Binary Black Widow Optimization Algorithm for Feature Selection Problems

Ahmed Al-Saedi
alsa0290@mylaurier.ca

Binary Black Widow Optimization Algorithm for Feature Selection Problems

By:

Ahmed Al-Saedi

Master of Applied Computing, Wilfrid Laurier University, 2021

THESIS

Submitted to the Department of Physics and Computer Science

Faculty of Science

in partial fulfillment of the requirements for the

Master of Applied Computing

Wilfrid Laurier University

Abstract

This thesis addresses feature selection (FS) problems, which is a primary stage in data mining. FS is a significant pre-processing stage to enhance the performance of the process with regards to computation cost and accuracy to offer a better comprehension of stored data by removing the unnecessary and irrelevant features from the basic dataset. However, because of the size of the problem, FS is known to be very challenging and has been classified as an NP-hard problem. Traditional methods can only be used to solve small problems. Therefore, metaheuristic algorithms (MAs) are becoming powerful methods for addressing the FS problems. Recently, a new metaheuristic algorithm, known as the Black Widow Optimization (BWO) algorithm, had great results when applied to a range of daunting design problems in the field of engineering, and has not yet been applied to FS problems. In this thesis, we are proposing a modified Binary Black Widow Optimization (BBWO) algorithm to solve FS problems. The FS evaluation method used in this study is the wrapper method, designed to keep a degree of balance between two significant processes: (i) minimize the number of selected features (ii) maintain a high level of accuracy. To achieve this, we have used the k-nearest-neighbor (KNN) machine learning algorithm in the learning stage intending to evaluate the accuracy of the solutions generated by the (BBWO). The proposed method is applied to twenty-eight public datasets provided by UCI. The results are then compared with up-to-date FS algorithms. Our results show that the BBWO works as good as, or even better in some cases, when compared to those FS algorithms. However, the results also show that the BBWO faces the problem of slow convergence due to the use of a population of solutions and the lack of local exploitation. To further improve the exploitation process and enhance the BBWO's performance, we are proposing an improvement to the BBWO algorithm by combining it with a local metaheuristic algorithm based on the hill-climbing algorithm (HCA). This improvement method (IBBWO) is also tested on the twenty-eight datasets provided by UCI and the results are then compared with the basic BBWO and the up-to-date FS algorithms. Results show that the (IBBWO) produces better results in most cases when compared to basic BBWO. The results also show that IBBWO outperforms the most known FS algorithms in many cases.

Acknowledgement

First, I would like to express my deepest appreciation to my supervisor Assoc. Prof. Dr. Abdul-Rahman Mawlood-Yunis, for his valuable guidance, encouragement, and continuous support throughout the writing of my thesis. His extensive knowledge, vision, and creative thinking have been the source of inspiration to me. Your insightful feedback motivated me to elevate my knowledge and brought my work to a higher level of research.

Besides my supervisor, I would like to deliver my appreciation to my committee members, Prof. Dr. Chinh Hoang, Prof. Dr. Ilias S. Kotsireas, and Assoc. Prof. Dr. Xu (Sunny) Wang.

I am very grateful to WLU and all the staff and members of the Department of Computer Science for their help during my study, especially Prof. Dr. Chinh Hoang who has supported and facilitated the whole process whenever required.

To my dearest mother and father, thank you for bringing me up to who I am today. My success symbolizes and reflects the support and love from both of you. My deepest appreciation goes to my wife for her love, who always supports and encourages. Special love goes to my daughters Maryam and Sarah. All thanks to my loving brothers and sisters.

Finally, I thank all my friends at WLU who have always helped and motivated me throughout my study journey.

Contents

List of Abbreviations

ACO	Ant Colony Optimization
ALO	Ant lion optimizer
BA	Bat Algorithm
BBA	Binary Bat Algorithm
BBWO	Binary Black Widow Optimization
BGWO	Binary Grey Wolf Optimizer
BMFO	Binary Moth Flame Optimization
BMVO	Binary Multi Verse Optimizer
BPSO	Binary Particle Swarm Optimization
BWO	Black Widow Optimization
BWOA	Binary Whale Optimization Algorithm
DM	Data Mining
EA	Evolutionary Algorithms
FS	Feature Selection
GA	Genetic Algorithm
GDA	Great Deluge Algorithm
GWO	Grey Wolf Optimizer
HCA	Hill-Climbing Algorithm
IBBWO	An Improved BBWO Algorithm for Feature Selection
KDD	Knowledge Discovery in Database

KNN	K-Nearest Neighbors Algorithm
MAs	Metaheuristic Algorithms
MBA	Mine Blast Algorithm
MFO	Moth Flame Optimization
ML	Machine Learning
MVO	Multi Verse Optimizer
PB	Population-Based Methods
PSO	Particle Swarm Optimization
SA	Simulated Annealing
LS	Local-Search Methods
SVM	Support Vector Machine
TA	Tabu Search
UCI	University of California Irvine
VNS	Variable Neighborhood Search
WOA	Whale Optimization Algorithm

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Problem Statement and Motivations

