order to capture malicious behaviors or patterns [18]. IDSs involve three major modules—the sensor module that collects data from the environment, the analysis module, and the reporting module [18]. We focus on the analysis module which is responsible about processing massive amount of traffic data in order to extract abnormal patterns or malicious behaviors. Therefore, the analysis module is a smart module that deploys data science and data mining techniques so as to protect the corresponding network. Accordingly, developing a lightweight security solution for IoT is at significant requirement.

IoT networks involve large number of devices which result in large amount of data with high dimensions. Processing such large amount of data requires the utilization of specialized mining techniques to handle it. One of the crucial steps of data preprocessing is the feature selection process.

Considering a dataset with N number of features, it has a search space of 2^N. Having redundant and irrelevant features in a classification task may deteriorate the classification performance especially in a large-scale dataset. Feature selection (FS) aims to tackle this problem by choosing the relevant features, hence, minimizing the number of features, minimizing the classifier training time, and maximizing the classification performance. Since the size of the search space is highly affected by the number of features, the process of finding the optimal set of features is considered to be an exhaustive task [57]. Several search techniques have been proposed in the literature, such as greedy search [12], sequential search [12], and tabu search [59]. However, these traditional search methods suffer from trapping in local optima [57]. On the other hand, nature-inspired optimizers are popular for their global search capability and the successful ability to handle feature selection problem [1, 3, 6, 22, 23, 36–40].

In the literature, there is a tendency toward machine learning methods for addressing the problem of attack detection or botnet detection in IoT. In [48], for instance, the authors used neural network (NN) for intrusion detection in IoT, in which they utilized the negative selection algorithm to construct a new training set and predict new unseen intrusions. Although they reduced the computational complexity by doing the training phase remotely, the designed algorithm showed a limited performance in the intrusion detection process [48].

In [42], the authors deployed Naive Bayes (NB) classification algorithm with a multi-agent system for detecting distributed denial-of-service attacks. Xiao and Liang in [56] stated how the use of artificial intelligence techniques enhances the security of IoT devices. As a result, they classified state-of-the-art studies into studies that used supervised machine learning (ML) techniques (or some other used unsupervised ML techniques) or the use of reinforcement learning methods. Other methods used in the literature are random forest (RF), gradient boosting (GB), and NB to detect malicious activities, such as scanning and DoS attacks. However, NB performed the worst among RF and GB in terms of recall and precision [53].

In [35], the authors used suppressed fuzzy clustering and principal component analysis (PCA) for IoT intrusion detection. The results were promising, but they deteriorate with the increase of data volume in terms of detection efficiency. In [34], the authors combined a RF classifier with the Bat algorithm (BA) for feature selection in an attempt to detect intrusions in IoT networks. The results were superior compared to different algorithms such as support vector machine (SVM), AdaBoost, and decision tree (DT). However, they aim to test the proposed approach on real IoT network traffic. Another relevant work was conducted in [50], in which the authors used an extreme learning machine-based semi-supervised fuzzy C-means clustering for detecting network-level attacks in IoT. The proposed methodology achieved better accuracy than traditional machine learning algorithms. In [47], the authors developed AdaBoost ensemble learning method using DT, NB, and artificial neural network

(ANN) for intrusion detection in IoT. The proposed algorithm outperformed SVM, Markov chain (MC), and Bayesian system based on detection rate, accuracy, and processing time. However, the proposed algorithm designed to detect specifically three types of attacks depending on simulated IoT data.

Hard optimization problems cannot be solved using traditional algorithms in deterministic polynomial time since the required time to solve them increases exponentially with the growing size of the problem [15].

Optimization algorithms can be categorized based on the search strategy into deterministic algorithms or stochastic algorithms [58]. The deterministic algorithms use repetitive process for the values assignment of variables and functions. Thus, as they start from the same starting point, they always produce the same output results [15]. Stochastic algorithms utilize a randomization component, where a typical example is the evolutionary algorithms. Stochastic algorithms might be heuristic or metaheuristic, where heuristic means searching for solutions by trial and error, whereas *meta* is a Greek prefix, means higher level or beyond. Metaheuristic can solve hard optimization problems and can find good solutions in deterministic, reasonable amount of time as they can balance between exploration (searching at global level) and exploitation (searching at local level). However, there is no guarantee that the good found solutions are optimal solutions [58].

Generally, metaheuristic algorithms are nature-inspired algorithms that consist the evolutionary algorithms and the swarm-based algorithms [58]. Swarm-based algorithms concern on the behavior of a collection of interacting agents in a population that imitates natural systems behaviors, such as the social behavior of ants, bees, insects or birds flocking, and fish schooling [58].

To the best of our knowledge, there is little in the literature on the use of evolutionary algorithms or even evolutionary multi-objective algorithms for attack detection in the field of IoT. In [49], authors proposed a biometric authentication technique based on genetic algorithms in order to prevent black-hole attacks in IoT networks, whereas authors in [32] utilized single-objective ant colony optimization to detect Sybil attacks in social IoT networks. As well as, in [4, 5, 30, 52], particle swarm optimization algorithm is integrated for attack detection and secure authentication in cloud computing-related networks. Nonetheless, [54] utilized artificial bee colony algorithm for detecting extensible markup language denial-of-service attacks over clouds' architectures. While in [51], the authors used Voronoi diagram-based evolutionary algorithm (VorEAl) for intrusion detection in IoT, and VorEAl evolves Voronoi diagrams which were utilized to do the classification of data. Moreover, VorEAl applied a multi-objective optimization principle. Despite the promising results, they aimed to construct a dataset that is more close to a real IoT traffic.

In this chapter, we first formulate the IoT botnet detection as a multi-objective optimization problem for feature selection for IoT traffic classification. FS has multiple objectives by nature, herein; it is handled as a two-objective optimization problem. The first objective is maximizing the classification performance by reducing the classification error rate. The second objective is minimizing the selected number of features. Therefore, MOPSO algorithm is utilized as wrapper-based feature selection method. Additionally, the MOPSO algorithm is used to find the Pareto opti-

mal front of this problem. MOPSO is compared with NSGA-II and several machine learning algorithms. Furthermore, the MOPSO, as wrapper-based feature selection, is compared with different filter-based techniques.

The rest of the chapter is organized as follows: Sect. 2 presents a background about IoT botnets, MOPSO, non-dominating sorting genetic algorithm, and multi-objective feature selection. Section 3 discusses the creation of datasets, binary multi-objective particle swarm optimization, and the designed experiments. Section 4 shows the experimental evaluation and results. Finally, Sect. 5 summarizes the work and suggests possible future work.

2 Internet of Things Botnets

The existence of large number of connected devices with limited computational and technical capabilities leads to having highly vulnerable IoT networks. Generally speaking, IoT devices go through three main operations—data collection, data transmission, and data processing. In each of these operations, such devices are vulnerable to different types of attacks. Besides, the low computational resources of IoT made these devices relatively easier to be flooded, especially for the attackers whom orchestrate DDoS attacks.

DDoS attacks seek to overwhelm IoT resources like the memory, central processing unit (CPU), and bandwidth [7]. Often, DDoS is conducted using TCP, UDP, or HTTP flooding attacks, which results in stopping server's services or dropping legitimate requests sent from end users [7]. DDoS attacks are launched from botnets which are group of connected network devices infected with a malware and called bots. Bots might be IoT devices like cameras, thermostat, and so on; they execute several tasks like scanning the network for vulnerable devices, send spam messages, infect vulnerable devices, and carry out several types of attacks [19–21, 41]. Nowadays, IoT devices form an attracting platform for performing extensive DDoS attacks. As a result, different IoT malwares have emerged triggering large-scale DDoS attacks that surpassed 1.2 terabit per second (Tbps) [41].

Bashlite and Mirai are two kinds of botnets intended for IoT devices. Bashlite (also known as Gafgyt, Torlus, and Lizkebab) is a common malware targeting Linux-based IoT devices with the aim of launching DDoS attacks. DDoS attacks often conducted by performing TCP or UDP floods. Bashlite infects IoT appliances by brute-force its authentication credentials using the telnet port. In 2015, Bashlite's source code has been revealed, which led to the evolution of different variants [7].

Mirai is a malware which infects IoT devices, same as Bashlite in order to perform DDoS attacks. Originally, Mirai's source code is based on Bashlite's. Mirai has a large attack size; it had infected up to 65,000 devices in its first 20 hours. It has been identified that Mirai had been targeting digital video recorders (DVRs), IP cameras, routers, and webcams [8]. Mainly, Mirai works by first scanning a potential victim network by sending TCP-SYN floods to IPv4 addresses on telnet ports 23 or 2323. Once a vulnerable device detected, it starts brute-forcing its credentials. Upon having

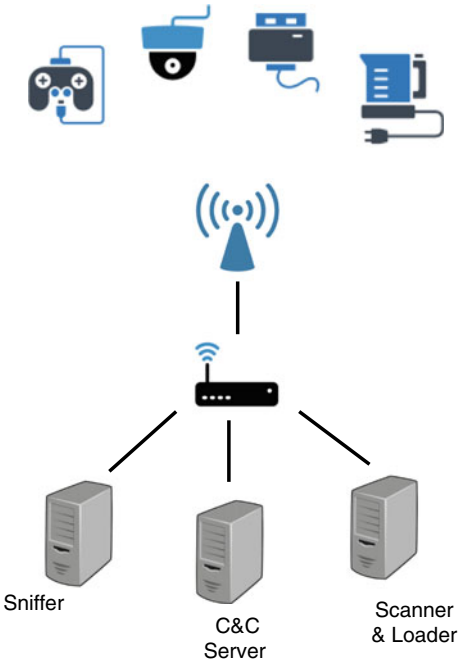


Fig. 1 Botnet attack model for IoT

successful credentials, the loader login to the device and infects it with the malware. The created botmaster listens to the command and controller server (C&C) and keeps searching for other victim devices [31]. Figure 1 illustrates how such botnet might infect a network of IoT devices.

3 Botnet Detection Model Based on MOPSO

Originally, the intrusions' objective is to deplete network's resources and services, which results in damaging the integrity, confidentiality, and validity of the target network's data and functionality [28]. Intrusion detection systems are software or hardware devices that iteratively monitor the network traffic in order to detect malicious activities or intrusions [28]. Generally, IDSs are classified into signature-based systems or anomaly-based systems. Signature-based systems recognize previously defined attacks by identifying patterns that matching the signatures of the known attacks. However, its main drawback is the inability to identify and mitigate zero-day threats, whereas anomaly-based systems monitor the network behavior and classify it to normal or abnormal, by utilizing either statistical-based techniques, knowledge-based techniques, or machine learning techniques [25]. Anomaly-based IDSs are well known for their ability to identify unforeseen attacks [28]. This chapter examines the use of multi-objective particle swarm optimization as a detection algorithm

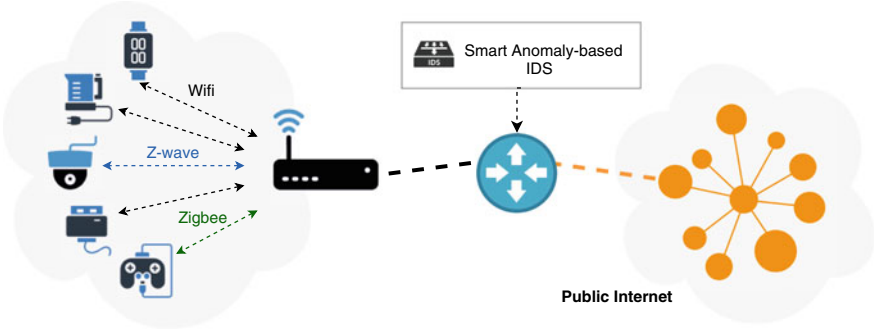


Fig. 2 Illustration of deploying machine learning-based intrusion detection system over IoT networks

for such anomaly-based intrusion detection system for IoT. Figure 2 illustrates the process of deploying smart algorithms (as in our case the MOPSO) for intrusion detection for IoT networks.

3.1 Multi-objective Particle Swarm Optimization

In 1995, Kennedy and Eberhart proposed particle swarm optimization (PSO) algorithm that mimics the swarm social behavior of bird flocks and fish schooling. PSO is a stochastic search algorithm aims to find the optimal solution within the search space of an objective function, likewise a swarm of birds that searches for the best source of food. In PSO, a set of randomly generated solutions, called particles, search for the best solution by adjusting their flight directions and velocities. It is important to note that the position and velocity of particles are influenced by local and global experiences of particles. The local experience (denoted as pbest) is known as the cognitive part of particle's movement and depends on the particle own experience on finding best position among its neighbors and over each iteration within the search process. The global experience (denoted as gbest) is known as the social component of the particle's movement and depends on finding the best position among all particles [17]. The particles modify their positions and velocities using Eqs. 1 and 2, respectively.

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

$$\begin{aligned} v_{id}(t+1) = w * v_{id}(t) + r_1 * c_1 * [p_{id}(t) - x_{id}(t)] \\ + r_2 * c_2 * [g_d(t) - x_{id}(t)] \end{aligned} \tag{2}$$

where d is the dimension number. w is the inertia weight, r_1 *and* r_2 are random numbers $\in [0, 1]$, c_1 *and* c_2 are acceleration constants, p_{id} shows the personal best position for a particle (*pbest*), and g_d indicates the global best position among the population (*gbest*).

Multi-objective optimization problems are problems that have normally a set of conflicting objectives that needs optimization. The aim of multi-objective optimization is to find a set of trade-off (non-dominated) solutions rather than one optimal solution [13]. The set of non-dominated solutions is called Pareto optimal set. Experimentally, PSO proved to outperform classical methods to solve optimization problems [10]. The formulation of MOPSO integrates the use of Pareto dominance to produce a group of leaders that guide the search process. Certainly, those leaders represent the set of non-dominated solutions which stored in external memory (known as repository or archive). Selecting a leader among set of potential leaders to be the global guide is done using roulette wheel selection based on a hypercube score. Mainly, the archive composed of two parts the controller and the grid. The controller is responsible about accepting new solutions into the archive or not accepting, whereas the grid is to enhance the diversity of solutions since the objective space is divided into hypercubes regions that assigned a fitness value based on the number of solutions they contain. The fittest and preferred hypercube is the one with less number of solutions.

The following steps summarize the procedure of MOPSO algorithm [10].

1. Initialize the population POP_i, where i =[1, 2,…,N], N is the population size.
2. Initialize the velocity of each particle VEL_i.
3. Evaluate each particle and assign it with a fitness value.
4. Store non-dominated solutions in external archive (REP).
5. Create hypercubes.
6. Initialize the memory of each particle and store the initial positions as the best particles' positions found.
7. Calculate the speed of each particle using Eq. 3.
8. Calculate the new positions of each particle using Eq. 1.
9. Preserve the particles within the search space constraints' boundaries.
10. Evaluate the particles.
11. Update the REP and hypercubes by inserting the new non-dominated solutions.
12. Update the memory of personal best position of each particle.
13. If the termination condition is met, then stop, otherwise go back to step 7.

Updating the velocity in MOPSO is slightly different from that in PSO, in which the global best solution should be chosen from the repository using leader selection mechanism as given by Eq. 3:

$$v_{id}(t+1) = w * v_{id}(t) + r_1 * c_1 * (p_{id}(t) - x_{id}(t)) + r_2 * c_2 * (REP(h) - x_{id}(t)) \quad (3)$$

where $REP(h)$ is a to be taken from the repository using a roulette wheel selection methodology.

3.2 Binary MOPSO

Originally, MOPSO was designed to deal with continuous variables [13]. Since we are handling feature selection problem, this requires transforming the search space into binary search space. If a feature is selected it is indicated by one, otherwise it is a zero. Converting the continuous PSO into binary PSO requires the adoption of such kind of mathematical transfer functions (TF) [45]. The TF defines a probability to select a feature or not select it. If the random probability is less than the value of the transfer function, then the feature is assigned a value zero, otherwise it is one. We use the Sigmoid transfer function that is defined by Eq. 4.

$$T(v_{id}(t)) = \frac{1}{1 + e^{-2 * v_{id}(t)}} \tag{4}$$

where $v_{id}(t)$ is subject to the particle velocity at iteration t and dimension d.

Using the produced value of the TF from Eq. 4, the particles' position will be updated using Eq. 5

$$x_{id}(t+1) = \begin{cases} 1 & \text{if } rand \geq T(v_{id}(t)) \\ 0 & \text{if } rand < T(v_{id}(t)) \end{cases} \tag{5}$$

Consequently, the velocity in Eq. 3 will be updated using the produced binary position from Eq. 5.

3.3 Fitness Formulation

MOPSO and NSGA-II are multi-objective optimization algorithms used for FS. Evaluating the selected subset of features is done based on two objectives: The first objective $f_1(x)$ is the classification error rate (ER), and the second one $f_2(x)$ is the number of selected features. The formulation of the fitness function is as described by Eq. 6:

$$Min.\ F(x) = \begin{cases} f_1(x) = \frac{FP+FN}{TP+FP+TN+FN} \times 100\% \ , \ (P+N) \in \mathbb{R}^+ \\ f_2(x) = \frac{l}{A} \qquad\qquad\qquad\qquad , \ l \in A,\ A \in \mathbb{R}^+ \end{cases} \tag{6}$$

l is the selected number of features. A is the total number of features. FP is the false positives, FN is the false negatives, TP is the true negatives, and the TN is the true negatives [27].

3.3.1 K-Nearest Neighbor (K-NN)

K-NN is a common, nonparametric machine learning algorithm used for different purposes, such as classification and regression. For classification, the objective of K-NN is to classify new objects depending on the features and the K-nearest neighbors of the training samples. Hence, the new object assigned a class label that is the most common class of its neighbors [11]. K-NN is simple and easy to implement, which is applied to different application areas, as feature selection [9, 46] and pattern recognition [2, 44]. The K-NN (k=5) is used for assessing the performance of the utilized optimizers, with fivefold cross-validation [33, 46], in which the classification error rates were averaged to get the final fitness of each candidate solution.

The output of multi-objective optimization problem is a set of non-dominated vectors. For FS problems, the non-dominated vectors represent two objective functions—the error rate and the number of features. An example of a non-dominated solution for FS is v_1=[0.001,5], where 0.001 is the error rate and 5 is the number of features.

3.4 Experimental Setup

The implementation of both MOPSO and NSGA-II has been done in MATLAB R2010b. All datasets are split into 34, 66% for testing and training, respectively. Basically, MOPSO generates initial vectors of solutions that indicate the potential optimal subset of features. In order to evaluate the selected subsets of features, the 5-NN classifier is used with fivefold cross-validation on the training set, during the fitness evaluation of MOPSO [33, 46]. Afterward, the selected subsets of features were evaluated on the testing set (34%) using (5-NN) in order to have the final testing classification error rate. The methodology is repeated, iteratively, until the termination criterion is met, which is the maximum number of iterations. Figure 3 shows an overview of the designed methodology. Each algorithm had 10 independent runs, of which each run includes 100 iterations, the population size is 30, and the maximum archive size is 100. The parameter settings for MOPSO and NSGA-II are as shown in Table 1.

3.5 Dataset Description

We have constructed five datasets from nine public IoT datasets which were drawn from UCI repository [16]. The datasets represent real network traffic collected from nine IoT devices. The devices are a thermostat, a webcam, two types of doorbells, four types of security cameras, and a baby monitor. A presents a detailed description about the nine IoT datasets. The raw traffic has been collected as follows:

Table 1 Parameter settings

Algorithm	Parameter	Value
MOPSO	Acceleration constants	[1.49618, 1.49618]
	Inertia weight, inertia damping rate	[0.7298, 0.99]
	Number of grids per dimension	7
	Inflation rate	0.1
	Leader selection pressure	2
	Deletion selection pressure	2
NSGA-II	Crossover percentage (CP)	0.9
	Number of parents	2*round(CP*nPop/2)
	Mutation percentage (MP)	0.4
	Number of mutants	round(MP*nPop)

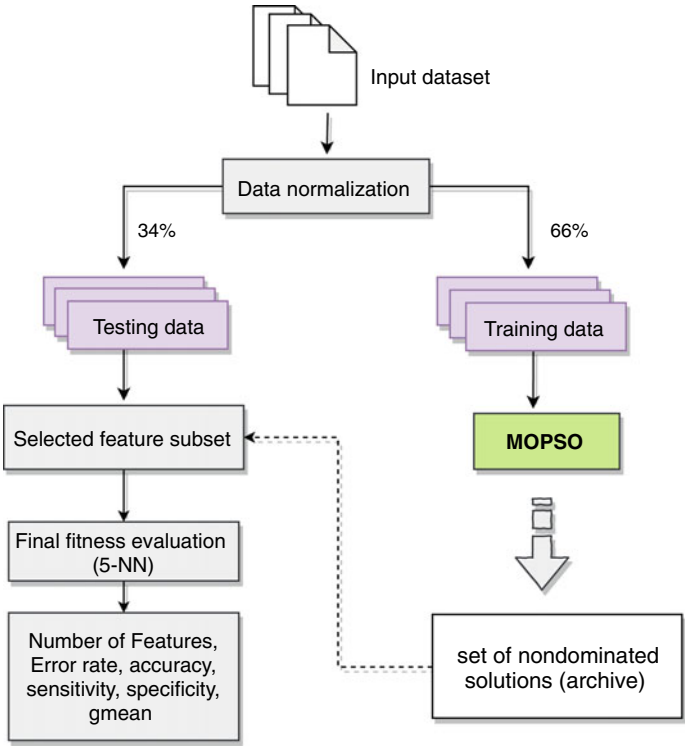


Fig. 3 Overview of methodology design

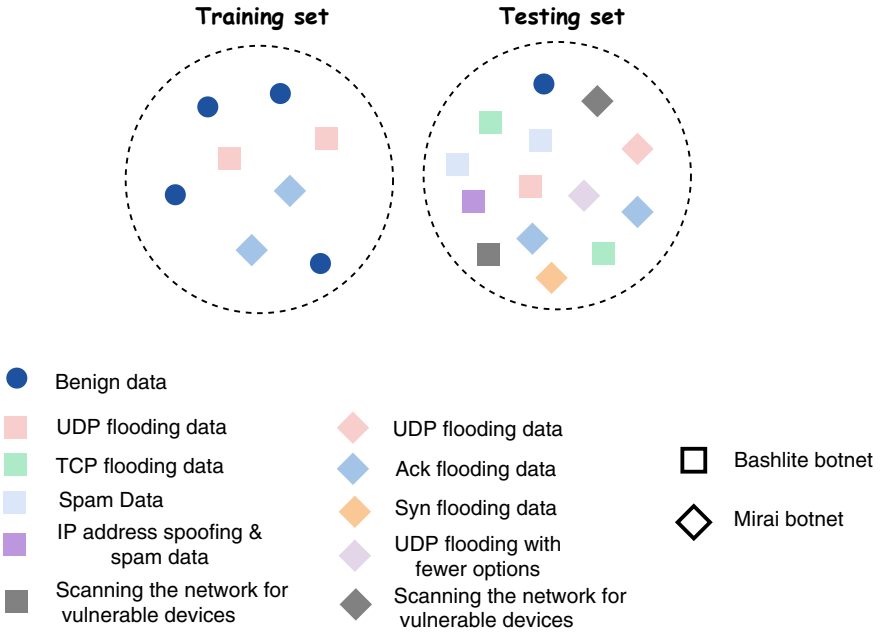


Fig. 4 Training process

- Connecting the devices via Wi-Fi to various access points.
- Recording behavioral snapshots of the network packets using Wireshark software and port mirroring technique.
- Extracting 115 traffic statistical features.

The statistical features are time intervals between packet arrivals, packet sizes, and counts. For each device, the data are obtained under both normal working conditions, and ten different attacks are performed by Bashlite and Mirai botnets. The conducted attacks are like UDP, TCP, ACK, SYN flooding, sending spam data, and scanning for vulnerable devices [43].

The novel approach that we are adopting is that the training set for each dataset consisting two different types of attacks, whereas the testing set contains ten different types of attacks. In other words, the algorithm will be trained on just two types of attacks and will be tested on ten, in which two attacks of them the algorithm has already trained on while the remaining eight attacks are unseen as attacks. Figure 4 illustrates the designed process for training and testing data.

Furthermore, it is important to note that the training set has a balance class ratio. In contrast, the testing set is imbalance, in which the ratio of normal traffic to malicious traffic is 1:13. Table 2 presents a description of the constructed datasets.

Table 2 Datasets specifications

	Training set		Testing set	
	Benign-to-malicious ratio	Types of attacks	Benign-to-malicious ratio	Number of attacks
Dataset 1	1:1	UDP flooding, spam data	1:13	10
Dataset 2	1:1	UDP and TCP flooding	1:13	10
Dataset 3	1:1	Scan for vulnerable devices, SYN flooding	1:13	10
Dataset 4	1:1	UDP and ACK flooding	1:13	10
Dataset 5	1:1	UDP flooding with fewer options, TCP flooding	1:13	10

4 Experimental Results and Discussion

This section illustrates the used evaluation metrics as well as a discussion of the obtained results, in both of comparing MOPSO vs NSGA-II, versus traditional machine learning algorithms, and vs filter-based methods.

4.1 Evaluation Measures

Five evaluation measures are used to compare MOPSO with other methods. They are the false alarm rate, true-positive rate, true-negative rate, geometric mean, and the area under curve [27].

False alarm rate (FAR) is the percentage of classifying normal instances as malicious instances (Eq. 7).

$$FAR = \frac{FP}{FP + TN} \tag{7}$$

True-negative rate (TNR) is also known as specificity (Eq. 8).

$$TNR = \frac{TN}{FP + TN} \tag{8}$$

True-positive rate (TPR) is known as sensitivity and probability of detection rate (Eq. 9).

$$TPR = \frac{TP}{FN + TP} \tag{9}$$

G-mean is known as the geometric mean (Eq. 10).

$$G\text{-}mean = \sqrt{specificity.sensitivity} \tag{10}$$

Area under curve (AUC) is the area under the ROC curve (Eq. 11).

$$AUC = \frac{(1 - FPR) * (1 + TPR)}{2} + \frac{FPR * TPR}{2} \tag{11}$$

4.2 Results and Discussions

The investigation of MOPSO performance in the field of IoT botnet detection is done by first comparing MOPSO against the NSGA-II algorithm. Furthermore, the results of MOPSO are verified by comparing it with traditional classifiers (zero rule classifier, k-nearest neighbor (K-NN), decision tree (DT), and multilayer perceptron (MLP)), and filter-based feature selection methods.

4.2.1 Comparison with NSGA-II

Table 3 compares the performance of MOPSO with NSGA-II over five IoT datasets using the aforementioned performance measures (FAR, TNR, TPR, G-mean, and AUC). The results show the superiority of MOPSO over NSGA-II in four datasets (2, 3, 4, and 5). In datasets 3, 4, and 5, MOPSO achieved better results than NSGA-II in all measures. Considering AUC values, for instance, MOPSO obtains 0.878 versus 0.808 in dataset 3, similarly in dataset 4; MOPSO reaches 0.911, whereas NSGA-II reaches 0.867. Interestingly, in dataset 4 the results show how much the two algorithms are competitive, yet a trivial increase of the performance in favor to MOPSO is noticeable. In such cases, the TPR is (1.000 vs. 0.990). Inspecting the results on dataset 2, it is evident that NSGA-II does slightly better according to FAR and TNR, yet MOPSO performs better in the rest of measures. Despite the encouraging results obtained so far in favor to MOPSO, NSGA-II performs slightly better than MOPSO in dataset 1. Overall, we see highly competitive results of the swarm-based algorithm (MOPSO) against the evolutionary-based algorithm (NSGA-II), while MOPSO tends to have the upper hand.

Figure 5 shows the error rate (y-axis) against the number of selected features (x-axis), in which each sub-figure corresponds to a different dataset. The curves show the average Pareto front of both MOPSO and NSGA-II. Obviously, NSGA-II

Table 3 MOPSO versus NSGA-II performance results

		FAR	TNR	TPR	G-mean	AUC
Dataset 1	MOPSO	0.070	0.930	0.982	0.955	0.956
	NSGA-II	**0.044**	**0.956**	**0.986**	**0.970**	**0.971**
Dataset 2	MOPSO	0.059	0.941	**0.982**	**0.961**	**0.962**
	NSGA-II	**0.047**	**0.953**	0.943	0.947	0.948
Dataset 3	MOPSO	**0.222**	**0.778**	**0.978**	**0.869**	**0.878**
	NSGA-II	0.280	0.720	0.895	0.797	0.808
Dataset 4	MOPSO	**0.120**	**0.880**	**1.000**	**0.936**	**0.940**
	NSGA-II	0.148	0.852	0.990	0.915	0.921
Dataset 5	MOPSO	**0.177**	**0.823**	**0.999**	**0.902**	**0.911**
	NSGA-II	0.250	0.750	0.984	0.854	0.867

holds fewer numbers of features than MOPSO in all datasets, but it obtains higher average error rate values in datasets 3 and 5 than MOPSO. Although MOPSO has relatively higher number of features than NSGA-II, it obtains low error rate values as it is obvious in sub-figure 2. Furthermore, there are fluctuations of the error rate over the number of features at sub-figures 1 and 4, but at some points MOPSO records lower values of error rate than NSGA-II.

4.2.2 Comparison with Traditional Machine Learning Algorithms

We compare MOPSO against four classifiers—zero rule classifier (ZeroR) [27], K-NN, DT (decision-regression tree learner REPTree [26]), and multilayer perceptron (voted perceptron [24]) (MLP). ZeroR predicts the majority class in the training dataset; hence, we use it as a baseline performance for other classification methods [27]. We have run the experiments on Weka workbench [26], using algorithms' default parameters settings as defined by Weka.

Tables 4, 5, 6, 7, and 8 show the comparison results between MOPSO against ZeroR, K-NN, DT, and MLP classifiers. Note that the boldface highlights the best results in all tables. Obviously, MOPSO outperforms the baseline classifier ZeroR in all datasets in terms of G-mean, AUC, and TPR. As well as MOPSO outperforms 1-NN, DT, and MLP in datasets 1 and 4 in terms of TPR, G-mean, and AUC, achieving (0.982, 0.955, 0.956), (1.000, 0.938, 0.940), respectively. In addition, MOPSO holds the best TPR in datasets 3 and 5, obtaining (0.978, 0999), respectively. It can be seen that in all datasets, 1-NN and DT perform slightly better than MOPSO in terms of FAR and TNR, whereas MLP reaches better FAR and TNR than MOPSO in datasets 3 and 4. In contrast, in dataset 2 MOPSO outperforms other classifiers in terms of AUC and G-mean reaching a peak at (0.962, 0.961), respectively. Overall, we can conclude that MOPSO is a competitive classifier and optimization algorithm against other single-objective classifiers.

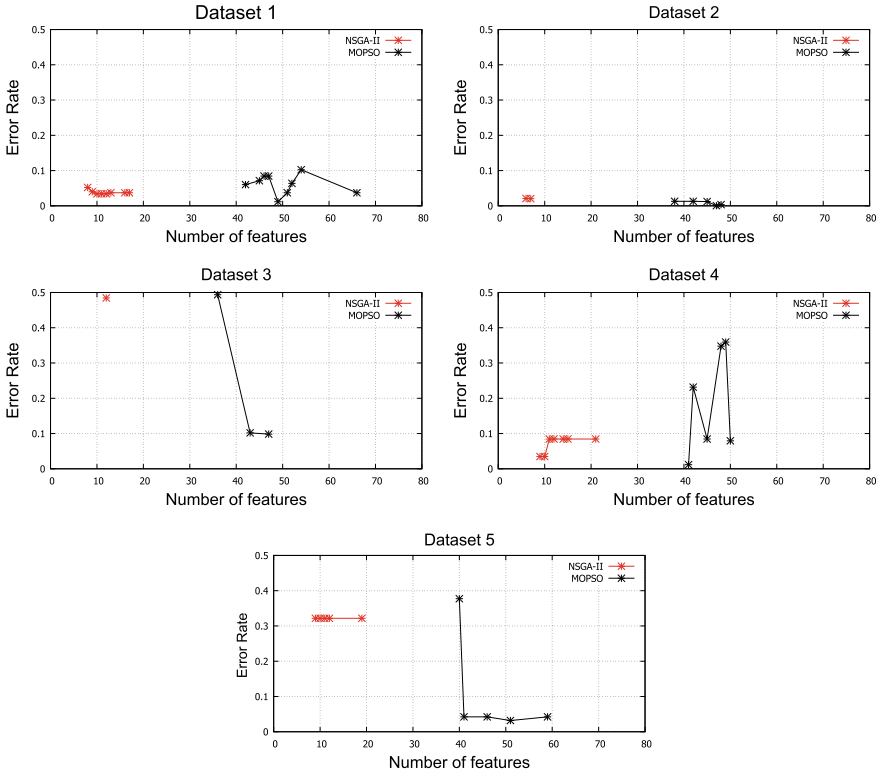


Fig. 5 Comparison of MOPSO and NSGA-II over all datasets

Table 4 Dataset 1 performance results

	FAR	TNR	TPR	G-mean	AUC
MOPSO	0.070	0.930	**0.982**	**0.955**	**0.956**
ZeroR	0.000	1.000	0.000	0.000	0.500
1-NN	0.000	1.000	0.763	0.873	0.882
DT	0.016	0.984	0.908	0.945	0.946
MLP	0.102	0.898	0.546	0.700	0.722

Table 5 Dataset 2 performance results

	FAR	TNR	TPR	G-mean	AUC
MOPSO	0.059	0.941	0.982	**0.961**	**0.962**
ZeroR	0.000	1.000	0.000	0.000	0.500
1-NN	0.000	1.000	0.596	0.772	0.798
DT	0.016	0.984	0.738	0.852	0.861
MLP	0.578	0.422	0.988	0.646	0.705

Table 6 Dataset 3 performance results

	FAR	TNR	TPR	G-mean	AUC
MOPSO	0.222	0.778	**0.978**	0.872	0.878
ZeroR	0.000	1.000	0.000	0.000	0.500
1-NN	0.000	1.000	0.711	0.843	0.856
DT	0.016	0.984	0.873	0.927	0.929
MLP	0.094	0.906	0.473	0.655	0.690

Table 7 Dataset 4 performance results

	FAR	TNR	TPR	G-mean	AUC
MOPSO	0.120	0.880	**1.000**	**0.938**	**0.940**
ZeroR	0.000	1.000	0.000	0.000	0.500
1-NN	0.000	1.000	0.636	0.797	0.818
DT	0.000	1.000	0.604	0.777	0.802
MLP	0.094	0.906	0.473	0.655	0.690

Table 8 Dataset 5 performance results

	FAR	TNR	TPR	G-mean	AUC
MOPSO	0.177	0.823	**0.999**	0.907	0.911
ZeroR	0.000	1.000	0.000	0.000	0.500
1-NN	0.000	1.000	0.668	0.817	0.834
DT	0.000	1.000	0.847	0.920	0.924
MLP	0.367	0.633	0.935	0.769	0.784

4.2.3 Comparisons with Filter FS Methods

In this sub-section, we compare the performance of MOPSO with conventional filter-based feature selection methods (ReleifF, correlation-based, information gain, and symmetrical filter-based methods) over all the datasets. The class of filter-based FS methods filters out features independently on the induction algorithm. The filtering criteria evaluate each feature individually using its, for example, Pearson's correlation, or symmetrical uncertainty with the target class [12]. We use the K-NN (k=5) as the induction algorithm.

All the experiments in this sub-section have been conducted on Weka [26]. Table 9 shows a comparison of the classification error rate of MOPSO with several conventional filter-based methods. It can be seen that MOPSO achieved the lowest error values at datasets 3, 4, and 5 obtained 0.207, 0.111, and 0.164, respectively. In dataset 1, however, the correlation-based filter achieved the minimum value which is 0.001. In dataset 2, although the symmetrical filter achieved 0.048, it is relatively close to MOPSO value which is 0.56. Furthermore, Tables 10, 11, 12, 13, and 14

Table 9 Comparison between MOPSO classification error with filter-based feature selection methods

	ReleifF	Correlation	InfoGain	Symmetrical	MOPSO
Dataset 1	0.071	**0.001**	0.011	0.011	0.066
Dataset 2	0.137	0.082	0.119	**0.048**	0.056
Dataset 3	0.322	0.318	0.276	0.276	**0.207**
Dataset 4	0.345	0.341	0.297	0.297	**0.111**
Dataset 5	0.274	0.230	0.264	0.263	**0.164**

Table 10 Dataset 1 classification measures of MOPSO versus filter-based feature selection methods

	FAR	TNR	TPR	G-mean	AUC
ReleifF	0.219	0.781	0.941	0.857	0.861
Correlation	**0.000**	**1.000**	**0.999**	**0.999**	**1.000**
InfoGain	**0.000**	**1.000**	0.988	0.994	0.994
Symmetrical	**0.000**	**1.000**	0.988	0.994	0.994
MOPSO	0.070	0.930	0.982	0.956	0.956

Table 11 Dataset 2 classification measures of MOPSO versus filter-based feature selection methods

	FAR	TNR	TPR	G-mean	AUC
ReleifF	**0.000**	**1.000**	0.854	0.924	0.927
Correlation	**0.000**	**1.000**	0.911	0.954	0.956
InfoGain	**0.000**	**1.000**	0.871	0.933	0.936
Symmetrical	**0.000**	**1.000**	0.948	**0.974**	**0.974**
MOPSO	0.059	0.941	**0.982**	0.962	0.962

Table 12 Dataset 3 classification measures of MOPSO versus filter-based feature selection methods

	FAR	TNR	TPR	G-mean	AUC
ReleifF	**0.000**	**1.000**	0.653	0.808	0.827
Correlation	**0.000**	**1.000**	0.657	0.811	0.829
InfoGain	**0.000**	**1.000**	0.702	0.838	0.851
Symmetrical	**0.000**	**1.000**	0.702	0.838	0.851
MOPSO	0.222	0.778	**0.978**	**0.872**	**0.878**

Table 13 Dataset 4 classification measures of MOPSO versus filter-based feature selection methods

	FAR	TNR	TPR	G-mean	AUC
ReleifF	**0.000**	**1.000**	0.628	0.792	0.814
Correlation	**0.000**	**1.000**	0.632	0.795	0.816
InfoGain	**0.000**	**1.000**	0.679	0.824	0.840
Symmetrical	**0.000**	**1.000**	0.679	0.824	0.840
MOPSO	0.120	0.880	**1.000**	**0.938**	**0.940**

Table 14 Dataset 5 classification measures of MOPSO versus filter-based feature selection methods

	FAR	TNR	TPR	G-mean	AUC
ReleifF	**0.000**	**1.000**	0.704	0.839	0.852
Correlation	**0.000**	**1.000**	0.751	0.867	0.876
InfoGain	**0.000**	**1.000**	0.714	0.845	0.857
Symmetrical	**0.000**	**1.000**	0.716	0.846	0.858
MOPSO	0.177	0.823	**0.999**	**0.907**	**0.911**

present the performance measures of classification among MOPSO and the aforementioned filter methods. The comparison is over all the five datasets in regard to FAR, TNR, TPR, G-mean, and AUC. It can be observed from the tables that MOPSO outperforms other filter methods in datasets 3, 4, and 5 in terms of TPR, G-mean, and AUC. It is evident that MOPSO outperforms other (considering only TPR) on dataset 2. In dataset 1, the correlation filter outperforms all other filter methods as well as MOPSO, where it obtains regarding the FAR, TPR, and AUC as 0.000, 0.999, and 1.000, respectively. Generally, MOPSO outperformed the used filter methods in most of the cases.

In summary, the results of this chapter show that MOPSO is able to outperform NSGA-II when minimizing the error and number of features in feature selection problems. This algorithm also showed competitive, often superior, results as compared to conventional feature selection problems. Therefore, we state that this algorithm has merits to be considered as a feature selection technique. Since we maintained the multi-objective formulation, there should be another step for decision making though.

5 Conclusion and Future Work

IoT devices are potentially in danger of extensive attacks. Different IoT malwares have been emerged and triggered large-scale malicious attacks in the last decade such as Mirai and Bashlite. This chapter analyzed the performance of multi-objective particle swarm optimization for the detection or classification of IoT traffic as attack or normal. The problem formulated for MOPSO had a discrete search space, so we would have to develop a binary version of this algorithm.

We constructed new five datasets from the original datasets that were drawn from UCI repository. The MOPSO algorithm was compared with NSGA-II and a large number of conventional feature selection algorithms. As per the obtained results, MOPSO outperformed NSGA-II in regard to false alarm rate, detection rate, G-mean, and AUC in IoT malicious network traffic. Moreover, MOPSO outperformed different machine learning classifiers as ZeroR, K-NN, decision tree, and MLP. MOPSO, as wrapper-based FS method, showed promising results as per comparing it with conventional filter-based methods. For future works, we would like to test MOPSO against other multi-objective evolutionary algorithms like SPEA-II, PESA-II, MODE, and NSGA-III. Also, one of the drawbacks of wrapper-based methods over filter-based methods is the high computational cost, which should be reduced for computationally expensive datasets. An automatic decision-making mechanism is recommended to be developed as well to choose one of the Pareto optimal solutions obtained by MOPSO.

Appendix

IoT Datasets

Dataset	Class		No. of instances	Rate (%)
Security camera XCS7_1003	Normal		19,529	2.40
	Mirai botnet	ACK flooding	107,188	13.15
		Scan	43,675	5.36
		SYN flooding	122,480	15.02
		UDP flooding	157,085	19.27
		UDP plain flooding	48,837	5.99
	Sum		47,9265	58.79
	Gafgyt botnet	Combo (spam data)	59,399	7.29
		Junk (spam data)	27,414	3.36
		Scan	28,573	3.50
		TCP flooding	98,076	12.03
		UDP flooding	102,981	12.63
	Sum		316,443	38.82
	Total sum		815,237	100.00

Dataset	Class		No. of instances	Rate (%)
Baby monitor (Philips_B120N10)	**Normal**		**175,241**	**15.95**
	Mirai botnet	ACK flooding	91, 124	8.29
		Scan	103, 622	9.43
		SYN flooding	118, 129	10.75
		UDP flooding	217, 035	19.75
		UDP plain flooding	80, 809	7.36
	Sum		**610,719**	**55.59**
	Gafgyt botnet	Combo (spam data)	58, 153	5.29
		Junk (spam data)	28, 350	2.58
		Scan	27, 860	2.54
		TCP flooding	92, 582	8.43
		UDP flooding	105, 783	9.63
	Sum		**312,728**	**28.46**
	Total sum		**1,098,688**	**100.00**
Danmini doorbell	**Normal**		**49,549**	**4.87**
	Mirai botnet	ACK flooding	102, 196	10.04
		Scan	107, 686	10.57
		SYN flooding	122, 574	12.04
		UDP flooding	237, 666	23.34
		UDP plain flooding	81, 983	8.05
	Sum		**652,105**	**64.04**
	Gafgyt botnet	Combo (spam data)	59, 719	5.86
		Junk (spam data)	29, 069	2.85
		Scan	29, 850	2.93
		TCP flooding	92, 142	9.05
		UDP flooding	105, 875	10.40
	Sum		**316,655**	**31.10**
	Total sum		**1,018,309**	**100.00**

Dataset	Class		No. of instances	Rate (%)
Ennio doorbell	**Normal**		**39,101**	**11.00**
	Mirai botnet	ACK flooding	0	0.00
		Scan	0	0.00
		SYN flooding	0	0.00
		UDP flooding	0	0.00
		UDP plain flooding	0	0.00
	Sum		**0**	**0.00**
	Gafgyt botnet	Combo (spam data)	53,015	14.91
		Junk (spam data)	29,798	8.38
		Scan	28,121	7.91
		TCP flooding	10,1537	28.56
		UDP flooding	103,934	29.24
	Sum		**316,405**	**89.00**
	Total sum		**355,506**	**100.00**
Ecobee thermostat	**Normal**		**13,114**	**1.57**
	Mirai botnet	ACK flooding	113,286	13.55
		Scan	43,193	5.17
		SYN flooding	116,808	13.97
		UDP flooding	151,482	18.12
		UDP plain flooding	87,369	10.45
	Sum		**512,138**	**61.27**
	Gafgyt botnet	Combo (spam data)	53,013	6.34
		Junk (spam data)	30,313	3.63
		Scan	27,495	3.29
		TCP flooding	95,022	11.37
		UDP flooding	104,792	12.54
	Sum		**310,635**	**37.16**
	Total sum		**835,887**	**100.00**

Dataset	Class		No. of instances	Rate (%)
Samsung webcam (SNH_1011_N)	**Normal**		**52,151**	**13.90**
	Mirai botnet	ACK flooding	0	0.00
		Scan	0	0.00
		SYN flooding	0	0.00
		UDP flooding	0	0.00
		UDP plain flooding	0	0.00
	Sum		**0**	**0.00**
	Gafgyt botnet	Combo (spam data)	58,670	15.64
		Junk (spam data)	28,306	7.54
		Scan	27,699	7.38
		TCP flooding	97,784	26.06
		UDP flooding	110,618	29.48
	Sum		**323,077**	**86.10**
	Total sum		**375,228**	**100.00**
Security camera PT_737E	**Normal**		**62,155**	**7.50**
	Mirai botnet	ACK flooding	60,555	7.31
		Scan	96,782	11.68
		SYN flooding	65,747	7.94
		UDP flooding	156,249	18.86
		UDP plain flooding	56,682	6.84
	Sum		**436,015**	**52.64**
	Gafgyt botnet	Combo (spam data)	61,381	7.41
		Junk (spam data)	30,899	3.73
		Scan	29,298	3.54
		TCP flooding	104,511	12.62
		UDP flooding	104,012	12.56
	Sum		**330,101**	**39.85**
	Total sum		**828,271**	**100.00**

Dataset	Class		No. of instances	Rate (%)
Security camera PT_838	**Normal**		**98,515**	**11.77**
	Mirai botnet	ACK flooding	57, 998	6.93
		Scan	97, 097	11.60
		SYN flooding	61, 852	7.39
		UDP flooding	158, 609	18.95
		UDP plain flooding	53, 786	6.43
	Sum		**429,342**	**51.30**
	Gafgyt botnet	Combo (spam data)	57, 531	6.87
		Junk (spam data)	29, 069	3.47
		Scan	28, 398	3.39
		TCP flooding	89, 388	10.68
		UDP flooding	104, 659	12.51
	Sum		**309,045**	**36.93**
	Total sum		**836,902**	**100.00**
Security camera XCS7_1002	**Normal**		**46,586**	**5.62**
	Mirai botnet	ACK flooding	107, 188	12.93
		Scan	43, 675	5.27
		SYN flooding	122, 480	14.77
		UDP flooding	157, 085	18.95
		UDP plain flooding	48, 837	5.89
	Sum		**479,265**	**57.81**
	Gafgyt botnet	Combo (spam data)	54, 284	6.55
		Junk (spam data)	28, 580	3.45
		Scan	27, 826	3.36
		TCP flooding	88, 817	10.71
		UDP flooding	103, 721	12.51
	Sum		**303,228**	**36.57**
	Total sum		**829,079**	**100.00**

References

1. Ahmed S, Mafarja M, Faris H, Aljarah I (2018) Feature selection using salp swarm algorithm with chaos. In: Proceedings of the 2nd international conference on intelligent systems, metaheuristics & swarm intelligence. ACM, pp 65–69
2. Al-Dabagh MZN, Alhabib MHM, AL-Mukhtar FH (2018) Face recognition system based on kernel discriminant analysis k-nearest neighbor and support vector machine. Int J Res Eng 5(3):335–338
3. Aljarah I, Al-Zoubi AM, Faris H, Hassonah MA, Mirjalili S, Saadeh H (2018) Simultaneous feature selection and support vector machine optimization using the grasshopper optimization algorithm. Cogn Comput 1–18
4. Aljarah I, Ludwig SA (2013) Mapreduce intrusion detection system based on a particle swarm optimization clustering algorithm. In: 2013 IEEE congress on evolutionary computation. IEEE, pp 955–962
5. Aljarah I, Ludwig SA (2013) Towards a scalable intrusion detection system based on parallel pso clustering using mapreduce. In: Proceedings of the 15th annual conference companion on Genetic and evolutionary computation. ACM, pp 169–170
6. Aljarah I, Mafarja M, Heidari AA, Faris H, Zhang Y, Mirjalili S (2018) Asynchronous accelerating multi-leader salp chains for feature selection. Appl Soft Comput 71:964–979
7. Angrishi K (2017) Turning internet of things (iot) into internet of vulnerabilities (iov): Iot botnets. *arXiv preprint* arXiv:1702.03681
8. Antonakakis M, April T, Bailey M, Bernhard M, Bursztein E, Cochran J, Durumeric Z, Halderman JA, Invernizzi L, Kallitsis M et al (2017) Understanding the mirai botnet. In: USENIX security symposium, pp 1092–1110
9. Atallah DM, Badawy M, El-Sayed A, Ghoneim MA (2019) Predicting kidney transplantation outcome based on hybrid feature selection and knn classifier. Multimed Tools Appl 1–25
10. bin Mohd Zain MZ, Kanesan J, Chuah JH, Dhanapal S, Kendall G (2018) A multi-objective particle swarm optimization algorithm based on dynamic boundary search for constrained optimization. Appl Soft Comput
11. Bramer M (2007) Principles of data mining, vol 180. Springer
12. Chandrashekar G, Sahin F (2014) A survey on feature selection methods. Appl Soft Comput 40(1):16–28
13. Coello CAC, Pulido GT, Lechuga MS (2004) Handling multiple objectives with particle swarm optimization. IEEE Trans Evol Comput 8(3):256–279
14. Conti M, Dehghantanha A, Franke K, Watson S (2018). Challenges and opportunities. Internet Things Secur Forensics
15. Cormen TH, Leiserson CE, Rivest RL, Stein C (2009) Introduction to algorithms. MIT press
16. Dua D, Efi KT (2017) UCI machine learning repository
17. Eberhart R, Kennedy J (1995) A new optimizer using particle swarm theory. In: Micro machine and human science, 1995. MHS'95., Proceedings of the sixth international symposium on. IEEE, pp 39–43
18. Elrawy MF, Awad AI, Hamed HFA (2018) Intrusion detection systems for iot-based smart environments: a survey. J Cloud Comput 7(1):21
19. Faris Al-Zoubi AM, Heidari AA, Aljarah I, Mafarja M, Hassonah MA, Fujita H (2019) An intelligent system for spam detection and identification of the most relevant features based on evolutionary random weight networks. Inf Fusion 48:67–83
20. Faris H, Aljarah I, Al-Shboul B (2016) A hybrid approach based on particle swarm optimization and random forests for e-mail spam filtering. In: International conference on computational collective intelligence. Springer, pp 498–508
21. Faris H, Aljarah I et al (2015) Optimizing feedforward neural networks using krill herd algorithm for e-mail spam detection. In:2015 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT). IEEE, pp 1–5