The data boom in many areas including, business management, pattern recognition, image processing, financial analysis, and medicine, has brought an obligation on researchers to cope with substantial amounts of data, the dimensions of which are expanding daily [1]. The process of extracting patterns and meaningful information out of these large volumes of data that could be used for many important decisions is known as data mining (DM) [2]. The most important category of the methods in the field of DM is classification, which works on the features representing the dataset to make a prediction or to select useful information from such datasets [3]. Classification is the term used to describe the process of allocating each sample to a specific class [4]. However, expansions of the dimensions, that is to say, an increase in the number of features contained in the datasets, have a marked effect on the nature of the results gained [2]. High dimensional datasets present a range of difficulties: these include the more substantial amount of time required to build the learning model, the potential presence of immaterial and extraneous features, and a deterioration in performance caused by the extraneous nature of features that render evaluation or classification of the data [5]. Feature selection (FS) is a thought-provoking challenge in the arena of machine learning (ML) designed to bring down the number of features by eradicating immaterial, extraneous, and noisy data while ensuring that the level of classification accuracy remains satisfactory [6, 7]. FS is primarily used to determine the best subset of instructive features while preserving a high level of classification accuracy in portraying the original dataset features. FS constitutes a pre-processing stage in DM designed to remove redundant and inconsequential features and determine a final set of features that cast the greatest degree of light on the matter, boosting the quality of the data obtained [3, 8].

FS methods consist of two important phases: feature generation and evaluation. During the first phase, a subset of features is selected by a variety of techniques, while in the second phase, the quality of the chosen subset of features that were generated by the search strategy in the generation phase is evaluated [9]. The three key methods to evaluate the selected features are (i) the filter method, (ii) the wrapper method, and the embedded method. The three methods are different in terms of the presence or absence of the learning algorithm when the resultant feature subsets are evaluated. In general, the filter methods rely on statistical data dependency techniques, i.e., the correlations between the conditional features and the class in the absence of a particular learning algorithm (principal component analysis [10], Chi-Square [11]). In contrast, wrapper methods, make use of a learning algorithm during the assessment, while in the embedded methods, both the feature selection algorithm and learning algorithm are integrated with each other to find the best subset. Therefore, wrapper and embedded methods are observed to provide more accurate results than filter methods. Conversely, a greater degree of the computational cost may be necessary for the embedded and wrapper methods, in comparison with the filter methods [5, 12]. However, the wrapper methods are broadly utilized in many fields due to their acceptability in terms of the computational cost and the accuracy [21].

Because of the scale of the problem, and the fact that it seeks out the almost optimal subset, FS is known to pose a serious challenge and has been categorized as an NP-hard problem [13]. It also produces all conceivable solutions to acquire only the best. For example, high computational cost arises if a dataset contains S features, because 2^S solutions must be formulated and assessed [14]. The paramount selected subset of features is sought using classical approaches such as random search, complete search, breadth search, and depth search [3]. However, even though these methods ensure the optimal solution for small datasets, their render is impractical for large datasets because of the enormous amount of computational power required and the excessive amount of time taken up [8].

In the last few years, metaheuristic algorithms (MAs) have been considered to be the ideal and most reliable optimization algorithms for FS problems, particularly in cases involving the challenges presented by high-dimensional problems. MAs have been used extensively to improve real-world problems [5]. A majority of such algorithms have been inspired by diverse spectacles in nature, mathematics, and physics. Researchers employ MAs as FS algorithms because of their potency and the outstanding results that have been attained. Some examples of them can be found in the literature such as Simulated Annealing (SA) [22], Ant Colony Optimization (ACO) [23], Particle Swarm Optimization (PSO) [15], Genetic Algorithm (GA) [16], Whale Optimization Algorithm (WOA) [17], Mine Blast Algorithm (MBA) [6], Ant Lion Optimizer (ALO) [16], Grey Wolf Optimizer (GWO) [7], and Bat Algorithm (BA) [24]. MAs are the most appropriate alternative method of addressing the limitations of a lengthy, far-reaching search that entails high computational cost [18]. But it is worth mentioning that most MAs are impeded by the limitations imposed by a local optimum and a disproportion between the explorative and exploitative scope of the algorithm [8]. Exploration technique diversifies the solutions within the population, so that the search space is explored globally whereas exploitation specializes in the neighbour's area of a current accurate solution. Exploration thru randomization lets in the solutions to avoid being trapped on the local optima and increase the population diversity. But exploitation lets in the searching procedure to converge into an optimal solution. Having the right balance between these two parts leads to an increase in global optimality [4]. Moreover, each dataset has a different number of features and no single method is the most appropriate for the FS, i.e., one can still find room for improvements in the results. These shortcomings spur the researchers to find a means of negating the FS obstacles [4].