22. Faris H, Hassonah MA, Al-Zoubi AM, Mirjalili S, Aljarah I (2018) A multi-verse optimizer approach for feature selection and optimizing svm parameters based on a robust system architecture. Neural Comput Appl 30(8):2355–2369
23. Faris H, Mafarja MM, Heidari AA, Aljarah I, Al-Zoubi AM, Mirjalili S, Fujita H (2018) An efficient binary salp swarm algorithm with crossover scheme for feature selection problems. Knowl-Based Syst 154:43–67
24. Freund Y, Schapire RE (1999) Large margin classification using the perceptron algorithm. Mach Learn 37(3):277–296
25. Garcia-Teodoro P, Diaz-Verdejo J, Maciá-Fernández G, Vázquez E (2009) Anomaly-based network intrusion detection: techniques, systems and challenges. Comput & Secur 28(1–2):18–28
26. Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH (2009) The weka data mining software: an update. ACM SIGKDD Explor Newsl 11(1):10–18
27. Han J, Pei J, Kamber M (2011) Data mining: concepts and techniques. Elsevier
28. Hemdan EE-D, Manjaiah DH (2018) Cybercrimes investigation and intrusion detection in internet of things based on data science methods. In: Cognitive computing for big data systems over IoT. Springer, pp 39–62
29. Jing Q, Vasilakos AV, Wan J, Lu J, Qiu D (2014) Security of the internet of things: perspectives and challenges. Wirel Netw 20(8):2481–2501
30. Kesavamoorthy R, Soundar KR (2018) Swarm intelligence based autonomous ddos attack detection and defense using multi agent system. Clust Comput 1–8
31. Kolias C, Kambourakis G, Stavrou A, Voas J (2017) Ddos in the iot: mirai and other botnets. Computer 50(7):80–84
32. Kowshalya MA, Valarmathi ML (2016) Detection of sybil's across communities over social internet of things. J Appl Eng Sci 14(1):75–83
33. Kuhn M, Johnson K (2013) Applied predictive modeling, vol 26. Springer
34. Li J, Zhao Z, Li R, Zhang H, Zhang T (2018) Ai-based two-stage intrusion detection for software defined iot networks. IEEE Internet Things J
35. Liu L, Xu B, Wu Zhang XX (2018) An intrusion detection method for internet of things based on suppressed fuzzy clustering. EURASIP J Wirel Commun Netw 1:113
36. Mafarja M, Aljarah I, Faris H, Hammouri AI, Al-Zoubi AM, Mirjalili S (2019) Binary grasshopper optimisation algorithm approaches for feature selection problems. Expert Syst Appl 117:267–286
37. Mafarja M, Aljarah I, Heidari AA, Faris H, Fournier-Viger P, Li X, Mirjalili S (2018) Binary dragonfly optimization for feature selection using time-varying transfer functions. Knowl-Based Syst 161:185–204
38. Mafarja M, Aljarah I, Heidari AA, Hammouri AI, Faris H, Al-Zoubi AM, Mirjalili S (2018) Evolutionary population dynamics and grasshopper optimization approaches for feature selection problems. Knowl-Based Syst 145:25–45
39. Mafarja M, Heidari AA, Faris H, Mirjalili S, Aljarah I (2020) Dragonfly algorithm: theory, literature review, and application in feature selection. In: Nature-inspired optimizers. Springer, pp 47–67
40. Mafarja MM, Mirjalili S (2018) Hybrid binary ant lion optimizer with rough set and approximate entropy reducts for feature selection. Soft Comput 1–17
41. Marzano A, Alexander D, Fonseca O, Fazzion E, Steding-Jessen K, Chaves MHPC, Cunha Í, Guedes D, Meira W (2018) The evolution of bashlite and mirai iot botnets. In: 2018 IEEE symposium on computers and communications (ISCC). IEEE, pp 00813–00818
42. Mehmood A, Mukherjee M, Ahmed SH, Song H, Malik KM (2018) Nbc-maids: naïve bayesian classification technique in multi-agent system-enriched ids for securing iot against ddos attacks. J Supercomput 1–15
43. Meidan Y, Bohadana M, Mathov Y, Mirsky Y, Shabtai A, Breitenbacher D, Elovici Y (2018) N-baiot network-based detection of iot botnet attacks using deep autoencoders. IEEE Pervasive Comput 17(3):12–22

44. Mir A, Nasiri JA (2018) Knn-based least squares twin support vector machine for pattern classification. Appl Intell 48(12):4551–4564
45. Mirjalili S, Lewis A (2013) S-shaped versus v-shaped transfer functions for binary particle swarm optimization. Swarm Evol Comput 9:1–14
46. Mohemmed AW, Zhang M (2008) Evaluation of particle swarm optimization based centroid classifier with different distance metrics. In: 2008 IEEE congress on evolutionary computation (IEEE world congress on computational intelligence). IEEE, pp 2929–2932
47. Moustafa N, Turnbull B, Choo K-KR (2018) An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of internet of things. EEE Internet Things J
48. Pamukov ME, Poulkov VK, Shterev VA (2018) Negative selection and neural network based algorithm for intrusion detection in iot. In: 2018 41st international conference on telecommunications and signal processing (TSP). IEEE, pp 1–5
49. Rana S, Hossain S, Shoun HI, Abul Kashem M (2018) An effective lightweight cryptographic algorithm to secure resource-constrained devices. Int J Adv Comput Sci Appl 9(11):267–275
50. Rathore S, Park JH (2018) Semi-supervised learning based distributed attack detection framework for iot. Appl Soft Comput 72:79–89
51. Sanchez-Pi N, Martí L, Molina JM (2018) Applying voreal for iot intrusion detection. In: International Conference on Hybrid Artificial Intelligence Systems. Springer, pp 363–374
52. Selvarani P, Suresh A, Malarvizhi N (2018) Secure and optimal authentication framework for cloud management using hgapso algorithm. Clust Comput 1–10
53. Shaikh F, Bou-Harb E, Crichigno J, Ghani N (2018) A machine learning model for classifying unsolicited iot devices by observing network telescopes. In: 2018 14th international wireless communications & mobile computing conference (IWCMC). IEEE, pp 938–943
54. Vijayalakshmi J, Robin CRR (2018) An exponent based error detection mechanism against dxdos attack for improving the security in cloud. Clust Comput 1–10
55. Whitter-Jones J (2018) Security review on the internet of things. In: 2018 Third international conference on fog and mobile edge computing (FMEC). IEEE, pp 163–168
56. Xiao L, Wan X, Lu X, Zhang Y, Wu D (2018) Iot security techniques based on machine learning: how do iot devices use ai to enhance security? IEEE Signal Process Mag 35(5):41–49
57. Xue B, Zhang M, Browne WN (2013) Particle swarm optimization for feature selection in classification: a multi-objective approach. IEEE Trans Cybern 43(6):1656–1671
58. Yang X-S (2010) A new metaheuristic bat-inspired algorithm. In: Nature inspired cooperative strategies for optimization (NICSO 2010). Springer, pp 65–74
59. Zhang H, Sun G (2002) Feature selection using tabu search method. Pattern Recognit 35(3):701–711

Evolutionary and Swarm-Based Feature Selection for Imbalanced Data Classification

Feras Namous, Hossam Faris, Ali Asghar Heidari, Monther Khalafat, Rami S. Alkhawaldeh and Nazeeh Ghatasheh

Abstract Recently, feature selection task has gained more attention in classification of problems. This task aims to find the most important features in a large search space of potential solutions. Hence, a challenging problem is manifested to find the optimal solution. In this paper, we study a metaheuristic-based approach for feature selection in binary classification problems. The scenario deals with several highly imbalanced datasets. In an attempt to handle the problem of imbalanced data, the common fitness function based on the classification accuracy is replaced with two more effective fitness functions: the area under the ROC curve and the geometric mean. To evaluate the effectiveness of the developed approach, two popular metaheuristic approaches are experimented with the three fitness functions for classifying six imbalanced datasets. The chapter discusses the impact of the used fitness function on the final

F. Namous · H. Faris (✉) · M. Khalafat
King Abdullah II School for Information Technology,
The University of Jordan, Amman, Jordan
e-mail: hossam.faris@ju.edu.jo

F. Namous
e-mail: ferasnamous@gmail.com

M. Khalafat
e-mail: Montherk.1987@gmail.com

A. A. Heidari
School of Surveying and Geospatial Engineering, College of Engineering,
University of Tehran, Tehran, Iran
e-mail: as_heidari@ut.ac.ir; aliasgha@comp.nus.edu.sg; t0917038@u.nus.edu

A. A. Heidari
Department of Computer Science, School of Computing, National University of Singapore,
Singapore, Singapore

R. S. Alkhawaldeh · N. Ghatasheh
Faculty of Information Technology and Systems, The University of Jordan, Aqaba, Jordan
e-mail: r.alkhawaldeh@ju.edu.jo

N. Ghatasheh
e-mail: n.ghatasheh@ju.edu.jo

S. Mirjalili et al. (eds.), *Evolutionary Machine Learning Techniques*,
Algorithms for Intelligent Systems, https://doi.org/10.1007/978-981-32-9990-0_11

performance of the proposed methods. The proposed methods demonstrated that some fitness functions like the accuracy rate can mislead the identification process of the relevant features in imbalanced datasets.

Keywords Feature selection · Evolutionary neural networks · Imbalanced data · Classification

1 Introduction

Metaheuristics and evolutionary methods are algorithms that aim to find a near-optimal solution without using gradient information [12]. These methods can be categorized into two families in terms of the number of solutions (candidates) to be processed in one iteration of the optimization process; single-solution-based algorithms and population-based algorithms. In the former family, only one candidate can be manipulated in each iteration, while in the later, a set of candidates can be processed in each iteration. There are more possible ways to classify optimizers. For instance, they can be fairly classified based on general metaphor and unique underlying mechanisms. This 'meta' term in the name shows that these algorithms tried to simulate similar intelligent behaviors observed in nature. In the other side, we have nature-inspired optimizers that use new mathematical components and show superior performance. This level of optimizers is the desired level in the heuristic area. Also, no free lunch (NFL) theorem [14] encourages all researchers to search for a new or modified optimizer for dealing with new problems because there is no ranked one optimizer if we consider average performance on all possible problems.

The initial idea of all swarm and evolutionary optimizers is to evolve the candidate solutions based on the existing knowledge on the problem itself and reach a better solution without any dependency to gradient information. This knowledge can be obtained by evaluating the fitness function for all solutions to detect which one is best. Genetic algorithm (GA) [13] is the first attempt that uses the idea of *Darwin's theory of evolution* in evolving the creatures and solves optimization problems without gradient information. The well-known GA has three main core bio-inspired components that are selection, crossover, and mutation. A classic GA only requires some initial information to solve the problem. First, a genetic representation of the solutions is needed to feed the input and make a pool of genes. Second, we need the equation of the objective function to evaluate the quality of solutions in terms of a specific metric. Hence, the evaluation metric is one of the most critical components in the procedure of GA.

To simulate the social behaviors of birds, another well-known swarm intelligence optimizer proposed in 1995 [3, 7]. This method called particle swarm optimization (PSO), and it inherits a different mechanism compared to GA to move the candidate solutions toward the *personal best* and *global best* solutions in each iteration. PSO was successfully applied to solve NP-hard optimization problems [9].

Evolutionary and swarm-based algorithms have been utilized in different complex machine learning tasks [4, 8]. One of their popular applications in this context is to perform the search process for finding the best representative set of features in the feature selection task in machine learning. In literature, these optimization algorithms were commonly deployed as a component of the wrapper feature selection methods. Wrapper methods are a type of feature selection that incorporates three main components which are the search algorithm, an induction algorithm, and an evaluation criteria. In order to evaluate the subsets of features searched by the search algorithm, the induction algorithm is trained based on each candidate set of features and evaluated using a predefined criteria.

In this chapter, we develop a wrapper-based feature selection framework through which we study the impact of the selection of evaluation component on the classification results in an imbalanced data distribution. As a search algorithm two metaheuristics are investigated: an evolutionary algorithm and a variant of PSO. While for fitness evaluation accuracy, geometric mean, and the area under ROC curve are investigated. Moreover, the framework incorporates a simple technique to indicate the importance of the input features. A series of experiments are conducted to study how the fitness evaluation component could affect both the classification results and the identification of the important variables when the training dataset is imbalanced.

The rest of the chapter is organized as follows; Sect. 2 presents a background for evolutionary search and PSO algorithm, Sect. 3 describes the proposed approach, while Sect. 4 presents the experiments and results. Finally, the conclusion in Sect. 5.

2 Preliminaries

2.1 Evolutionary Search

In this method, a set of individuals are initialized, which each of them represents a possible solution of the formulated problem [6]. Then, two individuals are selected as promising parents to generate two off-springs, and an iterative process of selecting parents is used to generate a group of off-springs using crossover and mutation techniques. Thereafter, the refined two children are replaced with non-relevant solutions in the search space called steady-state strategy or the group of children is replaced with the whole solutions called generational strategy. However, the strategy is repeated to hopefully end up with an optimal (or sub-optimal) solution for the specific problem.

Evolutionary strategy is considered one of the most popular evolutionary algorithms that uses generational strategy for updating the population either from the generated off-springs (μ, λ) or from the combination of the best individuals of parents and the generated off-springs $(\mu + \lambda)$; where the μ refers to parents and λ refers to off-springs with $\mu > \lambda$. In this chapter, the (μ, λ) evolution strategy is used that is shown in Algorithm 1. The algorithm in a loop generates λ individuals from ran-

dom parent P_i and then, replaces the first μ individuals in the population with the λ individuals.

For selection scheme, the binary tournament selection is used. Then a bit-flip mutation is applied and generational replacement with elitism is used to keep the best individual. These processes are applied in each iteration until the maximum number of iterations is reached [6].[1]

Algorithm 1 (μ, λ) Evolutionary Search

1: **Initialization**
2: Choose $x_1, x_2, \ldots, x_\mu \in \{0, 1\}^n$ uniformly at random
3: Collect $x_1, x_2, ..., x_\mu$ in P_0
4: $t := 0$
5: **while** $\neg StopCondition$ **do**
6: **for all** $i \in \{1, 2, \ldots, \lambda\}$ **do**
7: Select $y \in P_t$ uniformly at random
8: Create y_i by standard bit mutation of y with $p_m = 17/n$
9: **Selection for Replacement**
10: Collect the first μ individuals in Pt+1.
11: $t := t + 1$
12: **return** S_{best}

2.2 Particle Swarm Optimization

One of the main swarm-based methods is the PSO algorithm created by Kennedy and Eberhart for handling both constrained and unconstrained tasks [3, 7]. The structure of this method is simple. It uses a set of moving agents named as a particle that can change their locations based on some stochastic and iterative processes. The best particle (fittest one) among all particles is known as gbest. All particles try to reach the location of gbest. All particles also consider the best position explored by them during the searching process. Personal best location (pbest) is the best solution found by a specific search agent. If this pbest be the best location found by all particles, then, we have the same gbest and pbest. Based on this logic, all particles need to be updated during several iterations limited by the user. The main rules are obtained via:

$$v_i^j(t+1) = \omega_1 v_i^j(t) + c_1 r_1 (pbest_i^j - x_i^j(t)) + c_2 r_2 (gbest_i^j - x_i^j(t)) \quad (1)$$

$$x_i^j(t+1) = x_i^j(t) + v_i^j(t+1) \quad (2)$$

[1] Interested readers can refer to following links accessed on April 2019 (a) https://github.com/sebastian-luna-valero/EvolutionarySearch (b) http://weka.sourceforge.net/packageMetaData/EvolutionarySearch/index.html (c) NEO Group website at http://neo.lcc.uma.es.

where t represents the current iteration, ω_1 denotes an inertial weight, $v_i^j(t)$ is the velocity value of j-th dimension in i-th solution, $x_i^j(t)$ denotes j-th dimension in i-th solutions, r_1 and r_2 show random values in (0,1), c_1 and c_2 show the individual and social coefficients, respectively. These factors usually are set to 2. The $pbest$ and $gbest$ show the personal and global best solutions, respectively. As stated by Angeline [1], although the PSO is able to solve some multi-modal problems faster than other optimizers like GA, it may still be trapped in local optima when tackling cases with many variables.

In this chapter, we utilized the geometric PSO [10], which is a modified variant of PSO proposed in 2007. In this variant, all parts are similar, but the velocity term is removed, and the rule for the updating of position vector is the convex combination. In addition, a mutation scheme is utilized and three weighting coefficients are non-negative and their summation is one. For this PSO variant, we have some specific convex combination rules for the Euclidean, Manhattan, and Hamming spaces, which are explained in [10].

The pseudo-code of the geometric PSO is observed in Algorithm 2.

Algorithm 2 Pseudo-code of geometric PSO algorithm

```
Initialize the population x_i (i = 1, 2, ..., n)
while stopping condition is not met do
    for Each particle i do
        Obtain the fitness of all particles
        Update the positions of particles based on a randomized convex combination
        Mutate x_i
        Attain the fitness values f(x_i^j)
        if f(x_i^j) < f(pbest_i^j) then
            pbest_i^j ← x_i^j
        if f(x_i^j) < f(gbest_i^j) then
            gbest_i^j ← x_i^j
```

3 Framework

The mathematical formulation of feature selection is a combinatorial optimization problem for optimizing the generalization performance of the predictive model. This model depends on the error of selecting a subset of features from the dataset. In particular, a random population of individuals that is a set of possible predictive models (or solutions) is generated. Each individual consists of a sequence of genes represent the inclusion or not of particular features in the model. Thereafter, a predictive model or classifier is trained for each individual by only using inclusion genes (features). In order to evaluate the quality of the predictive model in selection of the features, the evolution technique uses the fitness function. It represents the evaluation metric used in the machine learning field. In a consecutive sequence of generations,

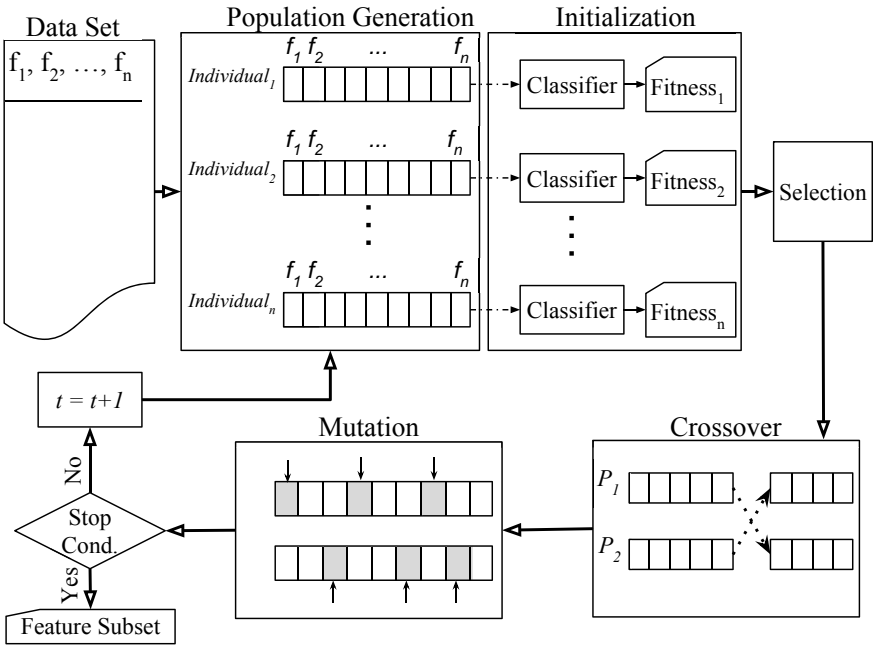


Fig. 1 Feature selection framework based on a metaheuristic algorithm

the individuals of high fitness values have a high probability of being selected in recombination and mutation to reach the optimal individuals of the relevant subset of features. Figure 1 illustrates this framework.

For instance, assume a set of features in a data set are $F = \{f_1, f_2, f_3, \ldots, f_n\}$, the feature selection technique aims to select a subset of such features $F_s \subseteq F$ that are more relevant and provide high generalization performance for the predictive model. The technique generates random individuals each of which contains a sequence of genes either 1 for inclusion or 0 for exclusion of features for building the predictive model. Let us use the decision tree (DT) algorithm as a classifier and the AUC as metric of fitness function evaluation. We build N predictive models (or DT classifiers) represent the population size with each individual relate to a specific label and evaluate the AUC metric as a fitness function. Finally, while using evolution processes, the feature selection technique picks the most relevant individual of high fitness value, where the inclusion features have resulted as a subset of near-optimal output for the performance generalization of predictive models.

3.1 Metaheuristic Search

In this framework, two metaheuristic algorithms are used as a search algorithm in the wrapper, which are the evolutionary search and particle swarm optimization. We described these methods earlier in Sect. 2. These two algorithms were selected due to their popularity in the literature and their wide range of applications in feature selection tasks.

3.2 Internal Evaluation (Fitness Functions)

Three different fitness functions are utilized and experimented independently to evaluate their effect on the classification performance in the imbalanced class distribution. The three functions are the accuracy rate, g-mean, and area under ROC (AUC). The accuracy rate and g-mean measures are derived from the confusion matrix for binary classification problems shown in Table 1.

- Accuracy: This metric reflects the total number of correctly classified instances compared to the total number of instances in the dataset and it can be represented as:

$$Accuracy\,rate = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

- G-mean: This metric reveals the balance between classification performances on the majority and minority class considering both Sensitivity and specificity.

$$G\text{-}Mean = \sqrt{\frac{TP}{P} \times \frac{TN}{N}} \tag{4}$$

- Area under ROC (AUC): This metric is a well-regarded metric that can be utilized to evaluate the tested classifiers with aim of providing a probabilistic score for their prediction rates. AUC can be obtained using a prior info that presented in graph of receiver operating characteristics (ROC). AUC is suitable to deal with imbalanced datasets because it does not dependent on the ratio of the classes [5].

Table 1 Confusion matrix for binary classification

	Predicted positive	Predicted negative
Actual positive	True positive (TP)	False negative (FN)
Actual negative	False positive (FP)	True negative (TN)

$$ROC = \frac{1}{N_{Positive} \times N_{Negative}} \sum_{i_{Positive}}^{N_{Positive}} \sum_{i_{Negative}}^{N_{Negative}} 1_{P_{i_{Positive}} > P_{i_{Negative}}} \quad (5)$$

3.3 Induction Algorithm (The Classifier)

The learning algorithm which is in our case will be a classifier that is frequently built during the search process of the wrapper. For each candidate subset of features, a new classifier is built using these features. Wrapper-based FS is well-known to be computationally expensive due to this learning process. Therefore, it is common to choose a fast and easy method to train algorithm. In this work, we chose the decision trees J48 algorithm for this task.[2] J48 is an implementation in Java of the popular C4.5 algorithm[3] [11].

3.4 External Evaluation

For evaluating the final prediction models that are developed based on the selected feature, six evaluation measures are used. Beside the accuracy rate, AUC, and g-mean, three other ratios are used which are:

- Sensitivity is the number of positive data instances that are correctly classified and divided by the number of the positive data instances (P), which is also called true positive rate. Sensitivity can be represented as given in Eq. 6.

$$Sensitivity = \frac{TP}{P} \quad (6)$$

- Specificity is the number of negative data instances that are correctly classified and divided by the number of the negative data instances (P), which is also called true negative rate. Specificity can be represented as given in Eq. 7.

$$Specificity = \frac{TN}{N} \quad (7)$$

- The average number of selected features. This number is calculated over the total number of repetitions of the experiments.

[2] Interested readers can refer to following link: https://machinelearningmastery.com/how-to-run-your-first-classifier-in-weka/.

[3] Interested readers can refer to following link: http://weka.sourceforge.net/doc.dev/weka/classifiers/trees/J48.html.

3.5 Relevance of Input Features

To quantifying the relevance of input features, we count the appearance of each feature in the best subset of features over the course of iterations. The more the feature is selected, the higher relevance is.

4 Experiments and Results

In this section, we conduct a set of experiments to evaluate the implementation of the EA and PSO as feature selection techniques based on a set of imbalanced datasets using various fitness functions derived from the conventional evaluation metrics.

4.1 Setup of Experiments

The framework of the experiments was implemented using Android studio and the open source Weka 3.9 library in order to utilize and upgrade the built-in functionality in Weka [6].[4] Two packages in Weka were used; which are evolutionary search and PSO search. The two packages were modified to integrate the G-mean based fitness function and the feature impact analyzer.

The initial values of the EA and PSO parameters are given in Tables 2 and 3.

4.2 Description of Datasets

In this study, six imbalanced datasets are drawn from the UCI repository [2]. The details of datasets are presented in Table 4. The ratio of the minor class in the datasets is varied from to around 8–35%. It is expected that the smaller ratio of the minor class the more challenging the dataset becomes.

4.3 Results and Discussion

The results of the experiments are shown in Tables 5, 6, 7, 8, 9 and 10. Each table presents the evaluation results of EA and PSO of different fitness functions when evaluated based on a specific dataset.

[4] Interested readers can refer to following links: (a) https://developer.android.com/studio (b) https://www.cs.waikato.ac.nz/ml/weka/documentation.html.

Table 2 Setting parameters for EA

Parameters	Value
Generations	20
Population size	20
Crossover probability	0.6
Mutation probability	0.1

Table 3 Setting parameters for PSO

Parameters	Value
Iterations	20
Population size	20
Individual weight	0.34
Inertia weight	0.33
Mutation probability	0.01

Table 4 Descriptions of datasets [2]

Dataset	No. of instances	No.of attributes	Class attribute	(Minority %: Majority %)
Pima	768	8	(1, 0)	(34.77: 65.23)
German	1000	20	(Bad, good)	(30.00: 70.00)
Ecoli	336	7	(iMU, remainder)	(10.42: 89.58)
Haberman	306	3	(Die, survive)	(26.47: 73.53)
Splice	3176	60	(ei, remainder)	(23.99: 76.01)
Glass	214	9	(Ve-win-float-proc, remainder)	(7.94: 92.06)

Table 5 Evaluation results based on ecoli dataset

Fitness function	Accuracy		AUC		G-mean	
	EA	PSO	EA	PSO	EA	PSO
Accuracy	0.9349 (0.0028)	0.9344 (0.0035)	0.9314 (0.0020)	0.9314 (0.0020)	0.9289 (0.0042)	0.9288 (0.0045)
Specificity	0.7268 (0.0151)	0.7273 (0.0149)	0.7353 (0.0038)	0.7353 (0.0038)	0.7507 (0.0154)	0.7507 (0.0164)
Sensitivity	0.9349 (0.0028)	0.9344 (0.0035)	0.9314 (0.0020)	0.9314 (0.0020)	0.9289 (0.0042)	0.9288 (0.0045)
AUC	0.8244 (0.0537)	0.8258 (0.0542)	0.8848 (0.0052)	0.8848 (0.0052)	0.8064 (0.0392)	0.8051 (0.0482)
G-mean	0.6769 (0.0243)	0.6781 (0.0239)	0.6923 (0.0052)	0.6923 (0.0052)	0.7158 (0.0223)	0.7157 (0.0238)
Features	2.43 (0.63)	2.47 (0.63)	2.13 (0.43)	2.13 (0.43)	4.77 (0.86)	4.43 (1.17)

Table 6 Evaluation results based on German dataset

Fitness function	Accuracy		AUC		G-mean	
	EA	PSO	EA	PSO	EA	PSO
Accuracy	0.7456 (0.0095)	0.7482 (0.0094)	0.7366 (0.0084)	0.7379 (0.0123)	0.7423 (0.009)	0.7378 (0.0076)
Specificity	0.6591 (0.0122)	0.6628 (0.0123)	0.653 (0.0176)	0.6435 (0.0181)	0.6748 (0.0151)	0.6625 (0.012)
Sensitivity	0.7456 (0.0095)	0.7482 (0.0094)	0.7366 (0.0084)	0.7379 (0.0123)	0.7423 (0.009)	0.7378 (0.0076)
AUC	0.712 (0.0134)	0.7236 (0.0095)	0.7236 (0.0095)	0.7272 (0.0093)	0.7091 (0.0145)	0.714 (0.0156)
G-mean	0.6273 (0.0163)	0.6172 (0.0306)	0.6172 (0.0306)	0.597 (0.0319)	0.6529 (0.0203)	0.6349 (0.0167)
Features	7.23 (2.05)	5.63 (0.96)	5.63 (0.96)	5.06 (0.87)	8.27 (1.55)	6.53 (1.79)

Table 7 Evaluation results based on glass dataset

Fitness function	Accuracy		AUC		G-mean	
	EA	PSO	EA	PSO	EA	PSO
Accuracy	0.9263 (0.0036)	0.9251 (0.0043)	0.9274 (0.0053)	0.9263 (0.0068)	0.9293 (0.002)	0.9291 (0.0022)
Specificity	0.6617 (0.0123)	0.6243 (0.0707)	0.6569 (0.0193)	0.6545 (0.0212)	0.6642 (0.0107)	0.6623 (0.0146)
Sensitivity	0.9263 (0.0036)	0.9251 (0.0043)	0.9274 (0.0053)	0.9263 (0.0068)	0.9293 (0.002)	0.9291 (0.0022)
AUC	0.7784 (0.0345)	0.6989 (0.1526)	0.8154 (0.0207)	0.8119 (0.0235)	0.8151 (0.0192)	0.8134 (0.0209)
G-mean	0.5817 (0.0217)	0.4465 (0.2513)	0.5717 (0.0361)	0.5678 (0.0395)	0.5843 (0.0197)	0.5808 (0.0272)
Features	3.6 (0.86)	2.77 (1.65)	5.2 (0.71)	5.3 (0.84)	4.57 (0.82)	4.57 (0.82)

Table 8 Evaluation results based on Haberman dataset

Fitness function	Accuracy		AUC		G-mean	
	EA	PSO	EA	PSO	EA	PSO
Accuracy	0.7353 (0)	0.7353 (0)	0.719 (0)	0.719 (0)	0.719 (0)	0.719 (0)
Specificity	0.5 (0)	0.5 (0)	0.5837 (0)	0.5837 (0)	0.5837 (0)	0.5837 (0.0146)
Sensitivity	0.7353 (0)	0.7353 (0)	0.719 (0)	0.719 (0)	0.719 (0)	0.719 (0)
AUC	0.4889 (0)	0.4889 (0)	0.6087 (0)	0.6087 (0)	0.6087 (0)	0.6087 (0)
G-mean	0 (0)	0 (0)	0.508 (0)	0.508 (0)	0.508 (0)	0.508 (0)
Features	0 (0)	0 (0)	2 (0)	2 (0)	2 (0)	2 (0)

Table 9 Evaluation results based on pima dataset

Fitness function	Accuracy		AUC		G-mean	
	EA	PSO	EA	PSO	EA	PSO
Accuracy	0.7568 (0.0038)	0.7557 (0.0053)	0.7508 (0.0061)	0.7508 (0.0061)	0.7502 (0.0078)	0.7494 (0.0078)
Specificity	0.7183 (0.0051)	0.7173 (0.0067)	0.7068 (0.0058)	0.7068 (0.0058)	0.712 (0.0074)	0.711 (0.0084)
Sensitivity	0.7568 (0.0038)	0.7557 (0.0053)	0.7508 (0.0061)	0.7508 (0.0061)	0.7502 (0.0078)	0.7494 (0.0078)
AUC	0.7614 (0.0138)	0.7587 (0.0156)	0.784 (0.0108)	0.784 (0.0108)	0.7676 (0.0138)	0.7658 (0.0141)
G-mean	0.7067 (0.0071)	0.7058 (0.009)	0.6916 (0.0059)	0.6916 (0.0059)	0.7006 (0.0082)	0.6994 (0.0098)
Features	4.57 (0.63)	4.7 (0.88)	4.47 (0.63)	4.47 (0.63)	5.63 (0.99)	5.8 (0.76)

Table 10 Evaluation results based on splice dataset

Fitness function	Accuracy		AUC		G-mean	
	EA	PSO	EA	PSO	EA	PSO
Accuracy	0.9738 (0.0005)	0.9746 (0.0004)	0.9726 (0.0014)	0.9726 (0.001)	0.9737 (0.0007)	0.9743 (0.0006)
Specificity	0.9718 (0.0009)	0.9732 (0.0008)	0.9702 (0.0021)	0.9704 (0.0015)	0.9719 (0.001)	0.9727 (0.001)
Sensitivity	0.9738 (0.0005)	0.9746 (0.0004)	0.9726 (0.0014)	0.9726 (0.001)	0.9737 (0.0007)	0.9743 (0.0006)
AUC	0.9734 (0.0015)	0.9754 (0.0014)	0.9733 (0.0023)	0.9754 (0.0016)	0.9734 (0.0018)	0.9745 (0.0017)
G-mean	0.9718 (0.0009)	0.9732 (0.0008)	0.9702 (0.0021)	0.9704 (0.0015)	0.9719 (0.001)	0.9727 (0.001)
Features	30.13 (2.71)	15.03 (2.51)	28.97 (3.21)	20.1 (3.84)	30.03 (4.25)	15.97 (2.24)

As per results for the first dataset, if we consider the accuracy as fitness function, we see that the EA perform better than PSO in terms of accuracy. However, in terms of AUC and g-mean, PSO is better than EA. Based on the number of features, again EA is better than PSO.

When we use AUC as fitness function, we see the PSO and EA have exactly the same performance based on all measures. If we consider g-mean as fitness function, we see almost similar accuracy, specificity, and sensitivity measures. However, EA outperforms its peer based on AUC metric and number of features. In terms of g-mean, they are almost similar.

As per result on German dataset, it is observed that the PSO provides better accuracy when we evaluate the results based on accuracy fitness function. This also remains valid in terms of AUC and number of features. It is interesting that we see that EA performs better in terms of g-mean. If we consider the AUC as the evaluation metric, we see PSO is preferable in terms of accuracy, number of features, and AUC. However, this time, EA is better in terms of g-mean. In the case of g-mean as fitness function, EA has obtained a higher accuracy and g-mean. Again, we see the PSO is better than EA based on AUC metric and number of features.

Based on results for glass dataset and we consider the accuracy as the fitness function, we see that EA outperforms the PSO in terms of the accuracy, AUC, and g-mean rates. However, in terms of number of features, we see that the PSO is better than GA. In the case of AUC metric, PSO is better than EA in terms of accuracy metric. However, EA outperforms the PSO based on AUC and g-mean metrics. In addition, EA provides a lower number of features. In the last case that we have set the g-mean as the evaluation function, EA is better than PSO in terms of all metrics.

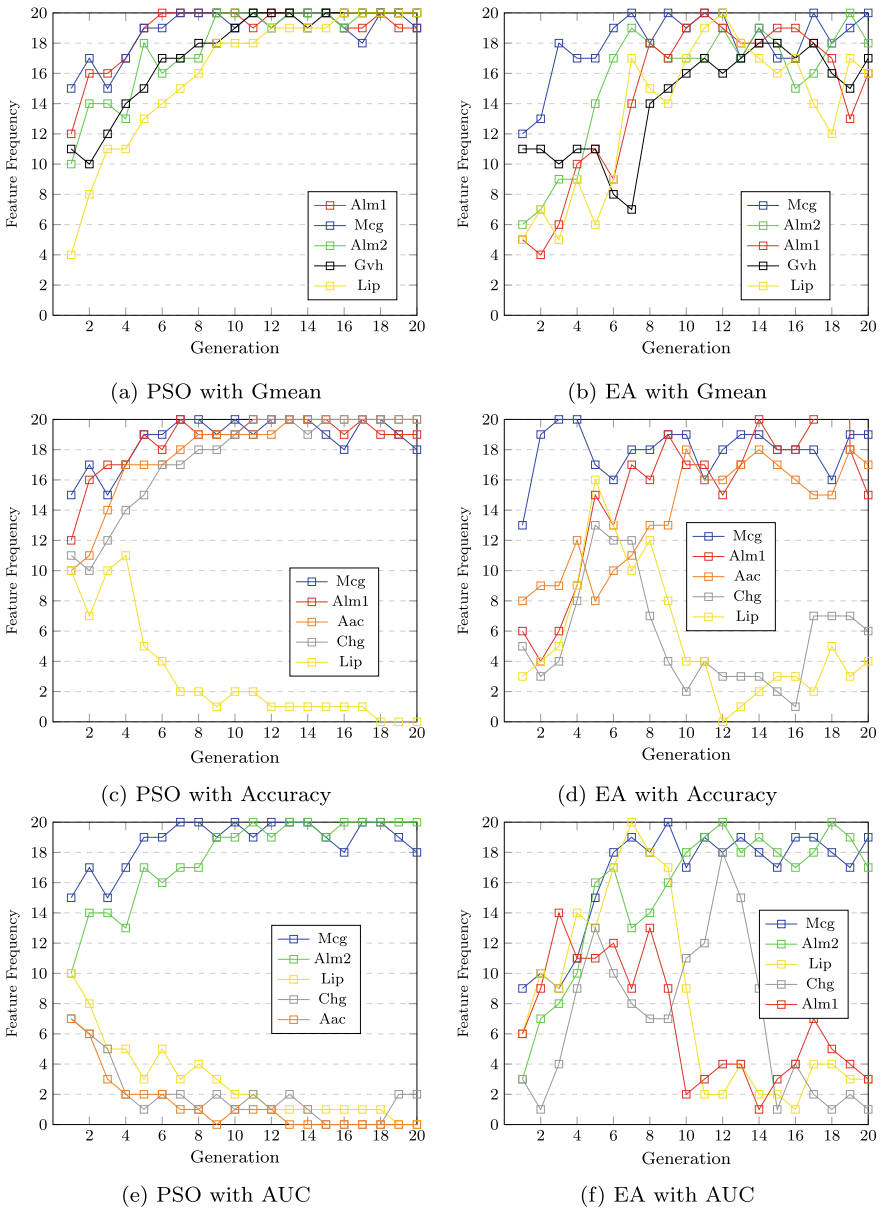
Based on Haberman case, we see there is no difference between PSO and EA when we consider any fitness function.

In the case of splice, we see PSO beats EA in all measures, when we consider the accuracy as the fitness function. In the case of AUC, the results are very competitive but PSO has a better value in terms of AUC and number of features. If we consider g-mean as the fitness function, it is observed that the PSO beats EA in all measures.

In the case of pima dataset, by using the accuracy function, EA outperforms PSO in all cases except number of features. It is interesting that based on second fitness, all results are same and no difference was observed. Based on the g-mean function, we see EA is slightly better than PSO in all measures.

From another point of view, the obtained results show that the accuracy on each class is maximized in terms of sensitivity and specificity when the g-mean fitness function is used. We can notice also that AUC comes second, while the accuracy fitness is worst for these imbalanced cases. However, this advantage of the g-mean fitness function comes at the cost of increase in the number of selected features compared to the AUC and accuracy functions.

In Figs. 2, 3, 4, 5, 6 and 7, we monitor the frequency of features over the course of generations for both methods in the six datasets. For clarity purpose, we plotted the most frequent five features obtained by both methods in dealing with these dataset. Examining these figures, it can be noticed that in most cases the most relevant five features and their rank change by changing the fitness function used in the metaheuristic. Therefore, any analysis of the relevant features performed after a bad selection of a fitness function can mislead the conclusions, and consequently affect the any decision process can be built based on this analysis.