1.2 Research Objectives

The major goal of this thesis is to propose efficient FS wrapper methods which are capable of finding good solutions for the FS problems. During this research, we realized that a new metaheuristic algorithm, known as the Black Widow Optimization (BWO) algorithm[19], had great results when applied to a range of daunting design problems in the field of engineering [19], and has not yet been applied to FS problems. For this reason, this study aims to investigate the use of the BWO algorithm when applied to solve FS problems. Therefore, we are proposing a modified Binary Black Widow Optimization (BBWO) algorithm to solve FS problems. To further maximize the BBWO's performance, we are proposing an improvement on the BBWO by combining it with a local metaheuristic algorithm based on the Hill-Climbing Algorithm (HCA). This improvement method (IBBWO) aims to enhance the exploitation process of the BBWO. The FS evaluation method used in these new approaches is the wrapper method, designed to keep a degree of balance between two significant processes: (i) minimize the number of selected features (ii) maintain a high level of accuracy. To achieve this, we have used the k-nearest-neighbor (KNN) ML algorithm in the learning stage intending to evaluate the accuracy of the solutions generated by the BBWO and IBBWO. The main contributions of this study are:

- Two new algorithms (BBWO and IBBWO) to solve FS problems.
- Test results; we tested the performance of our proposed algorithms on twenty-eight benchmark datasets that make use of low, medium, and high dimensional datasets. The obtained results can be used as new benchmarking results.
- New insights about existing FS solutions. Evaluating the performance results of our proposed methods against various up-to-date FS algorithms reveals new insights about the performance of existing algorithms.

1.3 Research Scope

This thesis employed twenty-eight well-known datasets of the University of California Irvine (UCI) [70] which have been used and adopted by many researchers [6, 7, 17, 20] to test the performance of the BBWO and IBBWO FS algorithms. These datasets belong to different domains: medical, physical, business, and electronic. A brief description of the datasets that have been used in this thesis is explained in Chapter 3.

1.4 Thesis Outline

This thesis is divided into six chapters as follows:

Chapter 1 shows the problem statement and motivations, research objective and scope.

Chapter 2 shows the background and concept of FS, moreover, shows a general review of several search metaheuristic algorithms that have been applied to solve the FS problems.

Chapter 3 illustrates a full description of the research methodology. In addition, a detailed description of the benchmark instances is presented in order to evaluate the proposed methods for this problem.

Chapter 4 presents and discusses the experimental results of the proposed method BBWO for FS problems.

Chapter 5 presents and discusses the experimental results of the proposed method IBBWO for FS problems.

Finally, in Chapter 6, conclusions are drawn, and the contributions of this research are set out. In addition, a number of areas to be pursued as future work are suggested.

Chapter 2

Background and Literature Review

The feature selection (FS) method is used to reduce the number of features while maximizing the accuracy of data as much as possible, thereby minimizing the computational complexity while ensuring minimal information loss [4]. FS is an important pre-processing step in the field of data mining (DM) because raw data may encounter many problems in applications [2]. In this chapter, the FS concept is described in Section 2.1. The FS process is then presented in Section 2.2. The metaheuristic algorithms are described in Section 2.3, and a literature review of the metaheuristic algorithms for FS problems is shown in Section 2.4.

2.1 Feature Selection

DM or machine learning (ML) techniques have attracted considerable attention from both the academe and industry because of their significant contributions to intelligent data analysis. DM and its applications are expected to become even more crucial in the future because real-world applications are growing rapidly as organizations continuously gather increasingly larger amounts and more diverse types of data [25, 26, 27]. Therefore, extracting useful and meaningful knowledge from databases (KDD) is becoming a core process in databases in many research areas such as medical, business, transportation, data visualization, statistics, optimization, ML, and pattern recognition [25, 26, 27]. Knowledge discovery in databases is defined as "*the nontrivial* *process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data."* [25, 27].

The knowledge discovery in the database process is divided into five steps (Figure 2.1): (a) Data selection, in which the dataset is chosen; (b) Data cleaning/pre-processing, which involves noise removal or reduction and the imputation of missing values; (c) FS or data reduction, which aims to obtain the most informative features from the dataset by deleting irrelevant and redundant features that may mislead the process and may not help the KDD; (d) DM, which involves selecting hidden predictive information from many databases depending on the goal of knowledge discovery; (e) Evaluation, which is performed to ensure the simplicity, novelty, usefulness, and validity of the discovered knowledge. This process may require some of the formal steps to be repeated [25, 27].

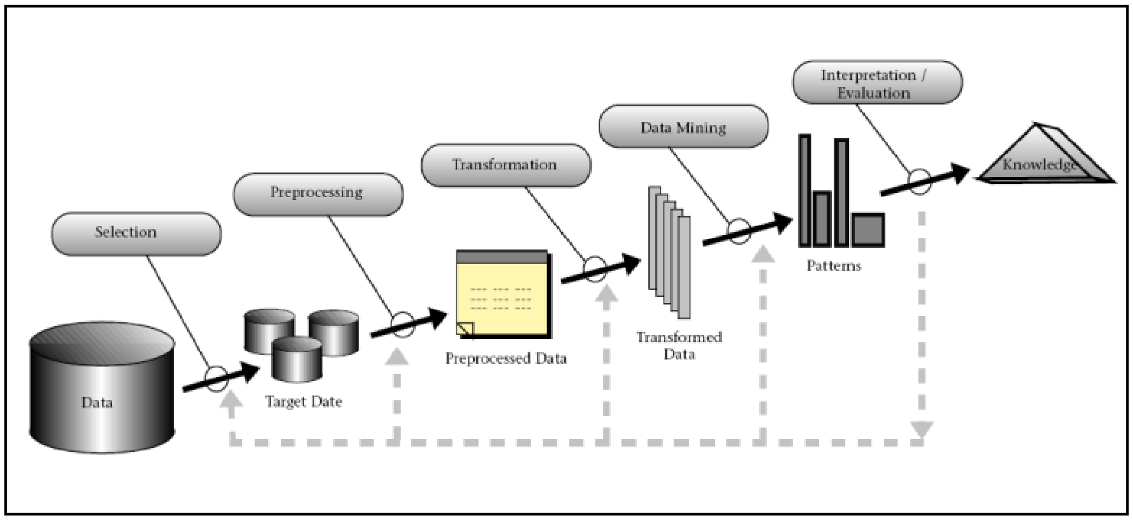


Figure 2.1 Knowledge discovery process steps [25]

FS is the third step in the knowledge discovery process and is the primary focus of this study. FS is a significant pre-processing step in the DM process and ML [28], and it could be defined as the process of extracting a minimum redact (subset) from the original set [29]. FS is also defined as “a process that chooses an optimal subset of features according to certain criterion” [2, 30].