(a) PSO with Gmean

(b) EA with Gmean

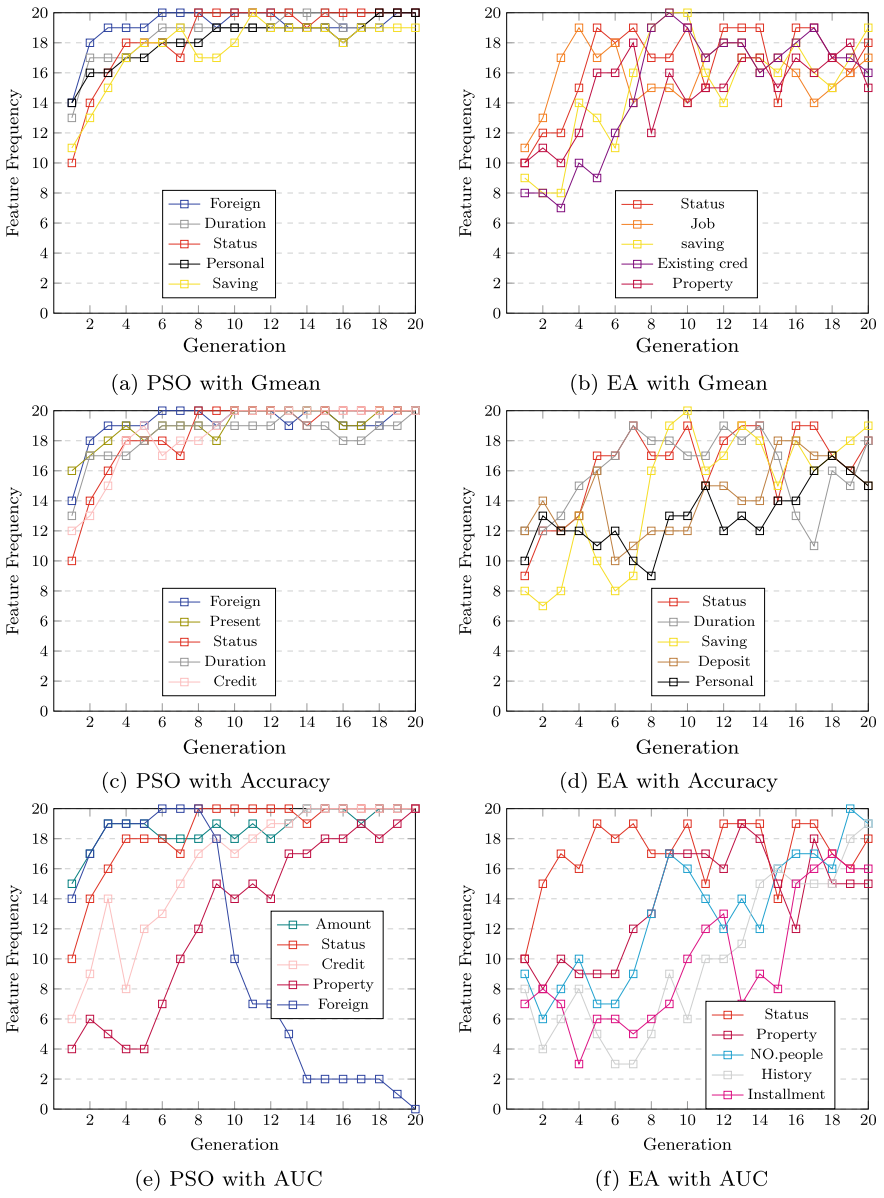
(c) PSO with Accuracy

(d) EA with Accuracy

(e) PSO with AUC

(f) EA with AUC

Fig. 2 Identification of the most frequent five features in Ecoli dataset

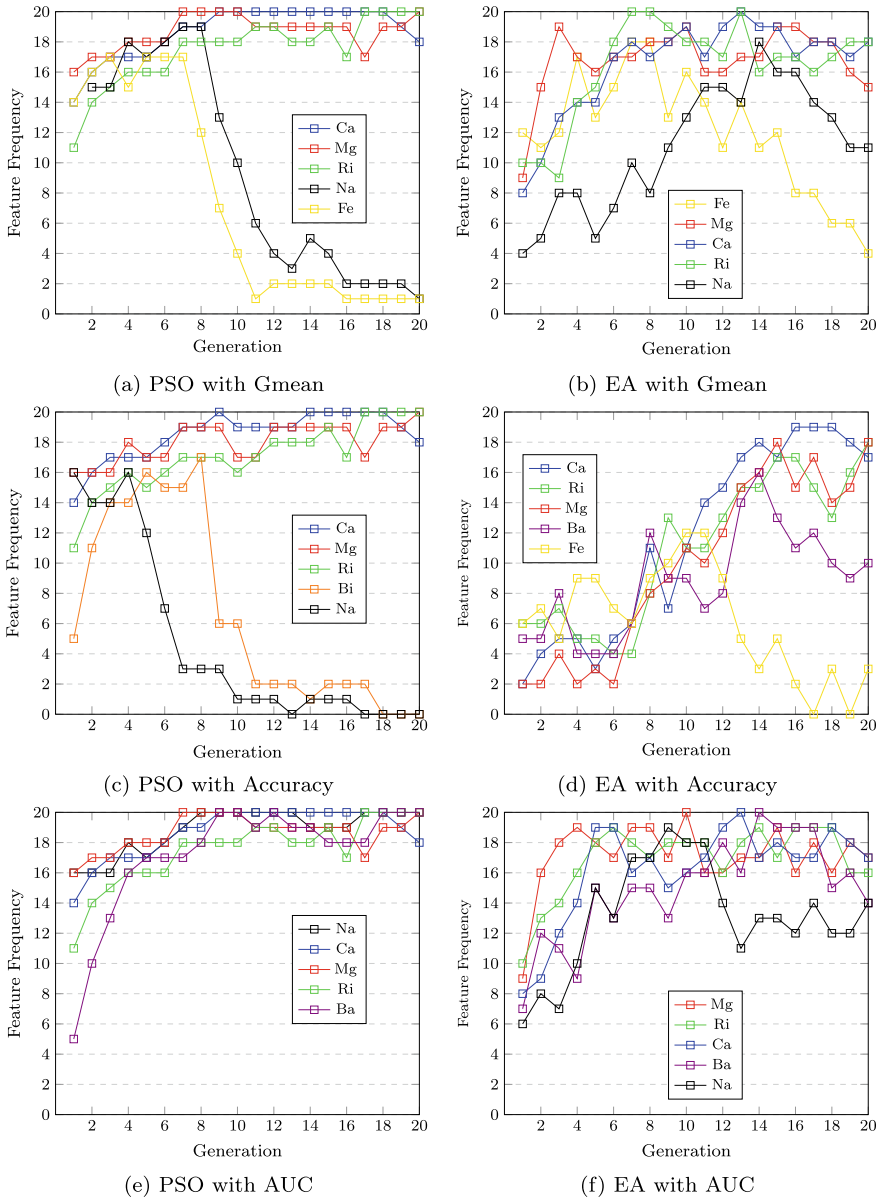


(a) PSO with Gmean (b) EA with Gmean

(c) PSO with Accuracy (d) EA with Accuracy

(e) PSO with AUC (f) EA with AUC

Fig. 3 Identification of the most frequent five features in German dataset



(a) PSO with Gmean

(b) EA with Gmean

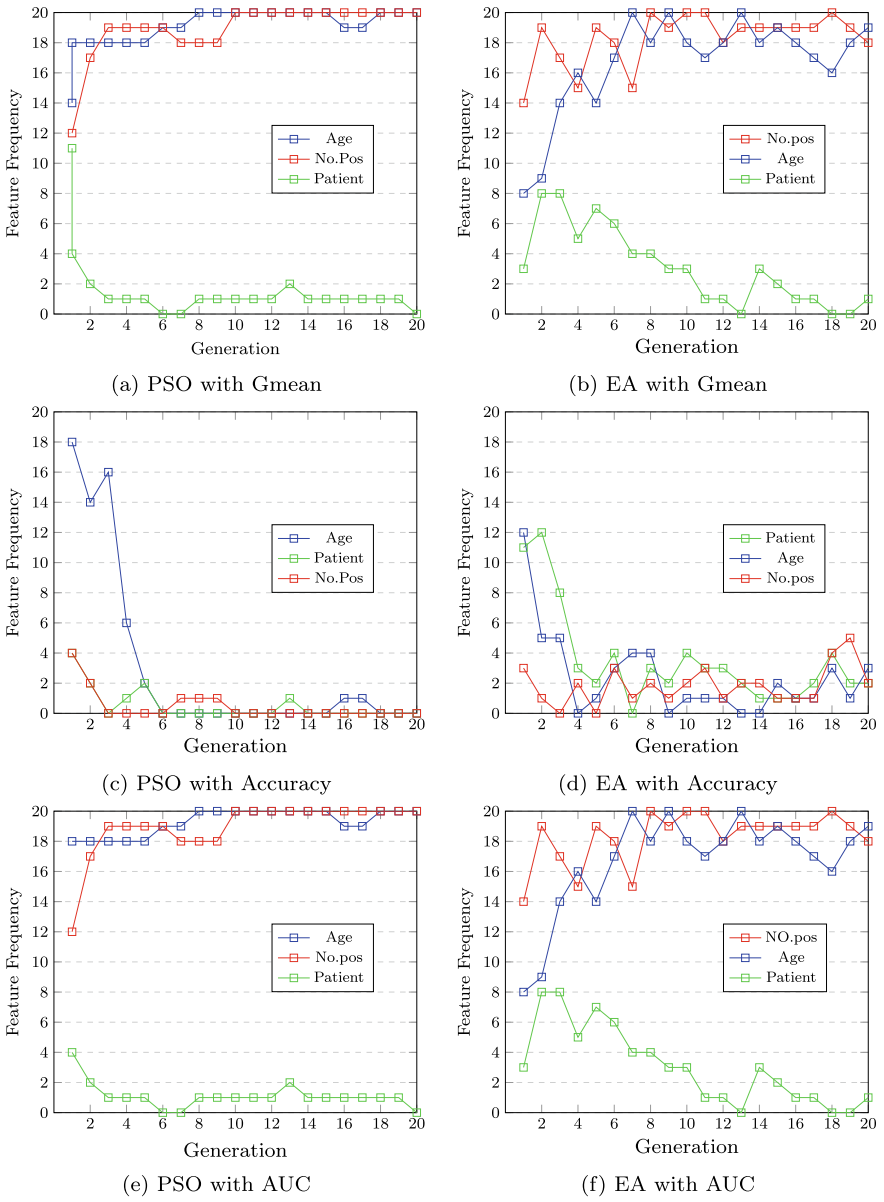
(c) PSO with Accuracy

(d) EA with Accuracy

(e) PSO with AUC

(f) EA with AUC

Fig. 4 Identification of the most frequent five features in glass dataset



(a) PSO with Gmean

(b) EA with Gmean

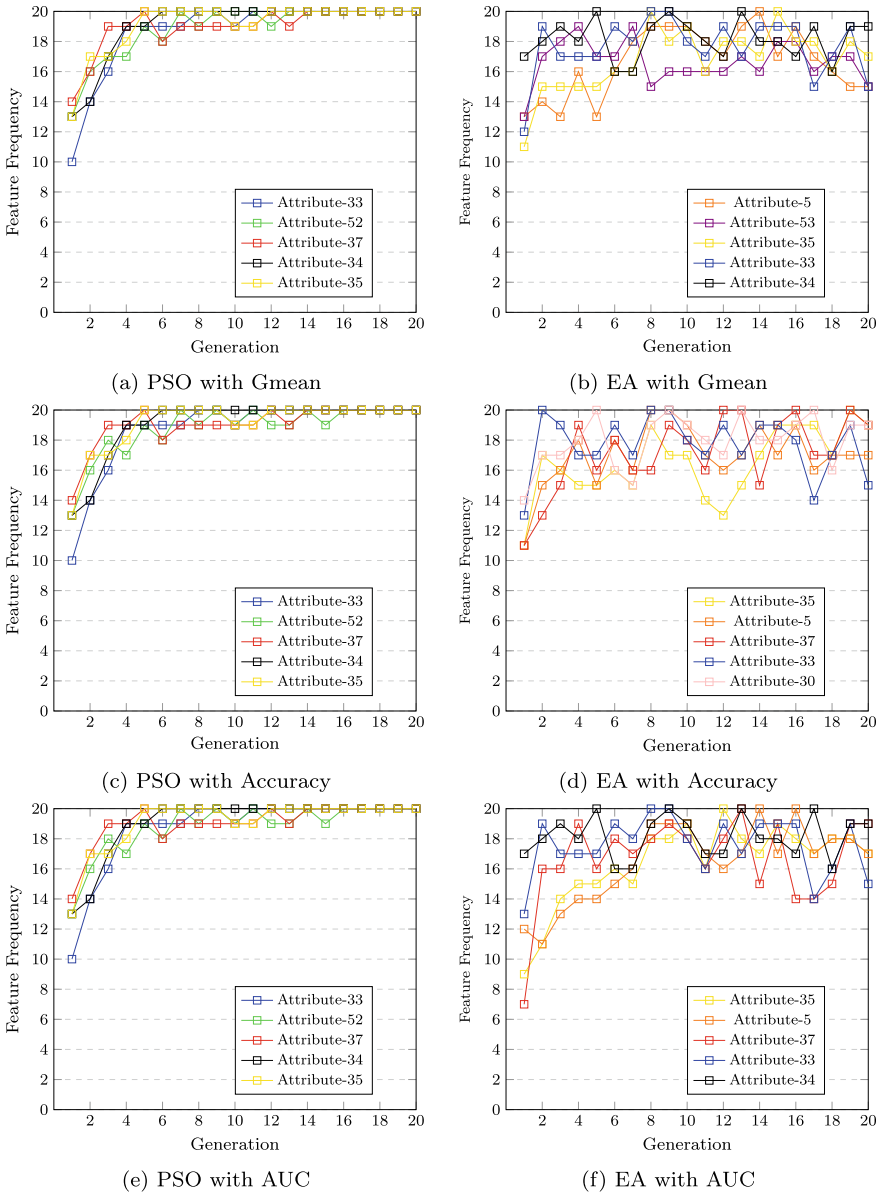
(c) PSO with Accuracy

(d) EA with Accuracy

(e) PSO with AUC

(f) EA with AUC

Fig. 5 Identification of the most frequent five features in Haberman dataset



(a) PSO with Gmean (b) EA with Gmean

(c) PSO with Accuracy (d) EA with Accuracy

(e) PSO with AUC (f) EA with AUC

Fig. 6 Identification of the most frequent five features in splice dataset

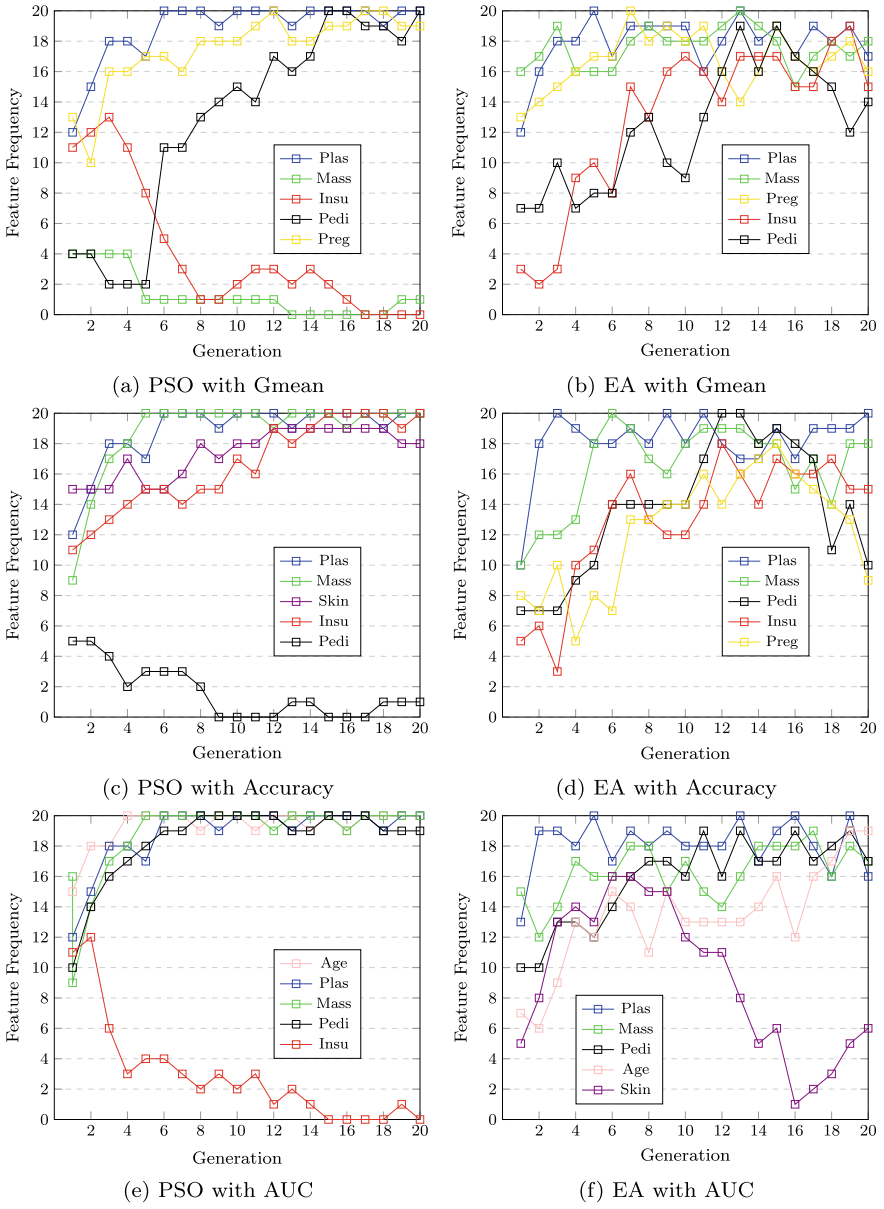


Fig. 7 Identification of the most frequent five features in pima dataset

5 Conclusion

In this chapter, we investigated the impacts of three different fitness functions on the performance of evolutionary and swarm-based feature selection algorithms in the context of imbalanced datasets. The fitness functions are the accuracy rate, the area under ROC curve (AUC), and the g-mean. The experiments were conducted on six datasets that are varied in their minor class ratio which ranged from 8 to 35%. Evaluation results show that by using g-mean and AUC fitness functions a better balance between the classification accuracy of each class can be achieved. Moreover, there is a slight advantage for g-mean that comes at the cost of a small increase in the number of selected features. Therefore, the g-mean fitness function is highly recommended when the targeted dataset has a high imbalanced class distribution. On the other side, the accuracy fitness function should be avoided as it tends to increase the classification accuracy of the major class at the expense of the minor class, and consequently can mislead the identification process of the relevant features.

References

1. Angeline PJ (1998) Using selection to improve particle swarm optimization. In: The 1998 IEEE International Conference on Evolutionary computation proceedings, 1998. IEEE World Congress on Computational Intelligence. IEEE, pp 84–89
2. Dua D, Graff C (2017) UCI machine learning repository
3. Eberhart R, Kennedy J (1995) A new optimizer using particle swarm theory. In: Proceedings of the sixth international symposium on Micro machine and human science, 1995. MHS'95. IEEE, pp 39–43
4. Faris H, Al-Zoubi Heidari AA, Aljarah I, Mafarja M, Hassonah MA, Fujita H (2019) An intelligent system for spam detection and identification of the most relevant features based on evolutionary random weight networks. Inf Fusion 48:67–83
5. Faris H, Aljarah I, Al-Madi N, Mirjalili S (2016) Optimizing the learning process of feedforward neural networks using lightning search algorithm. Int J Artif Intell Tools 25(06):1650033
6. Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH (2009) The WEKA data mining software: an update. SIGKDD Explor 11(1):10–18
7. Kennedy J, Eberhart R (1995) A new optimizer using particle swarm theory. In: Proceedings of the sixth international symposium on micro machine and human science, 1995. MHS '95, pp 39–43
8. Lin S-W, Ying K-C, Chen S-C, Lee Z-J (2008) Particle swarm optimization for parameter determination and feature selection of support vector machines. Expert Syst Appl 35(4):1817–1824
9. Mavrovouniotis M, Li C, Yang S (2017) A survey of swarm intelligence for dynamic optimization: algorithms and applications. Swarm Evol Comput 33:1–17
10. Moraglio , Di Chio C, Poli R (2007) Geometric particle swarm optimisation. In: European conference on genetic programming. Springer, pp 125–136
11. Quinlan R (1993) C4.5: programs for machine learning. Morgan Kaufmann Publishers, San Mateo, CA
12. Talbi E-G (2009) Metaheuristics: from design to implementation, vol 74. Wiley
13. Whitley D (1994) A genetic algorithm tutorial. Stat Comput 4(2):65–85
14. Wolpert DH, Macready WG et al (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82

Binary Harris Hawks Optimizer for High-Dimensional, Low Sample Size Feature Selection

Thaer Thaher, Ali Asghar Heidari, Majdi Mafarja, Jin Song Dong and Seyedali Mirjalili

Abstract Feature selection is a preprocessing step that aims to eliminate the features that may negatively influence the performance of the machine learning techniques. The negative influence is due to the possibility of having many irrelevant and/or redundant features. In this chapter, a binary variant of recent Harris hawks optimizer (HHO) is proposed to boost the efficacy of wrapper-based feature selection techniques. HHO is a new fast and efficient swarm-based optimizer with various simple but effective exploratory and exploitative mechanisms (Levy flight, greedy selection,

T. Thaher
College of Engineering and Information Technology, An-Najah National University, Nablus, Palestine
e-mail: thaer.thaher@gmail.com

Department of Information Technology, At-Tadamun Society, Nablus, Palestine

A. A. Heidari
School of Surveying and Geospatial Engineering, College of Engineering, University of Tehran, Tehran, Iran

A. A. Heidari · J. S. Dong
Department of Computer Science, School of Computing, National University of Singapore, Singapore, Singapore
e-mail: as_heidari@ut.ac.ir; aliasgha@comp.nus.edu.sg; t0917038@u.nus.edu

J. S. Dong
e-mail: dongjs@comp.nus.edu.sg; j.dong@griffith.edu.au

M. Mafarja
Department of Computer Science, Faculty of Engineering and Technology, Birzeit University, PoBox 14, Birzeit, Palestine
e-mail: mmafarja@birzeit.edu

J. S. Dong
Institute for Integrated and Intelligent Systems, Griffith University, Nathan, Brisbane 4111, Australia

S. Mirjalili (✉)
Torrens University Australia, Brisbane, QLD 4006, Australia
e-mail: ali.mirjalili@gmail.com

Griffith University, Brisbane, QLD 4111, Australia

S. Mirjalili et al. (eds.), *Evolutionary Machine Learning Techniques*,
Algorithms for Intelligent Systems, https://doi.org/10.1007/978-981-32-9990-0_12

etc.) and a dynamic structure for solving continuous problems. However, it was originally designed for continuous search spaces. To deal with binary feature spaces, we propose a new binary HHO in this chapter. The binary HHO is validated based on special types of feature selection datasets. These hard datasets are high dimensional, which means that there is a huge number of features. Simultaneously, we should deal with a low number of samples. Various experiments and comparisons reveal the improved stability of HHO in dealing with this type of datasets.

Keywords Harris Hawk optimizer · Optimization · Feature selection · Neural networks · Artificial intelligence · Machine learning · Data science

1 Introduction

With[1] the development of technologies, the dimension of the collected data is increased which has the largest effect on extracting the useful information to analyze the data [1]. However, some features of the collected data are irrelevant, and the redundant features lead to the degradation of the performance of the prediction methods. Also, it is a challenge to determine whether the features are relevant or not. To address the problem of the high dimensionality of data (also known as "the curse of dimensionality") and improving the performance of the prediction, the feature selection (FS) can be used as a preprocessing step. The FS selects only the relevant features and removing the other features (i.e., irrelevant and redundant) and decreases the time required to learn the predictors.

There are many factors that make the FS as a difficult task among these factors, like the high interaction among the features [2]. So, it is important to determine the subset of features that can capture the characteristics of the classes for distinguishing among them with a small number of features.

Whereas the large search domain makes the FS as a more challenging problem, because the size of the search domain is increased exponentially with increasing the number of features. For example, when the dimension of the dataset is l features, there exists 2^l solutions and it is impractical to use an exhaustive search for solving the FS. So, the FS methods require two components: (1) the evaluation criterion, which is used to assess the quality of the selected features, and (2) the search technique, which explores the domain of the feasible solutions to determine the optimal subset of features.

According to the evaluation criterion, there are two groups of the FS methods: the filter-based and the wrapper-based [3]. The filter-based approaches select the features subset independently from the predictors. This FS category includes information gain (IG), ReliefF, rain ratio, and chi-square [4]. Unlike the filter-based approaches, the

[1]Note that the codes of HHO method can be publicly downloaded from: http://www.alimirjalili.com/HHO.html and http://www.evo-ml.com/2019/03/02/hho.

category of the wrapper-based approaches used the predictors to evaluate the quality of the selected features.

Moreover, there are several search techniques that have been used to find the subset of features to improve the performance of classification such as the greedy search and the random search [5]. In the greedy search technique, all combinations of the features are generated and evaluated which make this technique be time-consuming. Meanwhile, the random search technique explores the search space randomly for the next subset of features. However, these methods have some limitations such as easily stuck at a local optimal point and, also, they have high space and time complexity. In order to solve these problems of the previous mentioned FS methods, the metaheuristic techniques have been used.

These well-spread methods emulate the physical, the biological, and the animal social behaviors in nature [6]. However, despite various mathematical models and inspirations, all metaheuristic approaches have two core phases: diversification (exploration) and intensification (exploitation) [7]. Harris hawks optimizer (HHO) is a new powerful swarm-based method proposed by Heidari et al. [8]. It utilizes two phases of exploration and four phases of exploitation, which makes it possible to get excellent results in dealing with several mathematical and engineering problems.

In this work, a wrapper-based FS method based on KNN classifier and the recently proposed HHO algorithm is designed for the first time to handle the high-dimensional, low sample medical datasets. Generally, the FS methods have two main objectives: minimizing the number of the selected features and minimizing the error rate of the classification. We considered both objective functions and after a sensitivity analysis and comparing S-shaped and V-shaped transfer functions (TFs), we compared the results of the best binary variant with other well-established methods.

2 Feature Selection

FS is the process of choosing a subset of features according to a certain criterion in order to eliminate the irrelevant and redundant features, and thus, improve the performance of learned model in terms of predictive power, speed, and simplicity. Formally speaking, FS can be considered as an optimization problem in which an initial set of features $F = \{X_1, X_2, \ldots, X_n\}$ is given, where n represents the total number of features in the dataset. The objective is to pick a subset $F' = \{X'_1, X'_2, \ldots, X'_m\}$, where $m < n$, and try to maintain the classification accuracy higher.

The framework of FS consists of two major components: The search process that considers the different feature subsets, and the evaluation mechanism for these subsets. From evaluations perspective, FS methods can be categorized into two main categories: filter-based and wrapper-based methods. In filter-based methods, the subset is not evaluated over the training examples (i.e., it does not involve a specific learning algorithm); however, it depends on the information content and selects the most informative subset [9]. On the other hand, the wrapper-based methods evaluate the feature subsets by involving an external learning algorithm (classification algo-

rithms), where the model is trained using the selected subset and its performance is evaluated based on a validation dataset. Comparing these two types, filter-based methods are computationally faster than wrapper-based classes because the evaluation mechanism does not consider any classifier (external algorithm). However, the predictive power of wrapper-based techniques are usually better than filters, which strongly depend on the utilized learning algorithm.

In the searching process, different searching strategies can be employed, such as complete search, heuristic search, and random search [3]. However, when dealing with high-dimensional datasets that have N features, the total number of possible subsets is 2^N and therefore, the computational cost increases, exponentially. A complete search is exhaustive and inapplicable approach for FS problem. Moreover, the random technique is also impractical to handle FS methods because it may perform a complete search in the worst case. Heuristic methods can be a trade-off between the complete and random search approaches. In heuristic strategy, the solutions are iteratively evolved to be improved toward the near-optimal solutions based on a given objective function. Heuristic methods can obtain high-quality solutions within a reasonable time. Therefore, heuristic algorithms are employed with FS methods.

In this chapter, a wrapper-based FS algorithm using KNN classifier and a new binary HHO algorithm is developed for the first time to handle the high-dimensional, low sample FS tasks in medical applications.

3 Harris Hawks Optimization (HHO)

HHO is a novel optimizer that inspires the chain of action and reactions of a hunting process performed by hawks and rabbits. As it was revealed in the original paper of HHO, the core mathematical foundation of this optimizer makes it an effective optimizer in dealing with various constrained and unconstrained problems. In this optimizer, the search agents are updated using two phases of exploration and four phases of exploitation. It utilizes several time-varying mechanisms with a greedy scheme to enhance the quality of results. In this part, we briefly review the core

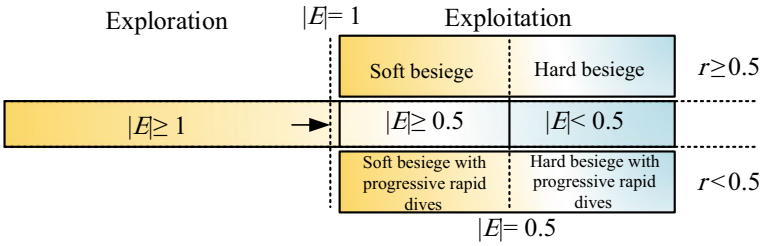


Fig. 1 Different steps of HHO

Table 1 Explanations of symbols used in the mathematical model of HHO

Description	Symbol
Position vector of hawks (search agents), location of ith hawk	X, X_i
Position of rabbit (best agent)	X_{rabbit}
Position of a random hawk	X_{rand}
Average position of hawks	X_m
Swarm size, iteration counter, maximum number of iterations	N, t, T
Random numbers inside (0,1)	$r_1, r_2, r_3, r_4, r_5, q$
Dimension, upper and lower bounds of variables	D, LB, UB
Escaping energy, initial state of energy	E, E_0

phases and equations of HHO, mathematically. The stages of HHO are represented in Fig. 1. Table 1 describes the symbols used in HHO's mathematical model.

3.1 Exploration Phase

In this phase, algorithm tries to enhance the diversity of population and scan all regions of the feature space. HHO has two simple phases to perform the exploration stage, which are obtained by:

$$X(t+1) = \begin{cases} X_{rand}(t) - r_1 \left| X_{rand}(t) - 2r_2 X(t) \right| & q \geq 0.5 \\ (X_{rabbit}(t) - X_m(t)) - r_3(LB + r_4(UB - LB)) & q < 0.5 \end{cases} \quad (1)$$

where

$$X_m(t) = \frac{1}{N} \sum_{i=1}^{N} X_i(t) \quad (2)$$

This average location can be obtained using any external function that can obtain a *center of gravity* for all agents.

3.2 Shifting from Exploration to Exploitation

To balance the main searching trends, an optimizer needs to properly shift the searching trends from extensive diversification to focused intensification. HHO uses the concept of decreasing energy of a rabbit when it escapes to handle this vital aspect of searching process. The model used here is time-varying and stochastic-based to ensure the dynamic and random nature of HHO. The energy rate of a prey is modeled by:

$$E = 2E_0(1 - \frac{t}{T}) \tag{3}$$

3.3 Exploitation Phase

3.3.1 Soft Besiege

After exploring favorite regions of the feature space, the HHO focuses on the vicinity of better solutions to finally reach optimum or near-optimum solution. These process is divided into four sub-level stages to efficiently exploit the search space using several alternative updating vectors. To perform a soft besiege, we need to use the following general rule:

$$X(t+1) = \Delta X(t) - E\,|JX_{rabbit}(t) - X(t)| \tag{4}$$

$$\Delta X(t) = X_{rabbit}(t) - X(t) \tag{5}$$

where $J = 2(1 - r_5)$ is the jump strength of the rabbit.

3.3.2 Hard Besiege

This phase is to put more emphasis to the exploitation power around some of high-quality solutions. This task is performed using Eq. (6):

$$X(t+1) = X_{rabbit}(t) - E\,|\Delta X(t)| \tag{6}$$

3.3.3 Soft Besiege with Progressive Rapid Dives

In soft besiege stage, there is a decision-making option for all hawks. Based on this rule, we represent the following rule in Eq. (7):

$$Y = X_{rabbit}(t) - E\,|JX_{rabbit}(t) - X(t)| \tag{7}$$

In HHO, LF-based patterns used to model this phase, which the rule can be obtained by:

$$Z = Y + S \times LF(D) \tag{8}$$

where S has the size of $1 \times D$ as a vector and LF shows the function that generate levy-based patterns. One of the ways to generate levy patterns is as follows:

$$LF(x) = \frac{u \times \sigma}{|v|^{\frac{1}{\beta}}}, \sigma = \left(\frac{\Gamma(1+\beta) \times sin(\frac{\pi\beta}{2})}{\Gamma(\frac{1+\beta}{2}) \times \beta \times 2^{(\frac{\beta-1}{2})}} \right)^{\frac{1}{\beta}} \tag{9}$$

where u, v are random parameters of LF and β is 1.5. However, any levy function can be used as an external function in HHO.

Hence, this phase is performed by Eq. (10):

$$X(t+1) = \begin{cases} Y \; if F(Y) < F(X(t)) \\ Z \; if F(Z) < F(X(t)) \end{cases} \tag{10}$$

where Y and Z are calculated using Eqs.(7) and (8).

3.3.4 Hard Besiege with Progressive Rapid Dives

In this stage, we have the following condition:

$$X(t+1) = \begin{cases} Y' \; if F(Y) < F(X(t)) \\ Z' \; if F(Z) < F(X(t)) \end{cases} \tag{11}$$

where Y' and Z' can be obtained by new rules in Eqs. (12) and (13).

$$Y' = X_{rabbit}(t) - E\,|JX_{rabbit}(t) - X_m(t)| \tag{12}$$

$$Z' = Y + S \times LF(D) \tag{13}$$

where $X_m(t)$ is obtained using Eq. (2). For more illustrations, please refer to original paper [8] and online materials.[2]

3.4 Pseudocode of HHO

The pseudocode of the HHO is described in Algorithm 1.

4 Proposed Binary HHO

In this section, a binary version of HHO (BHHO) should be developed to adapt the FS problem.

[2]Please visit these home pages that are publicly devoted to HHO algorithm: http://www.alimirjalili.com/HHO.html and http://www.evo-ml.com/2019/03/02/hho.

Algorithm 1 Pseudocode of HHO algorithm [8]

Inputs: N and T
Outputs: X_{rabbit}
Initialize $X_i (i = 1, 2, \ldots, N)$
while (stopping condition is not met) **do**
  Calculate the fitness values
  Set X_{rabbit} as the best solution
  for (each hawk (X_i)) **do**
    Update E_0 and jump strength J ▷ `E0=2rand()-1, J=2(1-rand())`
    Update E by Eq. (3)
    if $(|E| \geq 1)$ **then** ▷ `Exploration phase`
      Update the location vector by Eq. (1)
    if $(|E| < 1)$ **then** ▷ `Exploitation phase`
      if $(r \geq 0.5$ and $|E| \geq 0.5)$ **then** ▷ Soft besiege
        Update the location vector by Eq. (4)
      else if $(r \geq 0.5$ and $|E| < 0.5)$ **then** ▷ Hard besiege
        Update the location vector by Eq. (6)
      else if $(r < 0.5$ and $|E| \geq 0.5)$ **then** ▷ Soft besiege with progressive rapid dives
        Update the location vector by Eq. (10)
      else if $(r < 0.5$ and $|E| < 0.5)$ **then** ▷ Hard besiege with progressive rapid dives
        Update the location vector by Eq. (11)
Return X_{rabbit}

In literature, several approaches have been developed to adapt continuous algorithms with binary search spaces [10]. These binarization methods can be categorized into two major groups. The first one is called continuous-binary operator transformation in which the original real operators of metaheuristic equations are redefined into binary operators [11]. While, in the second group, which is called two-step binarization, the real operators are used without modifications, while the produced continuous solutions are converted into binary by utilizing two additional steps. The first step employs a transfer function (TF) that aims to transform the continuous solution R^n into an intermediate probability vector $[0, 1]^n$, where each element in this vector defines the probability of converting the corresponding element in R^n to 1 or 0. In the second step, the intermediate solution is transformed into binary by applying different binarization methods [10].

In literature, two main types of TFs were defined based on their shapes. The S-shaped TF as shown in Fig. 2a, which is firstly introduced by Kennedy and Eberhart [12] to convert the continuous PSO into binary using Eq. (14). While the V-shaped function shown in Fig. 2b was introduced by Rashedi et al. [13] to binarize GSA using Eq. (15), these two TFs are incorporated with two binarization rules (standard and complement), respectively.

We used the two basic TFs, i.e., S-shaped and V-shaped as in Eqs. (14) and (15), respectively, as a first step to binarize HHO algorithm.

$$T(x_i^j(t)) = \frac{1}{1 + e^{-x_i^j(t)}} \tag{14}$$

$$T(x_i^j(t)) = |\tanh(x_i^j(t))| \tag{15}$$

where x_i^j represents the jth dimension of the ith solution at iteration t, and $T(x_i^j(t))$ is the probability value obtained by TF.

As a second step of binarization process, the standard rule (Eq. 16) was used with the S-shaped function, while the complement rule (Eq. 17) was used with the V-shaped function to update the elements of the feature subset in the next iteration.

$$x_i^j(t+1) = \begin{cases} 0 & \text{If } rand < T(x_i^j(t+1)) \\ 1 & otherwise \end{cases} \tag{16}$$

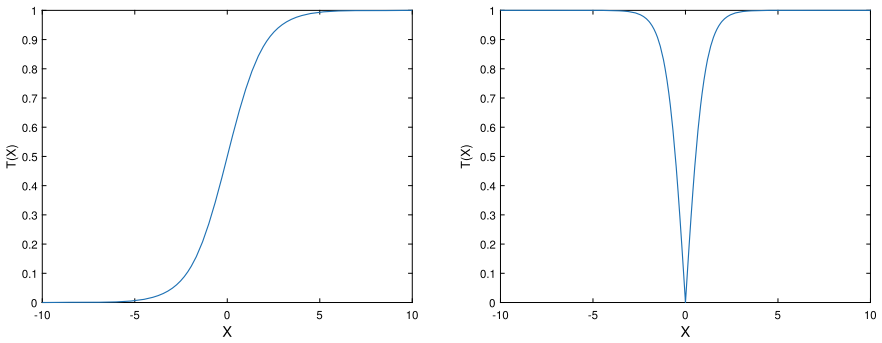
where $X_i^j(t+1)$ is the new binary value corresponds to the jth dimension of the ith solution in R^n, $T(x_i^j(t))$ is the probability value obtained by TF using Eq. (14), and r is a random number with uniform distribution in (0, 1) .

$$x_i^j(t+1) = \begin{cases} \neg x_i^k(t) & r < T(x_i^j(t+1)) \\ x_i^j(t) & otherwise \end{cases} \tag{17}$$

where $\backsim$ indicates the complement of $x_i^j(t)$, and $T(x_i)$ is the probability value obtained using Eq. (15)

5 BHHO-Based Feature Selection

In this chapter, a wrapper-based FS that considers the BHHO as a search algorithm is proposed for the first time in literature. Note that KNN classifier as an evaluator is



(a) S-shaped TF (b) V-shaped TF

Fig. 2 **a** S-shaped and **b** V-shaped transfer functions

used, as illustrated in Fig. 3. We used the KNN classifier [14] as one of the simplest, and most utilized nonparametric classification techniques. In addition, it has been widely used in literature and has shown a competitive performance with variety of FS methods [15].

When designing any optimizer, two main elements of the optimization problem should be considered: the solution representation and the evaluation function. For this purpose, candidate solutions should be encoded in a suitable way based on the nature of handled problem. It is an important step that implies the size of the search space and plays a key role in the overall efficiency of the optimizer. On the other hand, the evaluation function guides the search process by measuring the quality of the candidate solutions. FS has a binary nature, so each candidate solution can be represented as a one-dimensional vector with N elements $X = \{x_1, x_2, x_3, \ldots, x_N\}$, where N is the total number of features. Each element has a value of 0 which indicates that the feature is not selected or 1 which indicates that the feature is selected. The fitness of a feature subset is measured based on two contradictory metrics: the minimum number of selected features and the maximum classification accuracy. Since we are using a single-objective HHO, these two objectives are formulated using the fitness function in Eq. (18) [16].

$$\downarrow Fitness = \alpha\gamma_R(D) + \beta\frac{|R|}{|N|} \tag{18}$$

where $\gamma_R(D)$ is the classification error rate resulted by classifier, $|R|$ is the number of selected features by the optimizer, and $|N|$ is the total number of features $\alpha \in [0, 1]$, $\beta = (1 - \alpha)$ are two parameters [17].

6 Results and Discussion

In this section, we have performed several tests and experiments to detect the advantage, disadvantage, and efficacy of HHO and its place compared to previous optimizers. To have a fair comparison, we have to set all setting similarly and ensure that all algorithms have a same initial population.

Table 2 shows the used case studies that are high dimensional, low samples datasets. All datasets can be downloaded from an open public source.[3] A train/test model is necessary before performing any evaluation to substantiate the efficacy of the proposed BHHO algorithm, where 80% of the dataset were employed for training and 20% of it were processed for testing. All results and analysis are obtained using MATLAB 2017a, and overall results of 30 independent runs are compared. We used a system with an Intel Core i5 machine, 2.2 GHz CPU and 4 GB of RAM.

We highlighted the best results using a boldface format. In order to judge the difference of results and detect it is significant or not, we used Wilcoxon rank-

[3] http://www.gems-system.org/.

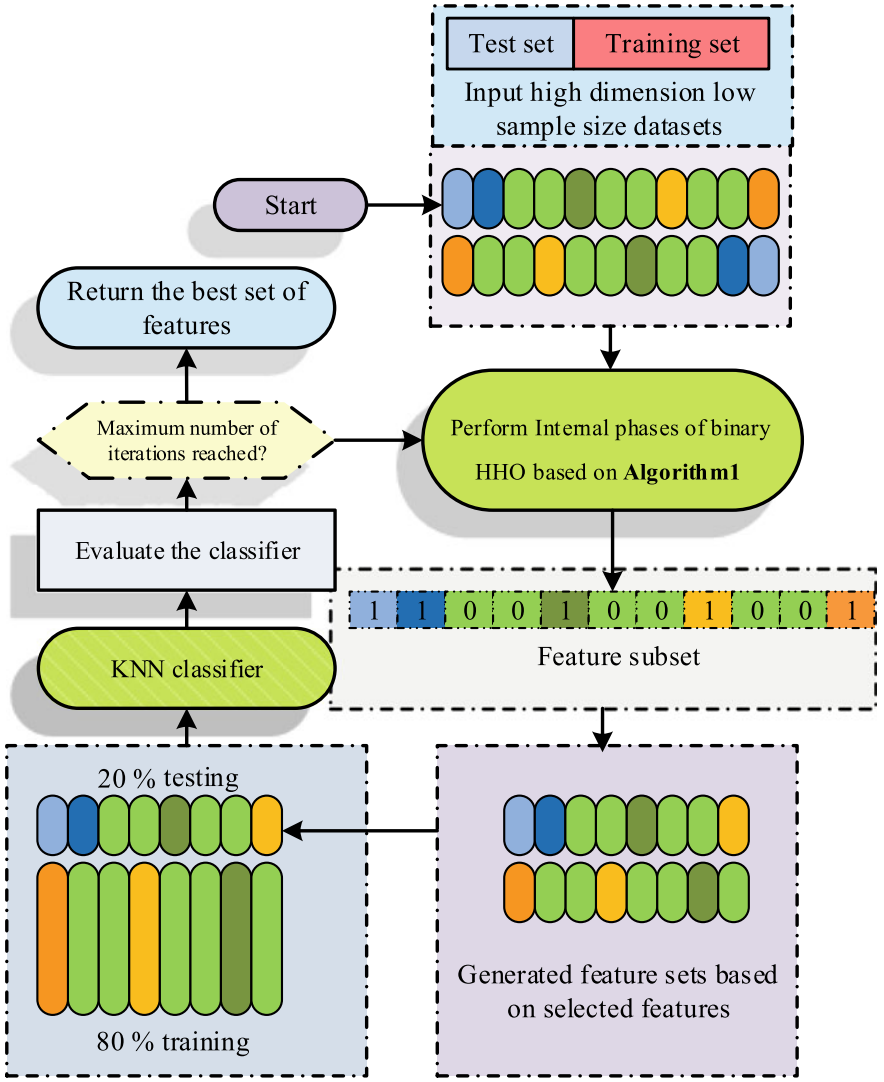


Fig. 3 Framework of the proposed HHO-based FS method

sum test with degree of 0.05 [19]. However, note that there are some new criticizes regarding the way we use p-value results to accept the differences or reject them.[4] There are also recommendations to improve the way we interpret the p-values [20]. In this study, we follow the current trends in using p-values to judge the significant [8]. Hence, if the p-value was lower than 0.05, the difference is significant and if not, the results are statistically similar.

[4]Please refer to https://www.nature.com/articles/d41586-019-00874-8.

Table 2 Summary of high-dimensional low samples data sets [18]

Dataset	No. of samples	No. of features	No. of classes
11_Tumors	174	12533	11
14_Tumors	308	15009	26
Brain_Tumor1	90	5920	5
Brain_Tumor2	50	10367	4
DLBCL	77	5469	2
Leukemia1	72	5327	3
Leukemia2	72	11225	3
Prostate_Tumor	102	10509	2
SRBCT	83	2308	4

In this work, we set the parameters of optimizers based on recommended settings in initial papers and related works on feature selection area. Table 3 shows the setting of internal parameters of optimizers.

6.1 Sensitivity Analysis

In this part, we are interested to investigate the sensitivity of binary variant to the initial population and number of iterations. Note that HHO has no initial user-defined parameter and its mechanisms are all randomized with time-varying nature. The other benefit from such experiments is to understand how to get the best results for binary HHO method before comparison with other methods. Such an analysis can show the size of the swarm, and the number of iterations have a significant impact on the performance or not.

Table 4 compared the accuracy rates realized by BHHO with different combinations of common parameters.

As per rank results in Table 4, we see that the binary HHO with 20 agents and 100 iterations is able to generate the best results for these cases. We utilized F-test to detect the best results as one of the well-known methods for raking different methods in a comparative test.

In Fig. 4, we set different sizes of swarm with a fixed iteration of 100 to analyze the sensitivity of HHO to population size. The convergence trends are also in accordance with the superiority of HHO when we use the population size of 20. Hence, we set to this setting for the next experiments and comparative studies.

Table 3 Used parameters settings

Config. Name	Value
Fitness function	
α	0.99
β	0.01
Common config.	
Number of runs	30
Population size	20
No. of iterations	100
Dimension	#features
Specific config.	
G_0 (for GSA)	100
α (for GSA)	20
a (for HHO)	From 2 to 0
Q_{min}(for BA)	0
Q_{max} (for BA)	2
A Loudness (for BA)	0.5
r Pulse rate (for BA)	0.5
ω (for PSO)	From 0.9 to 0.4
cp, cg (for PSO)	2
GA selection	Roulette wheel selection
Mutation Probability (in GA)	0.01
Crossover probability (in GA)	0.9
Elite size (in GA)	2
K for KNN	5

Table 4 Average classification accuracy obtained by BHHO with different combinations of common parameters

#iterations	50			100			150		
Population	5	10	20	5	10	20	5	10	20
11_Tumors	0.858	0.785	**0.877**	0.832	0.726	0.830	0.701	0.698	0.847
14_Tumors	0.539	0.587	0.551	0.611	**0.650**	0.541	0.603	0.648	0.581
Brain_Tumor1	0.889	0.883	0.902	0.909	0.785	0.980	0.889	0.944	**1.000**
Brain_Tumor2	0.700	0.717	0.800	0.700	0.787	**0.807**	0.610	0.630	0.517
DLBCL	0.938	0.875	0.777	0.921	0.904	**0.940**	**0.940**	0.938	0.802
Leukemia1	**1.000**	**1.000**	0.922	**1.000**	0.898	0.896	0.898	0.880	0.938
Leukemia2	0.736	**0.933**	0.867	0.867	0.871	0.916	**0.933**	0.811	0.867
Prostate_Tumor	0.951	0.929	0.841	**1.000**	0.905	**1.000**	0.887	0.865	0.905
SRBCT	0.945	0.888	0.886	0.941	0.839	**1.000**	0.943	0.855	0.859
Overall rank (F-test)	4.72	4.72	5.56	3.89	5.67	**3.44**	5.11	6.39	5.50

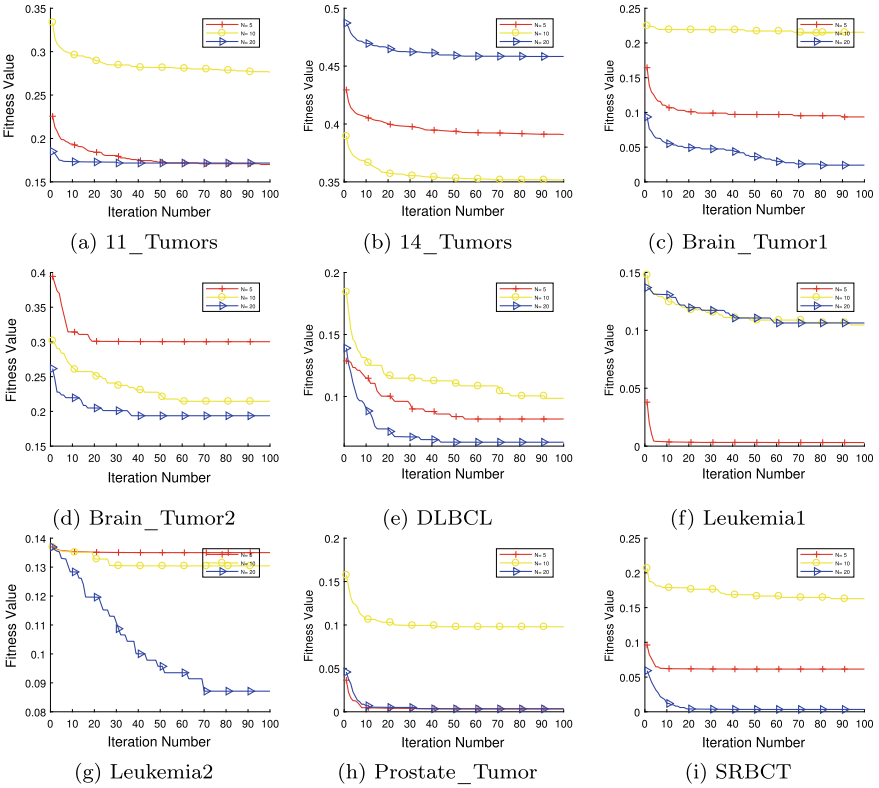


Fig. 4 Convergence behavior of BHHO with 100 iterations and different population sizes in dealing with all datasets

6.2 *Comparison Between BHHO with S-Shaped and V-Shaped TFs*

There are two general classes of TFs in literature: S-shaped and V-shaped TFs. Several binary optimizers such as binary GOA, DA, and SSA in FS works also utilized these functions. There are four classes of S-shaped cases and four types of V-shaped cases. In this section, we are interested to compare the proposed BHHO with S-shaped and V-shaped TFs to detect the suitable one. It is important to utilize a proper TF because the way we transfer the continuous space to binary space is highly significant in the final obtained classification results.

We used two classes of TFs and the results in terms of accuracy, selected feature, and fitness metrics are presented in Table 5.

As per results in Table 5, we see that the BHHO with S-shaped (SBHHO) has found higher accuracy rates for six test cases. We see that the SBHHO has reached to maximum accuracy for Prostate_Tumor and SRBCT cases. Based on selected

Table 5 Comparison between SBHHO and VBHHO in terms of all metrics

Dataset	Classification accuracy		Selected features		Fitness	
	SBHHO	VBHHO	SBHHO	VBHHO	SBHHO	VBHHO
11_Tumors	**0.830**	0.678	3651.43	**220.27**	**0.1717**	0.3203
14_Tumors	**0.541**	0.502	6214.57	**25.89**	**0.4583**	0.4963
Brain_Tumor1	**0.980**	0.946	2314.97	**80.80**	**0.0241**	0.0537
Brain_Tumor2	0.807	**0.900**	3172.10	**198.11**	0.1937	**0.0998**
DLBCL	0.940	**0.973**	1818.53	**1.21**	0.0631	**0.0279**
Leukemia1	0.896	**0.951**	1584.80	**62.41**	0.1064	**0.0490**
Leukemia2	**0.916**	0.838	3956.60	**111.22**	**0.0871**	0.1615
Prostate_Tumor	**1.000**	0.984	3489.40	**28.58**	**0.0033**	0.0171
SRBCT	**1.000**	0.947	784.50	**43.51**	**0.0034**	0.0555
Rank (W|T|L)	**6|0|3**	3|0|6	0|0|9	**9|0|0**	**6|0|3**	3|0|6

features, we see that VBHHO has superior results compared to the variant with S-shaped TF. Based on fitness results, we see that SBHHO reached the best rank. In this chapter, the first priority is with accuracy rates. Actually, based on application and priorities of decision makers, we can change the weight of objective functions. In this study, we supposed that the accuracy has a higher weight compared to the number of selected features. According to results, we detect that the HHO using S-shaped TF is able to realize better results.

6.3 Comparison with Other Algorithms

In this section, we compare the performance of the BHHO with S-shaped TF with other well-established optimizers in terms of different metrics such as accuracy of results. For this purpose, we utilized genetic algorithm (GA) [21] and binary versions of gravitational search algorithm (GSA) [13], Antlion optimizer (ALO) [22], bat algorithm (BAT/BA) [23], salp swarm algorithm (SSA) [24], and particle swarm optimizer (PSO) [12].

For all these methods, we recorded the average accuracy, number of features that are selected using the BHHO, and fitness values. All these results are gathered based on 30 runs.

Table 6 compares the results of SBHHO with those obtained by other peers. As per results in Table 6, it is observed that the SBHHO can obtain the best ranks on 11-Tumors, Brain-Tumor1, DLBCL, Prostate-Tumor, and SRBCT cases. The results reveal that the SBHHO has obtained the best place, followed by BALO, BPSO, BGSA, GA, BBA, and BSSA methods.

Table 7 provides the p-values of SBHHO compared to other peers. As per p-values, we detect that there is a significant difference between the SBHHO and its peers in

Table 6 Comparison of SBHHO versus other optimizers in terms of average classification accuracy

Dataset	BGSA	BPSO	BALO	BBA	BSSA	GA	SBHHO
11_Tumors	0.7506	0.8162	0.7143	0.6286	0.6765	0.8162	**0.8295**
14_Tumors	0.5122	0.5557	0.5942	0.4619	0.5322	**0.6381**	0.5412
Brain_Tumor1	0.8889	0.8333	0.9444	0.9185	0.8333	0.7778	**0.9796**
Brain_Tumor2	0.8852	0.6300	**0.9000**	0.8534	0.7000	0.6667	0.8074
DLBCL	0.8750	0.8208	0.8750	0.9021	0.7792	0.9375	**0.9396**
Leukemia1	0.8822	**0.9844**	0.9089	0.8778	0.8222	0.8444	0.8956
Leukemia2	**1.0000**	0.9333	0.9333	0.5667	0.8689	0.8667	0.9156
Prostate_Tumor	0.8556	0.8937	0.8587	0.8746	0.8191	0.9746	**1.0000**
SRBCT	0.9706	0.9628	0.9412	0.8804	0.8863	0.8804	**1.0000**
Mean rank (F_test)	3.722	3.722	3.111	5.056	5.833	4.333	***2.222***
Overall rank	3	3	2	5	6	4	**1**

Table 7 P-values of the Wilcoxon rank-sum test for the classification accuracy based on table 6 ($p > 0.05$ are shown in bold face)

Dataset	SBHHO versus					
	BGSA	BPSO	BALO	BBA	BSSA	GA
11_Tumors	1.11E−12	1.01E−04	2.71E−14	8.30E−13	2.71E−14	4.74E−04
14_Tumors	3.37E−09	3.51E−08	9.18E−12	2.50E−11	2.61E−04	2.13E−11
Brain_Tumor1	3.80E−13	3.80E−13	1.79E−07	4.15E−07	3.80E−13	3.80E−13
Brain_Tumor2	3.38E−08	2.63E−12	2.43E−13	**3.65E−01**	2.43E−13	2.43E−13
DLBCL	2.71E−14	1.32E−13	2.71E−14	1.27E−06	6.75E−13	**3.34E−01**
Leukemia1	**1.05E−01**	2.12E−10	**1.26E−01**	**2.21E−01**	6.45E−09	1.20E−05
Leukemia2	2.43E−13	2.70E−03	2.70E−03	3.96E−12	3.38E−08	5.36E−09
Prostate_Tumor	1.19E−13	1.97E−13	2.71E−14	5.64E−13	1.55E−13	1.02E−05
SRBCT	9.65E−06	1.79E−07	1.69E−14	5.77E−13	4.16E−14	5.65E−13

most of the cases. These results indicate that SBHHO is able to make a more stable trade-off between the core searching phases. It has a higher possibility to escape local optima and avoid immature convergence drawbacks, which results in better accuracy results.

Studying the outcomes in Table 8, we can observe the dominance of SBHHO in terms of selected features on 88.88% of cases. As per F-test results, we see the SBHHO is ranked one, followed by BBA, GA, BPSO, BGSA, BSSA, and BALO methods. If we observe the p-values in Table 9, the proposed SBHHO can outperform others in most of the cases, significantly.

Fitness results are presented in Table 10. It is seen from Table 10 that the SBHHO has outperformed all other competitors and reached the best rank. It outperforms all optimizers on 11-Tumors, Brain-Tumor1, DLBCL, Prostate-Tumor, and SRBCT

Table 8 Comparison of SBHHO versus other optimizers in terms of average number of features

Dataset	BGSA	BPSO	BALO	BBA	BSSA	GA	BHHO
11_Tumors	6239.93	6202.23	7083.57	5092.63	7160.63	5906.70	**3651.43**
14_Tumors	7517.57	7486.00	11298.97	**6063.40**	9472.67	7235.07	6214.57
Brain_Tumor1	2884.33	2822.57	2891.63	2404.77	2892.53	2670.63	**2314.97**
Brain_Tumor2	5128.90	4953.07	6309.53	4210.87	5095.43	4815.83	**3172.10**
DLBCL	2669.23	2538.83	3634.70	2193.73	2941.93	2459.10	**1818.53**
Leukemia1	2635.53	2552.90	3510.97	2191.17	2772.20	2443.33	**1584.80**
Leukemia2	5513.63	5394.40	5686.40	4567.67	5557.47	5221.63	**3956.60**
Prostate_Tumor	5209.60	5070.40	6658.57	4200.97	5766.40	4942.50	**3489.40**
SRBCT	1143.60	1071.13	1168.40	962.73	1193.17	1012.93	**784.50**
Mean rank (F_test)	5.111	4.000	6.667	1.889	6.222	3.000	***1.111***
Overall rank	5	4	7	2	6	3	**1**

Table 9 P-values of the Wilcoxon rank-sum test for the number of selected features based on table 8 ($p > 0.05$ are shown in bold face)

Dataset	SBHHO versus					
	BGSA	BPSO	BALO	BBA	BSSA	GA
11_Tumors	3.01E−11	3.02E−11	3.02E−11	3.02E−11	3.02E−11	3.02E−11
14_Tumors	7.22E−06	8.29E−06	1.33E−10	**4.38E−01**	4.50E−11	2.77E−05
Brain_Tumor1	1.53E−04	3.98E−04	1.04E−04	**9.19E−02**	1.25E−04	1.95E−03
Brain_Tumor2	3.01E−11	3.00E−11	3.02E−11	3.20E−09	3.01E−11	1.04E−10
DLBCL	3.01E−11	3.00E−11	3.02E−11	1.73E−06	3.02E−11	5.05E−E−10
Leukemia1	3.02E−11	4.50E−11	3.02E−11	1.25E−07	3.01E−11	4.60E−10
Leukemia2	8.47E−09	1.01E−08	7.10E−09	9.27E−04	7.77E−09	1.06E−07
Prostate_ Tumor	3.02E−11	3.02E−11	3.02E−11	3.96E−08	3.02E−11	3.00E−11
SRBCT	3.01E−11	2.15E−10	3.00E−11	3.82E−09	3.01E−11	5.04E−10

cases. If we consider the overall rankings, we see the SBHHO has achieved the best place, while BALO, BPSO, BGSA, BBA, GA, and BSSA are in the next preferences, respectively. According to the p-value results in Table 11, we observe the differences are meaningful in almost all cases, statistically.