The aim of FS is to eliminate irrelevant and redundant features from high-dimensional datasets, thereby increasing the likelihood that a DM algorithm will find generally invalid and spurious patterns [31]. According to [6, 31, 33], FS aims to (a) improve performance (speed of learning, predictive accuracy, or simplicity of rules); (b) visualize the data for model selection; and (c) reduce dimensionality and remove noise.

2.2 Feature Selection Process

The FS process aims to search for the minimum features that meet a certain criterion to build a prediction model that achieves the highest accuracy [34]. The FS process consists of four basic processes (Figure 2.2) [35], which are described below.

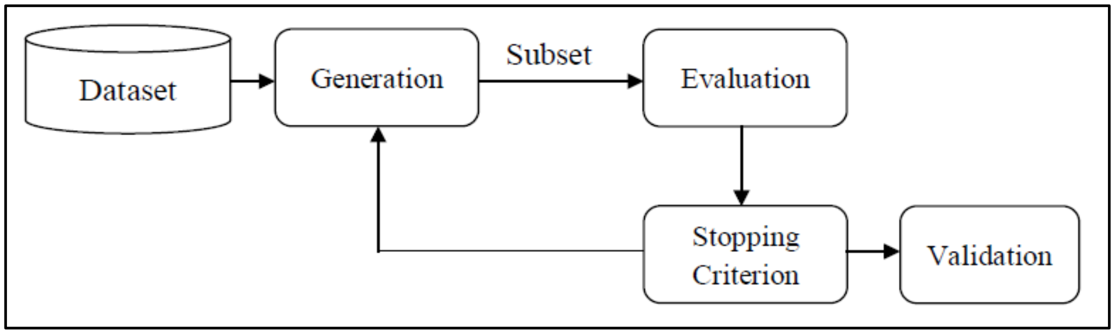


Figure 2.2 FS process [35]

2.2.1 Feature Subset Generation

Feature subset generation is the process of searching for the best subset of features. This process considers FS an NP-hard problem [36]. Theoretically speaking, a FS method must search all possible combinations to find optimal or near-optimal features. Simply put, FS must search 2^F features so it grows exponentially when F increases, where 2 is the binary representation of features (0: none are selected, 1: selected), and F is the total number of features [36].

At this stage, a partial set of features is searched using any one of the complete, random, or heuristic search methods [37]. Subset creation may begin with an empty set without variables, which are then added one by one until the error is decreased (forward), with a full set that contains all variables, which are removed one by one until highest accuracy is reached (backward), or with random set to achieve the highest accuracy [37, 38]. The three search techniques are described briefly below.

- Complete search: This method searches all possible combinations of features 2^Funtil the optimal subset that achieves the highest accuracy is obtained. This method guarantees the optimal solution. However, it needs a large amount of computational power and a long searching process, thereby making it infeasible for large datasets [20, 38].
- Random search: In this type of search method, features are searched randomly; the process is also called nondeterministic search [38]. The random nature enables the solution (optimal features) to be found during the early stages of the search process; this solution represents the best case. However, this method may need to visit the whole features in a similar way as complete search, which represents the worst case [18, 38].
- Heuristic search: This technique is used frequently when traditional methods take too much time to find the solution or to find a near-optimal solution by approximation when traditional methods are unable to do so. Heuristic methods use the minimum information available to execute an effective search without requiring all feature combinations. Two

types of heuristic methods exist: general-purpose metaheuristics, which solves a wide range of problems, and specified heuristics, which solves specific types of problems [38, 39]. Metaheuristic algorithms are more effective in solving optimization problems than other approaches such as complete search [39, 40].

2.2.2 Feature Selection Evaluation Methods

The evaluation phase is the next step after feature subset generation. FS evaluation methods are categorized into filter, wrapper, and embedded, as shown in Figure 2.3 [41]. In this phase, evaluation methods are used to measure the goodness of the generated subset. The evaluation result is then compared with previous results, and the best result is retained throughout the iteration. Different results are generated when different evaluation methods are used in the same dataset [35].