One of the required measures for evaluating the efficacy of optimizers is to compare their convergence behaviors based on a fair comparison. Note that in all experiments, we utilized the same initial population. Convergence trends can reveal one important feature of optimizers: ability to avoid local optima and immature convergence. If an optimizer cannot make a stable trade-off between exploration and exploitation, it will be converged to local optima (higher fitness results instead of lower fitness rates) and then, we witness the immature convergence.

Table 10 Comparison of SBHHO versus other optimizers in terms of average fitness values

Dataset	BGSA	BPSO	BALO	BBA	BSSA	GA2	BHHO
11_Tumors	0.2519	0.1869	0.2885	0.3422	0.3260	0.1867	**0.1717**
14_Tumors	0.4879	0.4449	0.4093	0.5108	0.4695	**0.3631**	0.4583
Brain_Tumor1	0.1149	0.1698	0.0599	0.0367	0.1699	0.2245	**0.0241**
Brain_Tumor2	0.1186	0.3711	0.1051	**0.0370**	0.3019	0.3347	0.1937
DLBCL	0.1286	0.1820	0.1304	0.0654	0.2240	0.0664	**0.0631**
Leukemia1	0.1216	**0.0202**	0.0968	0.0655	0.1812	0.1586	0.1064
Leukemia2	**0.0049**	0.0708	0.0711	0.3997	0.1348	0.1367	0.0871
Prostate_Tumor	0.1480	0.1101	0.1462	0.0978	0.1846	0.0298	**0.0033**
SRBCT	0.0341	0.0415	0.0633	0.0721	0.1178	0.1228	**0.0034**
Mean rank (F_test)	3.889	3.778	3.556	4.000	6.000	4.444	***2.333***
Overall rank	4	3	2	5	7	6	**1**

Table 11 P-values of the Wilcoxon rank-sum test for the fitness values based on table 10 ($p > 0.05$ are shown in bold face)

Dataset	SBHHO versus					
	BGSA	BPSO	BALO	BBA	BSSA	GA
11_Tumors	3.00E−11	5.00E−10	2.99E−11	2.99E−11	2.99E−11	8.12E−07
14_Tumors	3.82E−09	8.35E−08	3.01E−11	3.02E−11	3.83E−06	3.00E−11
Brain_Tumor1	2.88E−11	2.95E−11	3.01E−11	1.87E−02	2.89E−11	2.91E−11
Brain_Tumor2	5.53E−03	2.81E−11	2.98E−11	1.20E−08	2.89E−11	2.86E−11
DLBCL	2.95E−11	2.86E−11	3.01E−11	**2.28E−01**	3.01E−11	5.05E−10
Leukemia1	1.02E−06	8.32E−08	**6.14E−02**	3.33E−03	2.98E−11	9.60E−10
Leukemia2	3.00E−11	2.86E−02	2.51E−02	3.00E−11	7.10E−11	2.85E−11
Prostate_Tumor	2.97E−11	3.02E−11	3.00E−11	3.02E−11	3.00E−11	3.00E−11
SRBCT	3.01E−11	3.32E−11	3.00E−11	3.01E−11	3.01E−11	3.00E−11

The convergence trends of SBHHO compared to other approaches are shown in Figs. 5 and 6. As per convergence curves in Figs. 5 and 6, we observe the superiority of SBHHO in dealing with 5 out of 9 datasets. However, we still see some stagnation behaviors for different methods including SBHHO on 14-Tumors, Brain-Tumor2, Leukemia1, and Leukemia2 cases.

Here, we calculate the rank of all methods considering all evaluated metrics. The obtained F-test results are shown in Table 12. As we can see in Table 12, the SBHHO achieves the best rank, followed by BBA, BPSO, GA, BGSA, BALO, and BSSA techniques, respectively.

There are several reasons that the binary HHO can show better and very competitive results compared to other competitors. The main reason is that it can establish a more stable equilibrium between exploration and exploitation. It has a dynamic

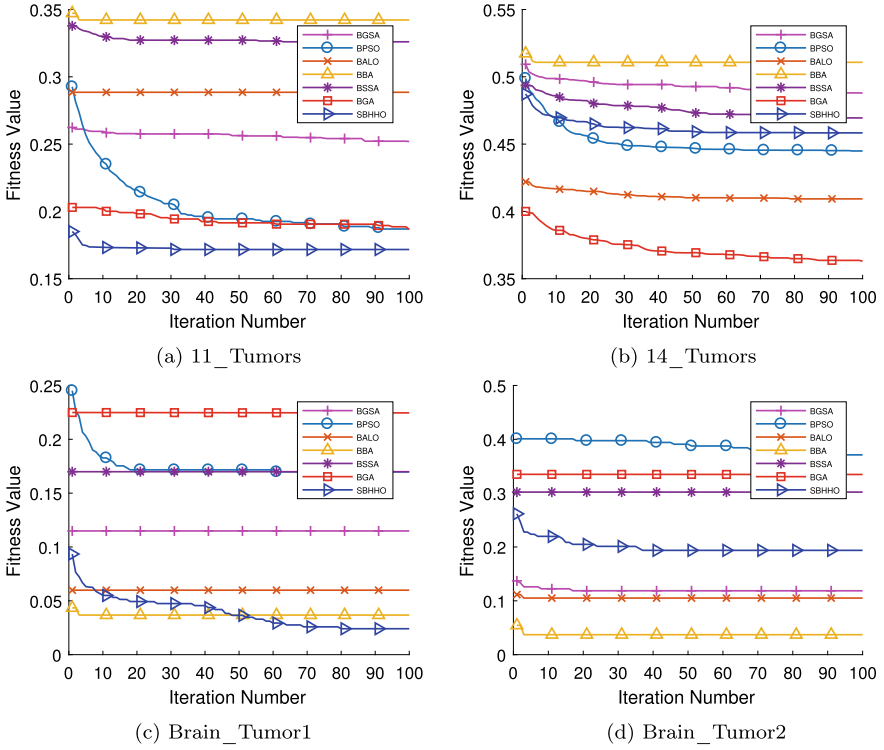
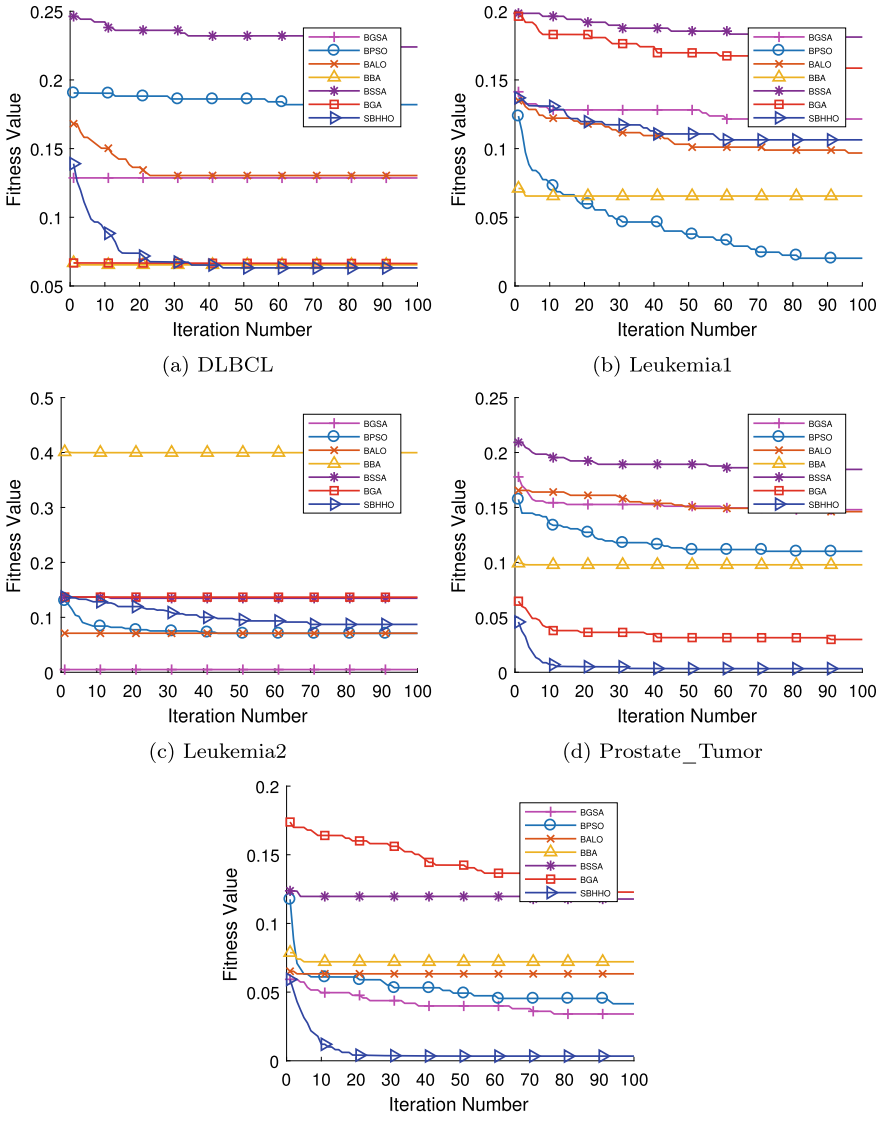


Fig. 5 Convergence curves of all approaches in dealing with 11_Tumors, 14_Tumors, Brain_Tumor1, and Brain_Tumor2 datasets

Table 12 Overall rank by the F-test for all approaches based on all metrics

Measure	BGSA	BPSO	BALO	BBA	BSSA	GA	SBHHO
Accuracy	3.722	3.722	3.111	5.056	5.833	4.333	**2.222**
Features	5.111	4.000	6.667	1.889	6.222	3.000	**1.111**
Fitness	3.889	3.778	3.556	4.000	6.000	4.444	**2.333**
Average rank	4.241	3.833	4.444	3.648	6.019	3.926	***1.889***
Final rank	5	3	6	2	7	4	1

structure with a relatively smooth switching behavior between the core phases using the time-varying E. It also utilizes greedy schemes that make it more exploitative in the last stages. In the case of immature convergence, HHO has several randomized operators and enough potential to jump out of local optima, effectively.



(a) DLBCL (b) Leukemia1

(c) Leukemia2 (d) Prostate_Tumor

(e) SRBCT

Fig. 6 Convergence curves of all approaches in dealing with DLBCL, Leukemia1, Leukemia2, Prostate_Tumor, and SRBCT datasets

7 Conclusions and Future Directions

Harris hawks optimizer is a new swarm-based method inspired by the life of intelligent birds. Two initial phases of HHO are devoted to exploration, and four phases are devoted to exploitative behaviors. In this chapter, we developed and substantiated the performance of a new binary HHO for the first time. We utilized 9 high-dimensional low sample cases that are challenging enough because a model has not adequate samples to be trained. In addition, when we use these datasets, we have a large feature space that makes the FS process much harder. The purpose of this chapter was to compare the efficacy in terms of different well-known metrics with BGSA, BSSA, GA, BPSO, BBA, and BALO methods. The results and analyses show the advantages of binary HHO with S-shaped TFs in terms of exploration and exploitation inclinations. The results suggest that the proposed HHO-based evolutionary FS technique can be utilized as a promising method in dealing with high-dimensional real-world datasets that have a low number of samples.

HHO is still new, and there are several new directions to extend the operations of HHO for tackling more real-world datasets. One of the potential directions is to apply HHO to hybrid wrapper–filter methods and evolutionary-based FS techniques. In next papers, we will develop new enhanced variants of binary HHO.

References

1. Pudil P, Novovičová J, Kittler J (1994) Floating search methods in feature selection. Pattern Recognit Lett 15:1119–1125
2. Seijo-Pardo B, Bolón-Canedo V, Alonso-Betanzos A (2019) On developing an automatic threshold applied to feature selection ensembles. Inf Fusion 45:227–245
3. Bolón-Canedo V, Alonso-Betanzos A (2019) Ensembles for feature selection: a review and future trends. Inf Fusion 52:1–12
4. Jin X, Xu A, Bie R, Guo P (2006) Machine learning techniques and chi-square feature selection for cancer classification using sage gene expression profiles. In: International workshop on data mining for biomedical applications. Springer, pp 106–115
5. Mafarja M, Aljarah I, Heidari AA, Hammouri AI, Faris H, Ala'M A-Z, Mirjalili S (2018) Evolutionary population dynamics and grasshopper optimization approaches for feature selection problems. Knowl-Based Syst 145:25–45
6. Heidari AA, Aljarah I, Faris H, Chen H, Luo J, Mirjalili S (2019) An enhanced associative learning-based exploratory whale optimizer for global optimization. Neural Comput Appl
7. Xu Y, Chen H, Heidari AA, Luo J, Zhang Q, Zhao X, Li C (2019) An efficient chaotic mutative moth-flame-inspired optimizer for global optimization tasks. Expert Syst Appl 129:135–155
8. Heidari AA, Mirjalili S, Faris H, Aljarah I, Mafarja M, Chen H (2019) Harris hawks optimization: algorithm and applications. Futur Gener Comput Syst 97:849–872
9. Kohavi R, John GH (1997) Wrappers for feature subset selection. Artif Intell 97:273–324
10. Crawford B, Soto R, Astorga G, Conejeros JG, Castro C, Paredes F (2017) Putting continuous metaheuristics to work in binary search spaces. Complexity 2017:1–19
11. Afshinmanesh F, Marandi A, Rahimi-Kian A (2005) A novel binary particle swarm optimization method using artificial immune system. In: EUROCON 2005-The international conference on Computer as a Tool, vol 1. IEEE, pp 217–220

12. Kennedy J, Eberhart RC (1997) A discrete binary version of the particle swarm algorithm. In: 1997 IEEE international conference on systems, man, and cybernetics, computational cybernetics and simulation, vol 5. IEEE, pp 4104–4108
13. Rashedi E, Nezamabadi-Pour H, Saryazdi S (2010) Bgsa: binary gravitational search algorithm. Nat Comput 9:727–745
14. Altman NS (1992) An introduction to kernel and nearest-neighbor nonparametric regression. Am Stat 46:175–185
15. Liao T, Kuo R (2018) Five discrete symbiotic organisms search algorithms for simultaneous optimization of feature subset and neighborhood size of knn classification models. Appl Soft Comput 64:581–595
16. Mafarja M, Aljarah I, Heidari AA, Faris H, Fournier-Viger P, Li X, Mirjalili S (2018) Binary dragonfly optimization for feature selection using time-varying transfer functions. Knowl-Based Syst 161:185–204
17. Faris H, Mafarja MM, Heidari AA, Aljarah I, Ala'M A-Z, Mirjalili S, Fujita H (2018) An efficient binary salp swarm algorithm with crossover scheme for feature selection problems. Knowl-Based Syst 154:43–67
18. Statnikov A, Aliferis CF, Tsamardinos I, Hardin D, Levy S (2004) A comprehensive evaluation of multicategory classification methods for microarray gene expression cancer diagnosis. Bioinformatics 21:631–643
19. Luo J, Chen H, Heidari AA, Xu Y, Zhang Q, Li C (2019) Multi-strategy boosted mutative whale-inspired optimization approaches. Appl Math Model
20. Benjamin DJ, Berger JO (2019) Three recommendations for improving the use of p-values. Am Stat 73:186–191
21. Goldberg DE, Holland JH (1988) Genetic algorithms and machine learning. Mach Learn 3:95–99
22. Emary E, Zawbaa HM, Hassanien AE (2016) Binary ant lion approaches for feature selection. Neurocomputing 213:54–65
23. Nakamura RY, Pereira LA, Costa KA, Rodrigues D, Papa JP, Yang X-S (2012) Bba: a binary bat algorithm for feature selection. In: 2012 25th SIBGRAPI conference on graphics, patterns and images. IEEE, pp 291–297
24. Sayed GI, Khoriba G, Haggag MH (2018) A novel chaotic salp swarm algorithm for global optimization and feature selection. Appl Intell 48:3462–3481

A Review of Grey Wolf Optimizer-Based Feature Selection Methods for Classification

Qasem Al-Tashi, Helmi Md Rais, Said Jadid Abdulkadir, Seyedali Mirjalili and Hitham Alhussian

Abstract Feature selection is imperative in machine learning and data mining when we have high-dimensional datasets with redundant, nosy and irrelevant features. The area of feature selection deals reducing the dimensionality of data and selecting only the most relevant features to increase the classification performance and reduce the computational cost. This problem has exponential growth, which makes it challenging specially for datasets with a large number of features. To solve this problem, a wide range of optimization algorithms are used of which grey wolf optimizer (GWO) is a recent one. This book chapter provides a brief review of the latest works on feature selection using GWO.

Keywords Grey wolf optimization · Feature selection · Classification · Machine learning · Artificial intelligence

Q. Al-Tashi · H. Md Rais · S. J. Abdulkadir · H. Alhussian
Department of Computer and Information Sciences, Universiti Teknologi Petronas, Bandar Seri Iskandar, 32610 Perak, Malaysia
e-mail: qasemacc22@gmail.com; qasem_17004490@utp.edu.my

H. Md Rais
e-mail: helmim@utp.edu.my

S. J. Abdulkadir
e-mail: saidjadid.a@utp.edu.my

H. Alhussian
e-mail: seddig.alhussian@utp.edu.my

Q. Al-Tashi
University of Albydha, Albydha, Yemen

S. J. Abdulkadir
Centre for Research in Data Science (CERDAS), Universiti Teknologi Petronas, Perak, Malaysia

S. Mirjalili (✉)
Torrens University
Australia, Brisbane, QLD 4006, Australia
e-mail: ali.mirjalili@gmail.com

Griffith University, Brisbane, QLD 4111, Australia

S. Mirjalili et al. (eds.), *Evolutionary Machine Learning Techniques*,
Algorithms for Intelligent Systems, https://doi.org/10.1007/978-981-32-9990-0_13

1 Introduction

Data mining has been identified in recent years as a fast-growing area of information technology research. This is due to the daily collection of massive data and the need to convert this data into useful information [1]. Data mining is a process in which patterns are discovered and knowledge extracted from a wide dataset. Data mining tasks include association analysis, detection of anomalies, clustering, regression and classification [2]. Feature selection is one of the main preprocessing steps aims to eliminate redundant and irrelevant attributes within a dataset. Commonly, feature selection methods are classified into two groups as wrappers or filters methods [3, 4]. Three categories of feature selection have been classified by some scientists which are: wrapper, filter and embedded methods [5]. There are a diverse kind of classifiers utilized with feature selection for learning and predicting the models, such as K-nearest neighbor (KNN), support vector machine (SVM), decision tree, artificial neural network (ANN) and naïve Bayes.

In the literature, the progress of optimization approaches encouraged through nature or interaction of animals to find food sources has increased demand [6]. Various optimization techniques have been established using biology-based swarm intelligence. For example, in 2014 Mirjalili et al. have developed a recent optimization technique named grey wolf optimization (GWO) [7]. This technique is essentially driven by the searching mechanism of grey wolves for the finest manner to hunt prey. GWO essentially utilized the nature mechanism, in which the package hierarchy is used to organize the wolves' different characters. Additionally, the candidates of the package are separated into four categories rely on the type's character of the wolf, which helps to advance the process of hunting. The four categories are alpha, beta, delta and omega. Alpha represents the fittest solution for the search. GWO has received considerable attention in the field of feature selection problems and has increased year by year. Figure 1 shows the number of studies conducted using GWO

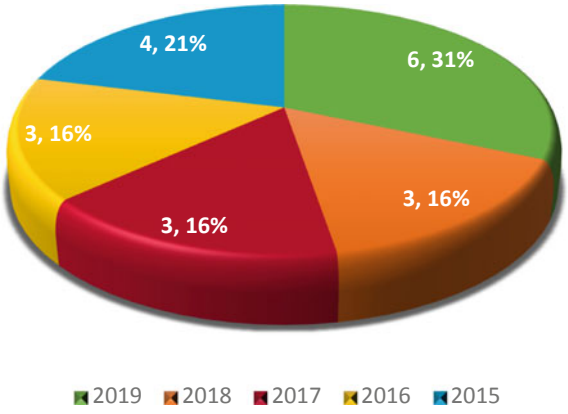


Fig. 1 Studies on feature selection using GWO

in feature selection from 2015 to 2019. The details of these studies are presented in Sect. 5.

The rest of this review is presented as follows. In the second section, feature selection concept and methods are defined, whereas the GWO inspiration and mathematical equations are introduced in the third section. Fourth section discussed the grey wolf feature selection mechanism. The performance of GWO-based feature selection is criticized and reviewed in the fifth section. The final section presents the conclusion and some research directions to be studied.

2 Feature Selection

Feature selection plays a significant task in machine learning and data mining, where high-dimensional datasets involve redundant, nosy and irrelevant features. Therefore, feature selection aims to reduce the dimensionality of the data and select only the most relevant features to increase the classification performance and reduce the computational cost [8]. Since the search space is huge when the number of features is large, the problem of finding an optimum feature is considered to be a complicated and complex problem.

The main objective of feature selection is to determine a small subset of features from a particular problem domain which reflects a high classification performance. In general, the selection of features is used for four reasons: first, to simplify data in order to make it easier for users in particular researchers to use and interpret data; second, to consume a short time by selecting only the important processing feature; third, to avoid the dimensionality curse; and finally, to improve generalization by reducing overfitting (reducing variance). Once these four reasons achieved, the misleading feature, irrelevant and redundant features are successfully removed which resulted in a good produced selection of features. Figure 1, adapted from [9], illustrates the process of feature selection (Fig. 2).

As stated in the introduction part, there are commonly two categories of feature selection algorithms: filter and wrapper approaches [3, 4]. Their key contrast is that in the feature subset evaluation step, wrapper approaches incorporate a classification algorithm. A wrapper method utilizes the classification algorithm to assess the goodness (e.g., accuracy) of the features selected as a "black box," while a process of selection for a filter method is independent of any algorithm for classification. Wrapper methods are claimed to be computationally more expensive and less general than filters. Nevertheless, filters disregard the goodness of the selected features on the algorithm of classification, whereas wrappers assess the classification performance-based subsets of feature, and this usually leads to better wrapper performance for classification than filters [3, 10, 11]. Moreover, some scientists classify the methods of feature selection into three classes: filter, wrappers and embedded methods [10, 11], where embedded method integrates both the selected features and the classifier into a single process. Figure 3 shows a summary of the feature selection approaches.

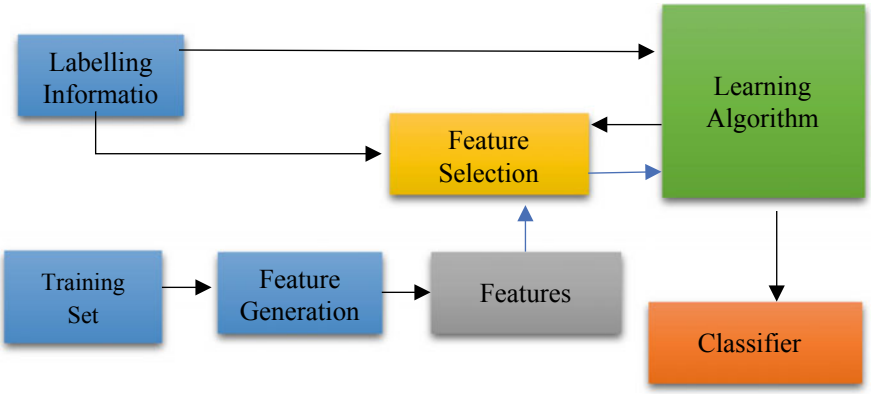


Fig. 2 General framework of feature selection

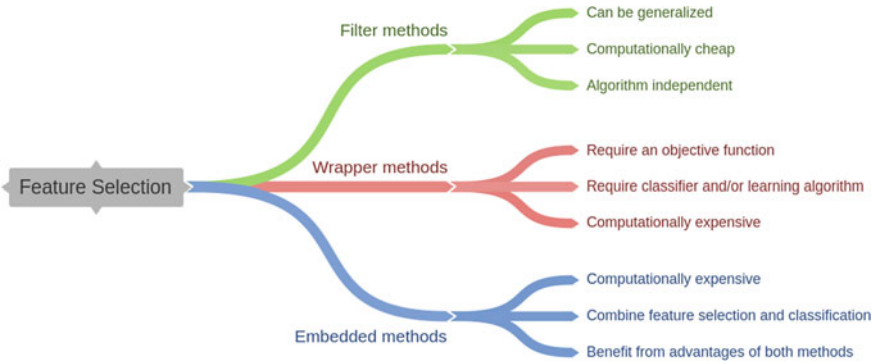


Fig. 3 Summary of feature selection approaches

3 Grey Wolf Optimization (GWO)

The main inspiration and the simple mathematical model of GWO algorithm are clarified in this section.