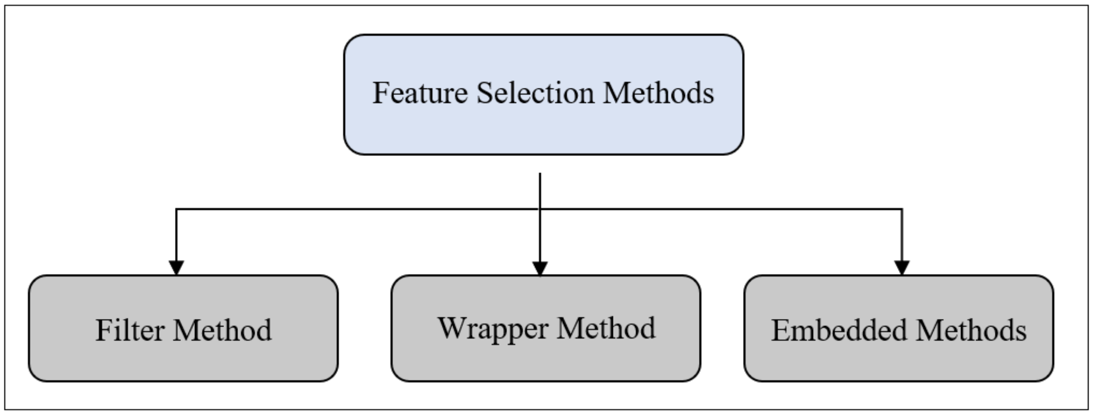


Figure 2.3 Feature selection methods [41]

The three FS methods have differences in terms of how the ML algorithm is used when the resulting feature sets are being evaluated [3, 18]. The ML algorithm is a subfield of artificial intelligence, and it gives computers the ability to learn without being explicitly programmed [35, 48]. Most ML algorithms are classified into supervised learning (examples are k-nearest neighbors (KNN) [46] and support vector machine (SVM) [47]) and unsupervised learning (an example is k-means clustering) [48]. Classifiers are used in supervised learning, and they are obtained from learning the labeled data and constructing a model based on that learning. Unsupervised learning is the opposite; clustering is used, which comes from learning unlabeled data and constructing a model based on that learning [48]. The filter, wrapper, and embedded FS evaluation methods are presented in the following subsections.

2.2.2.1 Filter Method

The filter method concentrates on selecting features based on performance and not on building algorithms. The modeling algorithm can then use the best features after they are selected. Not all filter methods can be applied to different ML problems. Different types of filter methods are used to address different types of problems, such as classification, clustering, or regression [41]. Univariate and multivariate feature filters have certain differences; univariate filters rate one feature, whereas multivariate filters rate a whole feature subset. Moreover, univariate feature filters are independent, scalable, and fast, yet they ignore learning algorithms and the relationship between features. In contrast, multivariate feature filters are independent and utilize feature relationships but are slow and have poor scalability. The filter method uses statistical methods such as distance in the evaluation process to measure the goodness of the features to be selected. Examples of this type of selection method are principal component analysis [10] and chi-square [11], which have high computational efficiency because of non-iterative computation on the dataset but are less accurate than other FS evaluation methods because the learning algorithm is not required [42]. Figure 2.4 illustrates the process of the filter method.

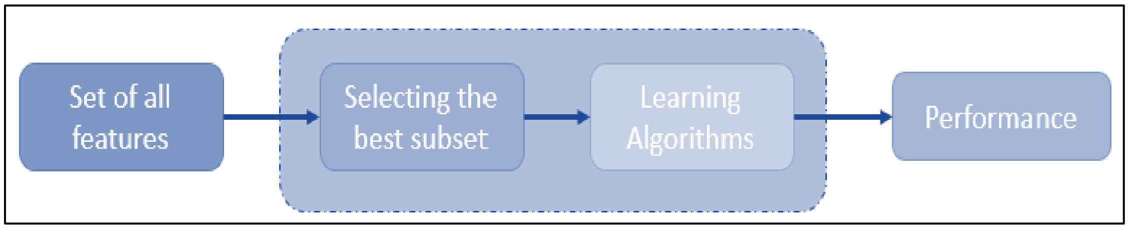


Figure 2.4 Filter method process [42]

2.2.2.2 Wrapper Method

Wrapper methods are different from filter methods because they focus on the quality of the modeling algorithm, using classifier accuracy to measure the performance of the selected subset as shown in Figure 2.5. The goal of this method is to maximize predictive accuracy by reducing the error rate. The wrapper method is an iterative process that generates subsets, where the accuracy of each subset is calculated. At the end of the process, the most accurate subset that was generated in the previous phase will be used in the learning algorithm over the training data. The result will then be compared with a testing dataset to measure the accuracy [38, 43].

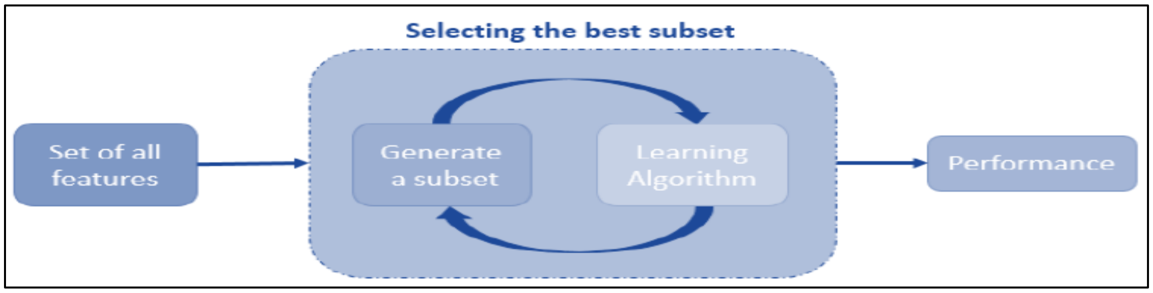


Figure 2.5 Wrapper method process [43]

2.2.2.3 Embedded Method

In the embedded method (Figure 2.6), the FS and learning algorithms are integrated to find the best subset. Therefore, the classifier becomes dependent on selections that might not work with any other classifier because the optimal set of genes is built when the classifier is constructed, and the selection is affected by the hypotheses made by the classifier [44]. This method is computationally demanding. One example of the embedded technique is the decision tree algorithm [45].

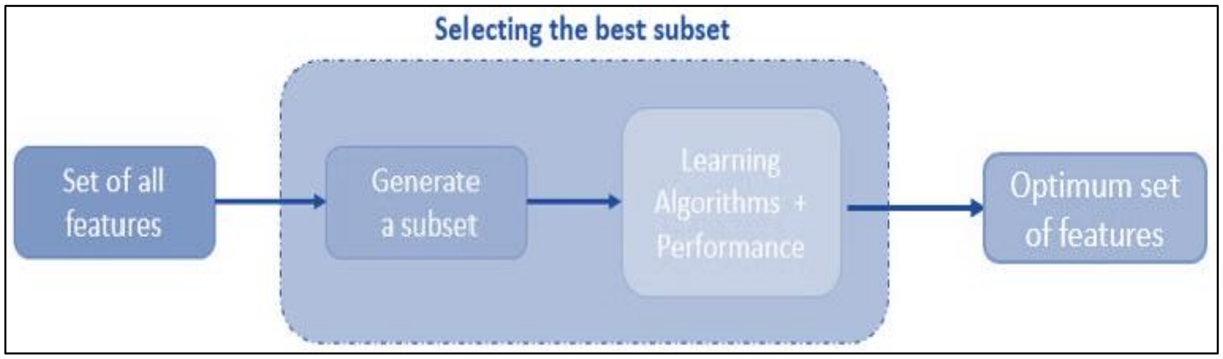


Figure 2.6 Embedded method process [44]

The wrapper method achieves higher accuracy than the other methods because the latter adjusts to specific interactions between the dataset and the classifier. Even though the wrapper method is slow because it requires a classifier to be trained for each feature, it is widely used in many fields because of its satisfactoriness, which is why wrapper method needs to be developed to solve FS problems. The wrapper method was selected in this thesis.

2.2.3 Stopping Criterion

Establishing the stopping criterion is a necessary step to prevent the algorithm from entering an infinite loop, which may reduce computer resources, especially the main memory, and cause the algorithm to crash. The algorithm stops searching when a certain condition is satisfied; at this point, the stopping criterion immediately terminates the search process. The stopping criterion is typically determined based on a combination of search strategies and function evaluation [38].

The stopping criteria can be reaching the loop limit of the algorithm (maximum iteration), the end of the search execution time, the use of time instead of iterations, the end of the complete search, or achieving an acceptable degree of feature subset quality (accuracy).

2.2.4 Validation

This phase is not considered part of the FS process. However, it is used to validate the accuracy of features. Validation is an iterative procedure that involves creating a classifier from the training data and validating its accuracy by using testing data until the highest accuracy is achieved. Validation takes place usually after the FS process is finished. The chosen solution is then validated through different tests, and the results are compared with those of other FS methods [35, 38].

2.3 Metaheuristic Algorithms

Optimization methods are found in different fields, including engineering, computing, and even everyday life, where maximization or minimization are applied to solve problems. For example, companies use optimization methods to maximize their sales while minimizing losses [49]. Finding the optimal solution or complete search is an expensive, time-consuming process. Accepting a relatively reasonable and not optimal solution is therefore a reasonable solution; metaheuristic algorithms can be used for this purpose [39]. The word “metaheuristic” is originally a Greek word that consists of two parts: “meta” means “upper-level methodology”, and “heuristic” means “exploring new ways (strategies) to solve problems” [39, 49].

A metaheuristic algorithm is an upper-level general methodology (template) that can be used as a guiding strategy in designing underlying heuristics to solve specific optimization problems

[39]. Metaheuristic algorithms are also defined as a group of techniques that guide the search process [50], with the primary aim of exploring the search space to find the best solutions [50].

Most metaheuristic algorithms are inspired by nature and could be classified into three groups: physical based, swarm based, and evolutionary based [19]. The basic inspiration of physical-based algorithms is physics rules, such as electromagnetic force, inertia force, and gravitational force. Considering these rules, the search agents of the algorithms communicate and move through the search space [19]; one example is Simulated Annealing (SA) [53]. Swarm-based algorithms are inspired by the collective manner of social beings, which refers to the interaction method among the members of a swarm and their environment [19]; examples are Particle Swarm Optimization, (PSO) [15], Gray Wolf Optimizer (GWO) [7], and Whale Optimization Algorithm (WOA) [17]. Evolutionary-based algorithms, an example of which is Genetic Algorithm (GA) [51], are mostly inspired by nature and biological evolution, such as selection, reproduction, combination, and mutation. These algorithms are derived from the natural selection theory of Darwin, which involves descent with modification, the idea of changing species over time, and generation of new ones. In the natural selection process, the main heritable traits are passed on, enabling species to survive and reproduce [19].