3.1 GWO Inspiration

As mentioned in the introduction part, GWO is a recent technique in the family of swarm intelligence. The GWO is encouraged by the guidance and hunting behavior of the grey wolf packs. There is a mutual social hierarchy in each group of grey wolves that identifies the dominance and power. The most influential wolf in hunting, feeding and migration is alpha, which guides the whole group. The second strongest

wolf is beta which will be in the lead if alpha is dead or sick. Omega and delta are less influential than alpha and beta. This type of social intelligence is the GWO algorithm's main inspiration.

3.2 GWO's Mathematical Model

Basically, the mathematical model of GWO is consisting of: encircling, hunting and attacking the prey and have been outlined as following:

3.2.1 Encircling the Prey

In mathematics, the encircling behavior of the grey wolf can be expressed as:

$$\vec{X}(t+1) = \vec{X}_p(t) + \vec{A}.\vec{D} \tag{1}$$

where $\vec{X}$ is the vector's position of the grey wolf, $\vec{X}_p$ is the vector's position of the prey, (t) is iterations number, and $\vec{A}$ is a vector of coefficient. Whereas $\vec{D}$ can be expressed as follows:

$$\vec{D} = \left| \vec{C} \cdot \vec{X}_p(t) - \bar{X}(t) \right| \tag{2}$$

$\vec{C}$ is a vector of coefficient, both coefficient vectors $\vec{A}$, $\vec{C}$ can be mathematically expressed as follows:

$$\vec{A} = 2\overrightarrow{a} \cdot \overrightarrow{r_1} - \overrightarrow{a} \tag{3}$$

$$\vec{C} = 2 \cdot \overrightarrow{r_2} \tag{4}$$

where $\overrightarrow{r_1}$ and $\overrightarrow{r_2}$ are vectors randomly in a range of [0, 1]. Whereas $\overrightarrow{a}$ is a set vector reduces linearly over iterations from 2 to 0.

3.2.2 Hunting the Prey

The grey wolf's hunting behavior is mathematically expressed, α is considered to be the finest applicant for the solution, and β and δ are expected to have more knowledge about the prey's possible positions. Therefore, the three finest solutions gained so far are kept and force ω wolf to modify their position based on the finest position in the decision space. This hunting behavior can be expressed as the following calculation:

$$\overrightarrow{D_\alpha} = \left|\overrightarrow{C_1} \cdot \overrightarrow{X}_\alpha - \overrightarrow{X}\right|, \overrightarrow{D_\beta} = \left|\overrightarrow{C_2} \cdot \overrightarrow{X}_\beta - \overrightarrow{X}\right|, \overrightarrow{D_\delta} = \left|\overrightarrow{C_3} \cdot \overrightarrow{X}_\delta - \overrightarrow{X}\right| \quad (5)$$

$$\overrightarrow{X_1} = \overrightarrow{X_\alpha} - A_1 \cdot \left(\overrightarrow{D_\alpha}\right), \overrightarrow{X_2} = \overrightarrow{X_\beta} - A_2.\left(\overrightarrow{D_\beta}\right), \overrightarrow{X_3} = \overrightarrow{X_\delta} - A_3.\left(\overrightarrow{D_\delta}\right) \quad (6)$$

$$\overrightarrow{X}(t+1) = \frac{\overrightarrow{X_1 + X_2 + X_3}}{3} \quad (7)$$

3.2.3 Attaching the Prey

Attacking the prey by grey wolfs is expressed based on a vector $\overrightarrow{a}$ where it is a randomly vector whose value is within the range $[-a,a]$, the value of $\overrightarrow{a}$ is reduced linearly from 2 to 0 over iterations and can be expressed as the following:

$$\overrightarrow{a} = 2 - t \cdot \frac{2}{\max\limits_i ter} \quad (8)$$

where $\max\limits_i ter$ is the total number of iterations for the optimization and t is the iteration number.

4 Grey Wolf Feature Selection Mechanism

The main step in solving GWO's selection of features is to demonstrate the subset of features in the representation of the solution. Figure 4 illustrates the representation of the solution. The solution's length is denoted by d, where d represents an integer of the feature. The position of the solution can take a value of "1" or "0." If the value of the bit is equal to 0, feature not selected; whereas if the value of the bit is equal

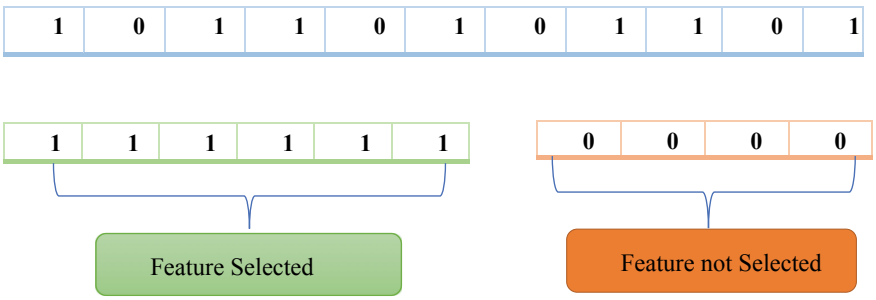


Fig. 4 Representation of solution in feature selection

to 1, feature is selected. Consequently, a value of 1's specifies the size of the subset feature.

GWO is therefore only appropriate for continuous search problems, and agents in GWO can move continuously in the search space because they have continuous real-domain position vectors. Thus, it is impossible to use the algorithms described in Sect. 3 to solve feature selection problems without amendment. There should be an operator to convert the original GWO into its binary version to suit the problem of feature selection. There were a numerous of binary operators which have been suggested in the literature such as: sigmoid() [12], crossover() [13], and/or tanh() [14] functions. Here is an example of how to covert GWO into BGWO, the wolf's update mechanism is a function of three vector positions, named $x_1, x_2; x_3$ pushes each wolf forward to the three finest solutions. To make the agent work in binary space, you can modify the location update (5) to the following equation [12]:

$$x_d^{t+1} = \begin{cases} 1 \text{ if sigmoid}\left(\frac{x_1+x_2+x_3}{3}\right) \geq rand \\ 0 \text{ otherwise} \end{cases} \tag{9}$$

where x_d^{t+1} is an updated binary position in dimension d at iteration t, whereas *rand* is a number randomly derived from a distribution uniform $\in [1, 0]$, and sigmoid (a) is indicated as follows [12]:

$$\text{sigmoid}(a) = \frac{1}{1 + \text{e}^{-10(x-0.5)}} \tag{10}$$

x_1, x_2, x_3 in (6) are modified and modeled using the below equations [12]:

$$\begin{aligned} x_1^d &= \begin{cases} 1 \text{ if } \left(x_\alpha^d + bstep_\alpha^d\right) \geq 1 \\ 0 \text{ otherwise} \end{cases} \\ x_2^d &= \begin{cases} 1 \text{ if} \left(x_\beta^d + bstep_\beta^d\right) \geq 1 \\ 0 \text{ otherwise} \end{cases} \\ x_3^d &= \begin{cases} 1 \text{ if } \left(x_\delta^d + bstep_\delta^d\right) \geq 1 \\ 0 \text{ otherwise} \end{cases} \end{aligned} \tag{11}$$

where $x_{\alpha,\beta,\delta}^d$ is the wolf's alpha, beta and delta position's vector in dimension d, whereas $bstep_{\alpha,\beta,\delta}^d$ is a step of binary in dimension d, that can be expressed as follows [12]:

$$bstep_{\alpha,\beta,\delta}^d = \begin{cases} 1 \text{ if } cstep_{\alpha,\beta,\delta}^d \geq rand \\ 0 \text{ otherwise} \end{cases} \tag{12}$$

where $cstep_{\alpha,\beta,\delta}^d$ is a continuous value of d's and can be expressed as in (13) [12], and d specifies dimension, whereas *rand* a value randomly derived from a distribution uniform $\in [1, 0]$:

$$cstep^{d}_{\alpha,\beta,\delta} = \frac{1}{1 + e^{-10\left(A^{d}_{1} D^{d}_{\alpha,\beta,\delta} - 0.5\right)}} \tag{13}$$

5 Feature Selection Methods Based GWO

As can be seen from the literature, the GWO algorithm has been utilized in many fields as a feature selection technique including the following fields: facial emotion recognition, EMG signal classification, disease diagnosis, gene selection and intrusion detection systems. In current year 2019, several studies have been working with GWO as a feature selection technique. For instance, in [15], Abd AL-Bast et al. present an effective face recognition system, they used GWO for feature selection, and k-NN for classification, they have evaluated their proposed method using Yale face database. They compared the performance of their method with the performance of other face recognition system. Their obtained results were better in terms of both accuracy and run time. In the same year, Al-Tashi et al. [12] proposed a binary hybrid GWO with PSO for feature selection and for classification they used k-NN classifier. They have evaluated their proposed method using 18 standard benchmark datasets collected from UCI repository. They have compared their method with four well-known approaches such as PSO, GWO, GA and hybrid WOSA-2. Their obtained result was better in term of classification accuracy, number of selected feature as well as computational time. Another work by [16], where a multi-strategy ensemble GWO was proposed. They integrated three updated search approaches to modify the solutions by first enhancing global best to expand local search capability of GWO, second adjustable approach by inserts a dimension of one vector into the GWO framework and finally the disperse foraging strategy based on parameter self-adjustment. They applied a numerous function of CEC2014 for experiments. They compared their attained outcomes with three other improved GWO as well as to other related work algorithms. Additionally, feature selection is applied to examine the efficiency of MEGWO. Their experimental outcomes demonstrated that the proposed method showed better performance in terms of convergence speed and accuracy. In [17], the authors proposed a diagnostic method that combines gas chromatography mass and machine learning (GC–MS), they used chaos enhanced grey wolf optimization (EGWO) to find the best subset feature to enhance the identification accuracy. For classification, they employed the extreme learning machine (ELM). They compared their method with six other methods. Their obtained outcomes had better performance in diagnosing patients with PQ poisoned. Another work in the same year was made by [18], the authors proposed a modified GWO (MGWO) which acts as a search strategy for feature selection. For classification, they used random forest, k-nearest neighbor classifier and decision tree. They evaluated their approach using several types of datasets of voice, speech and handwriting. Their obtained outcomes show that the proposed method can reduce the features number and increase the accuracy. Another work in the same direction made by Alzubi et al. [19], they proposed

a modified MBGWO for intrusion detection system, SVM is used. They employed NSL-KDD dataset to assess their proposed method. They compared their method with state-of-the-art methods such as BGWO, AdaBoost and PSO-discretize-HNB. Their method obtained a competitive result.

In 2018, much research studies have been conducted again using GWO as a feature selection technique. For example, in [20], the author proposed facial emotion recognition system. Their used GWO chooses the best features. A GWO-based neural network (NN) is then used to classify emotions from selected features. They compared the proposed method with conventional methods such as NN-LM and NN-PSO. The results they achieved are superior to traditional methods in terms of accuracy. Another work in the same year, Al-Tashi et al. [21], proposed a wrapper feature selection where GWO used as feature selection and SVM as a classifier for coronary artery diseases classification. They evaluated their method on Cleveland heart dataset and compared their proposed method with other competitive methods such as ABC, GA algorithms. Their obtained results outperformed the other approaches in term of classification accuracy, specificity and sensitivity. In the same direction, another study proposed by Jingwei et al. [22], the authors proposed a competitive BGWO (CBGWO) for EMG signals classification to handle the problem of feature selection. They compared their method with some competitive methods from the literature. Their obtained outcomes demonstrated the superiority of their method in both classification performance and feature reduction.

Moreover, in 2017 also much research works have been conducted using GWO as a feature selection. Anita and Satish [14] proposed two different wrapper approaches aiming to select finest feature to enhance cervix lesions classification. First, scalarized approach MOGWO, whereas the second is a nondominated sorting NSGWO. For evaluation, they used CT-Scan images of 62 patients. Their methods were compared with MOGA and firefly algorithm (MOFA). Their obtained results demonstrated better performance compared to other competitive methods. Also, they have conducted another experiment on microarray gene datasets. Their obtained results demonstrated better performance than other methods. In [23], the authors proposed a wrapper feature selection for medical diagnosis. They proposed an improved grey wolf optimization (IGWO) to find the finest feature for medical data. At the beginning, GA was utilized to produce the positions then GWO was used, then kernel extreme learning machine has been used for classification. They compared their approach against two disease datasets with two other competitive methods GA and GWO. Their obtained results outperformed the other two GA and GWO. Another study by [24], the authors proposed a wrapper feature selection for Parkinson's disease classification. They used GWO for feature selection and k-NN, SVM and naïve Bayes classifiers for classification. They have compared their proposed method with four state-of-the-art methods which are: PSO, GA, binary bat algorithm (BBA) and modified cuckoo search algorithm (MCS). They used speech dataset for evaluation. Their obtained results outperformed the other methods in terms of accuracy and number of features selected.

The following research studies have been mentioned in the review studied by Faris et al. [25]. In 2015, a wrapper feature selection approach has been introduced

by Emary et al. [26]. They have proposed a binary BGWO to select the subset of features and k-NN classifier as a fitness function to assess the selected features subsets. 8 benchmark datasets collected from UCI repository are used for evaluating their method. They compared their method with PSO and GA. Their attained outcomes outperformed the other methods in term of accuracy and reduced number of features. In 2016, the same authors Emary et al. [13] have extended their work and proposed two wrapper feature selection methods based binary GWO using sigmoid and crossover binary operators. Utilizing the same k-NN classifier. They used 18 datasets to evaluate their proposed method. Also, they compared their approaches with PSO and GA. The attained results assure their earlier outcomes. The same authors again in 2015 [27] have introduced multi-objective MOGWO for attribute reduction. Firstly, GWO used filter-based mutual information to find a set of optimal features. Secondly, GWO wrapper method was employed using k-NN classifier performance to improve the gained solutions for better classification accuracy. They compared their method against PSO, GA and another single objective method. Their obtained results outperformed other methods by achieving much robustness and stability. Furthermore, in 2015 Vosooghifard et al. [28] utilized the BGWO wrapper method on gene expression data for the classification of cancer. They used a decision tree C4.5 as a classifier with cross-validation. Their method incorporated the classification accuracy only. They used 10 microarray cancer datasets for evaluations and compared their method with different classifiers including SVM, multilayer perceptron (MLP) and self-organizing map (SOM). In 2016, Yamany et al. [29] have proposed a filter-based feature reduction method. They utilized GWO to select the optimal features and rough set fitness function for classification. They used 11 UCI data for evaluations and compared their proposed method with a GA and other rough set reduction methods. Their method obtained a competitive result. Also in 2016, Medjahed et al. [30] proposed a slightly modified GWO for band selection problem. They used three benchmark hyperspectral datasets for experiment namely: Pavia University, Indian Pine and Salinas. They compared their method with the state-of-the-art methods such as: mRmR, cmim, Relief, GA, PSO, GSA and BBA. Their obtained results can efficiently examine the spectral band selection and offer a high accuracy rate. Furthermore, in [31], they have used GWO to search the feature space to find optimal feature subset that improves classification accuracy. Firstly, GWO used filter-based based mutual information. Secondly, wrapper method utilized for guiding classifier performance. The proposed method measured and compared against several other metaheuristic algorithms with the help of NSL-KDD dataset. Table 1 illustrates the summary of GWO-based feature selection state-of-the-art methods from 2015 to 2019.

In summary, as can be noticed from Table 1 there has been less studies on filter comparing to wrapper method, and this is due to the fact that filters disregard the goodness of the subset of features on the algorithm of classification; however, wrappers assess the classification performance-based subsets of feature, and this usually leads to better wrapper than filters in classification performance [3, 10, 11]. In addition, wrapper methods are claimed to be less effective than filters methods, nonetheless experimentations have proved that is not true in most cases

Table 1 Summary of GWO-based feature selection

No	Author	Algorithm	Application	FS method	Classifier
1	[15]	GWO	Yale face database	Wrapper	k-NN
2	[12]	BGWOPSO	UCI benchmark datasets	Wrapper	k-NN
3	[16]	MEGWO	UCI benchmark datasets	Wrapper	k-NN
4	[17]	EGWO	PQ-poisoned patients	Wrapper	ELM
5	[18]	MGWO	Voice, handwriting (spiral and meander) and speech datasets	Wrapper	Random forest, k-NN & decision tree
6	[19]	MBGWO	Intrusion detection NSL-KDD dataset	Wrapper	SVM
7	[20]	GWO	Facial emotion recognition	Wrapper	GWO-Neural network (NN)
8	[21]	GWO	Disease diagnosis	Wrapper	SVM
9	[22]	CBGWO	EMG signals classification	Wrapper	k-NN
10	[14]	MOGWO NSGWO	Cervix lesion dataset (cancer diagnosis) + microarray gene datasets	Wrapper	SVM
11	[23]	IGWO	Medical diagnosis	Wrapper	KELM
12	[24]	GWO	Parkinson's diagnosis	Wrapper	k-NN, SVM & Naïve Bayes
13	[26]	BGWO	UCI benchmark datasets	Wrapper	k-NN
14	[13]	bGWO1 bGWO2	UCI benchmark datasets	Wrapper	k-NN
15	[27]	MOGWO	UCI benchmark datasets	Filter Wrapper	MI k-NN
16	[28]	BGWO	Microarray cancer datasets	Wrapper	C4.5 decision tree

(continued)

Table 1 (continued)

No	Author	Algorithm	Application	FS method	Classifier
17	[29]	GWO	UCI benchmark datasets	Filter	Rough set
18	[30]	Slightly modified GWO	Benchmark hyperspectral datasets	Wrapper	k-NN
19	[31]	GWO	intrusion detection NSL-KDD dataset	Filter Wrapper	MI k-NN

[32]. For instance, the rough set theory [29] which is filter method, it takes more time than a wrapper approach. Despite the existence of rapid filter methods, for instance, mutual information [27, 31], the performance of classification is generally worse comparing to the majority of wrapper methods.

The GWO, as mentioned above, has been extensively used to solve problems with selection of features. The main explanations for the algorithm's success are the simplicity of inspiration, few control parameters and the behavior of adaptive exploration. However, like other metaheuristic algorithms, it has some drawbacks and unavoidable disadvantages as follows: A binary operator should be equipped with GWO in order to solve binary problems (feature selection). In addition, when solving a large number of variables and local solution problems, fast convergence and the use of accelerated resulted to local solutions. Mechanisms should be designed to reduce convergence and utilization when GWO is trapped in a local optimum.

6 Conclusion and Possible Research Directions

This paper presented a comprehensive review of feature selection-based GWO algorithm. The feature selection and its approaches were explained. Then, the GWO inspiration and its mathematical model were mentioned. The mechanism of feature selection-based GWO was discussed. The performance of state-of-the-art of feature selection-based GWO was criticized and reviewed. Table 1 illustrated the summary of GWO-based feature selection state-of-the-art methods.

Despite GWO's superior performance in solving feature selection issues, this review suggested some possible directions for research work such as: It is recommended to investigate the use of angle modulated function or other binary operators with GWO to solve feature selection issues. Another possible direction of research is to hybridized filter and wrapper approach and used it with GWO. Moreover, feature

selection is a multi-objective problem aims to reduce both number of feature and classification error rate, and based on the literature, the multi-objective GWO has not fully investigated in such problem. Therefore, it is recommended to propose a new or modify the current multi-objective GWO for feature selection purpose.

References

1. Han J, Pei J, Kamber M (2011) Data mining: concepts and techniques. Elsevier
2. Kotu V, Deshpande B (2014) Predictive analytics and data mining: concepts and practice with rapidminer. Morgan Kaufmann
3. Dash M, Liu H (1997) Feature selection for classification. Intell Data Anal 1(3):131–156
4. Guyon I, Elisseeff A (2003) An introduction to variable and feature selection. J Mach Learn Res 3:1157–1182
5. Xue B, Zhang M, Browne WN, Yao X (2016) A survey on evolutionary computation approaches to feature selection. IEEE Trans Evol Comput 20(4):606–626
6. Binitha S, Sathya SS et al (2012) A survey of bio inspired optimization algorithms. Int J Soft Comput Eng 2(2):137–151
7. Mirjalili S et al (2014) "Grey Wolf Optimizer 1 1", vol 69, pp 46–61
8. Al-Tashi Q, Rais H, Abdulkadir SJ (2018) Hybrid swarm intelligence algorithms with ensemble machine learning for medical diagnosis. In: 2018 4th International Conference on Computer and Information Sciences (ICCOINS), pp 1–6
9. Tang J, Alelyani S, Liu H (2014) Feature selection for classification: a review. Data Classif Algorithms Appl, 37
10. Liu H, Zhao Z (2009) Manipulating data and dimension reduction methods: feature selection. In: Encyclopedia of complexity and systems science. Springer, Heidelberg, pp 5348–5359
11. Liu H, Motoda H, Setiono R, Zhao Z (2010) Feature selection: an ever evolving frontier in data mining. In: Feature selection in data mining, pp 4–13
12. Al-Tashi Q, Kadir SJA, Rais HM, Mirjalili S, Alhussian H (2019) Binary optimization using hybrid Grey Wolf Optimization for feature selection. IEEE Access 7:39496–39508
13. Emary E, Zawbaa HM, Hassanien AE (2016) Binary Grey Wolf Optimization approaches for feature selection. Neurocomputing 172:371–381
14. Sahoo A, Chandra S (2017) Multi-objective Grey Wolf Optimizer for improved cervix lesion classification. Appl Soft Comput J 52:64–80
15. Abd AL, El-Hafeez T, Zaki AM (2018) Face recognition based on Grey Wolf Optimization for feature selection. In: International conference on advanced intelligent systems and informatics, pp 273–283
16. Tu Q, Chen X, Liu X (2019) Multi-strategy ensemble Grey Wolf Optimizer and its application to feature selection. Appl Soft Comput 76:16–30
17. Zhao X et al (2019) Chaos enhanced grey wolf optimization wrapped ELM for diagnosis of paraquat-poisoned patients. Comput Biol Chem 78:481–490
18. Sharma P, Sundaram S, Sharma M, Sharma A, Gupta D (2019) Diagnosis of Parkinson's disease using modified grey wolf optimization. Cogn Syst Res 54:100–115
19. Alzubi QM, Anbar M, Alqattan ZNM, Al-Betar MA, Abdullah R. Intrusion detection system based on a modified binary Grey Wolf Optimisation. Neural Comput Appl, 1–13
20. Sreedharan NPN, Ganesan B, Raveendran R, Sarala P, Dennis B et al (2018) Grey Wolf optimisation-based feature selection and classification for facial emotion recognition. IET Biometrics 7(5):490–499
21. Al-Tashi Q, Rais H, Jadid S (2018) Feature selection method based on Grey Wolf Optimization for coronary artery disease classification. In: International conference of reliable information and communication technology, pp 257–266

22. Too J, Abdullah A, Mohd Saad N, Mohd Ali N, Tee W (2018) A new competitive binary Grey Wolf Optimizer to solve the feature selection problem in EMG signals classification. Computers 7(4):58
23. Li Q et al (2017) An enhanced grey wolf optimization based feature selection wrapped kernel extreme learning machine for medical diagnosis. Comput Math Methods Med
24. Rajalaxmi RR, Kaavya S (2017) Feature selection for identifying Parkinson's disease using binary Grey Wolf Optimization,. In: Proceedings of the International Conference on Intelligent Computing Systems (ICICS 2017–Dec 15th–16th 2017) organized by Sona College of Technology, Salem, Tamilnadu, India
25. Faris H, Aljarah I, Al-Betar MA, Mirjalili S (2018) Grey Wolf Optimizer: a review of recent variants and applications. Neural Comput Appl, 1–23
26. Emary E, Zawbaa HM, Grosan C, Hassenian AE (2015) Feature subset selection approach by Gray-Wolf Optimization. In: Afro-European conference for industrial advancement, pp 1–13
27. Emary E, Yamany W, Ella A, Snasel V (2015) Multi-objective Gray-Wolf Optimization for attribute reduction. Proc Proc Comput Sci 65:623–632
28. Vosooghifard M, Ebrahimpour H (2015) Applying Grey Wolf Optimizer-based decision tree classifer for cancer classification on gene expression data. In: 2015 5th International Conference on Computer and Knowledge Engineering (ICCKE), pp 147–151
29. Yamany W, Emary E, Hassenien AE (2016) New rough set attribute reduction algorithm based on Grey Wolf Optimization. In: The 1st international conference on Advanced Intelligent System and Informatics (AISI2015), 28–30 Nov 2015, Beni Suef, Egypt, pp 241–251
30. Medjahed SA, Saadi TA, Benyettou A, Ouali M (2016) Gray wolf optimizer for hyperspectral band selection. Appl Soft Comput 40:178–186
31. Devi EM, Suganthe RC (2017) Feature selection in intrusion detection grey wolf optimizer. Asian J Res Soc Sci Humanit 7(3):671–682
32. Xue B (2014) Particle swarm optimisation for feature selection in classification, vol 18, pp 261–276