Depending on the researchers [39, 56], there are two types of metaheuristic search algorithms as shown in the Figure 2.7; the Local-Search methods (LS) one improves one solution throughout an iteration, whereas the Population-Based methods (PB) one improves a set of solutions throughout iterations find the near-optimal solution [39, 56, 75]. These algorithms are described briefly in Subsections 2.3.1 and 2.3.2, respectively.

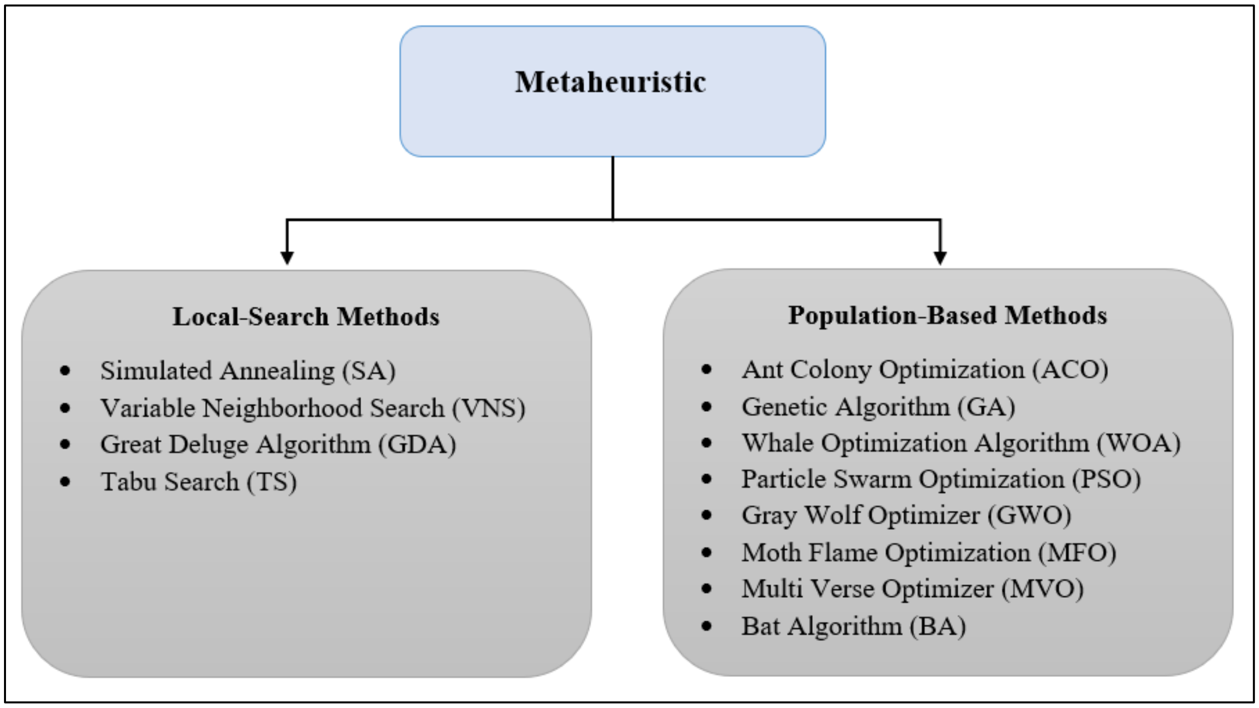

Metaheuristic
Local-Search Methods
Simulated Annealing (SA)
Variable Neighborhood Search (VNS)
Great Deluge Algorithm (GDA)
Tabu Search (TS)
Population-Based Methods
Ant Colony Optimization (ACO)
Genetic Algorithm (GA)
Whale Optimization Algorithm (WOA)
Particle Swarm Optimization (PSO)
Gray Wolf Optimizer (GWO)
Moth Flame Optimization (MFO)
Multi Verse Optimizer (MVO)
Bat Algorithm (BA)

Figure 2.7 Examples of metaheuristics search algorithms

2.3.1 Local-Search Methods

LS methods, such as SA [53], Variable Neighborhood Search (VNS) [52], Great Deluge Algorithm (GDA) [73], and Tabu Search (TS) [74], take a single solution and administer new and replicative processes, where a group of possible solutions are derived from the solution, during the generation process [39, 75]. The possible solutions often emerge through the solution's local evolution. During the replicative stage, the possible solutions are searched to find a new solution until the stopping point is reached [39]. Figure 2.8 illustrates the basic template of LS metaheuristic algorithms.

```
“Input: Initial solution s0.
t = 0;
Repeat
/* Generate candidate solutions (partial or complete neighborhood) from $s_t$*/
Generate(C ($s_t$) ;
/* Select a solution from C(s) to replace the current solution $s_t$*/
$s_{t+1}$= Select(C ($s_t$ )); t = t + 1;
Until Stopping criteria satisfied
Output: Best solution found.”
```

Figure 2.8 A basic template of LS metaheuristic algorithms [39]

2.3.2 Population-Based Methods

PB methods, such as GA [51, 75], begin with a group of solutions that then form another group of solutions as an improvement of that group of solutions; this process continues until the stopping criterion is met [39]. Figure 2.9 illustrates the basic template of PB metaheuristic algorithms.

```
“P =$P_0$; /∗ Generation of the initial population ∗/
t = 0 ;
Repeat
Generate ($\grave{P}_t$); /∗ Generation a new population ∗/
$P_{t+1}$= Select-Population ($P_t \cup \grave{P}_t$ ); /∗ Select new population ∗/
t = t + 1;
Until Stopping criteria satisfied
Output: Best solution(s) found.”
```

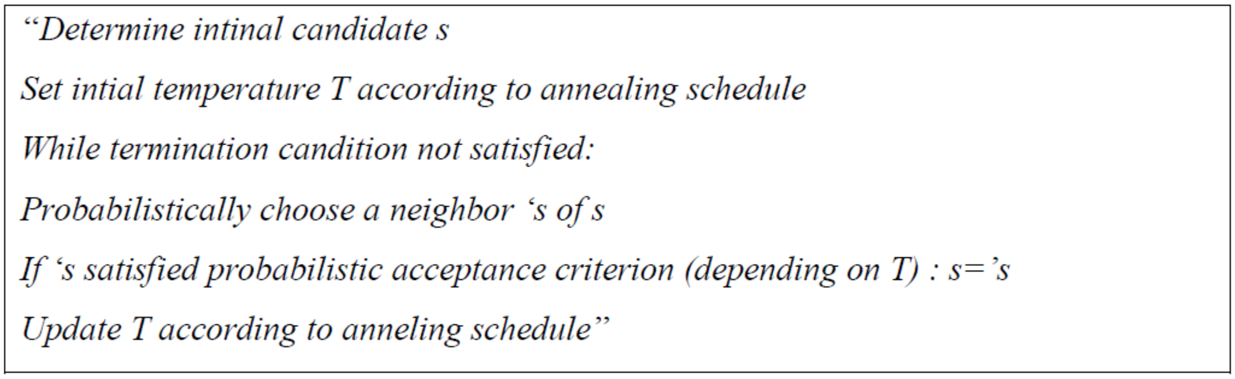
Figure 2.9 A basic template of PB metaheuristic algorithm [39]

2.4 A Short Review of Metaheuristic Algorithms for Feature Selection

FS is a dimensionality reduction technique that is used to remove redundant features from datasets. It is an NP-hard problem related to the search for the most informative features. This problem cannot be addressed by using traditional algorithms. Metaheuristic algorithms can be used by approximation. The latest important metaheuristic algorithms that have been applied to address FS problems are presented in the succeeding sections.

2.4.1 Simulated Annealing

SA is a LS metaheuristic algorithm based on an algorithm (known as hill climbing algorithm) [53]. The algorithm, resulting from the annealing process is used to address the combinatorial optimization problems, where a solid matter is heated to a higher temperature and then slowly cooled to be crystallized. This process of cooling down is controlled by the three parameters namely, final temperature, initial temperature, and cooling schedule. It was successfully engaged in a variety of optimization problems as vehicle routing problems [54], and timetabling problems [55]. During each application of a simulated annealing algorithm to a discrete optimization issue, a comparison is made between two solution values (the current and randomly chosen solution). Better solutions are always accepted, though a small number of non-improving solutions are also accepted with the goal of evading local optima in the pursuit of global optima. The likelihood of the reception of non-improving solutions is dependent on the temperature parameter, which usually remains constant through each iteration of the algorithm [56]. The SA procedure has been proposed for solving feature selection problems [22], and when it tested on some UCI datasets, the SA performed well in terms of the tested features reduction problems in that time. Figure 2.10 shows the basic pseudo-code of SA algorithm.



```
"Determine intinal candidate s
Set intial temperature T according to annealing schedule
While termination candition not satisfied:
Probabilistically choose a neighbor 's of s
If 's satisfied probabilistic acceptance criterion (depending on T) : s='s
Update T according to anneling schedule"
```

Figure 2.10 Pseudo-code of SA [59]

2.4.2 Variable Neighborhood Search

VNS is an LS metaheuristic algorithm that is used to resolve combinatorial issues [52]. The essential notion behind VNS is twofold: a methodical neighborhood changes during a suitable phase, and a disorder phase is used to escape the analogous valley [56]. The original aim of VNS was to find estimated solutions for combinatorial optimization issues. It has since been expanded to include nonlinear programs, mixed-integer programs, and mixed-integer nonlinear programs. VNS is also utilized for automatic or computer-assisted graph theory [52]. Figure 2.11 shows the basic pseudo-code of VNS.

A composite neighborhood structure based on VNS to solve FS problems suggested by [57]. This method consists of two parts. In the first part, a basic composite neighborhood structure approach is used to randomly select a neighborhood, and then the current solution is used to derive the new solution, which will be accepted if it supersedes the original solution. The second part is divided into two stages. The first stage randomly generates an initial solution, and then the dependency degree of the solution is calculated. While it is on loop, a neighborhood structure is randomly selected from two lists by using an intelligent selection mechanism. A new and improved solution is then generated until the degree of dependency is equal to 1. The second stage utilizes the superior solution discovered in the first stage as a starting point. Similar to the process in the first stage, a neighborhood structure is selected at random during the loop in accordance with certain rules to derive a new solution. The algorithm will continue to accept the superior solution. If both solutions have the same qualities, then the algorithm will select the solution with the fewest features. This method has been applied on 13 UCI datasets and has produced satisfactory outcomes in a few datasets.

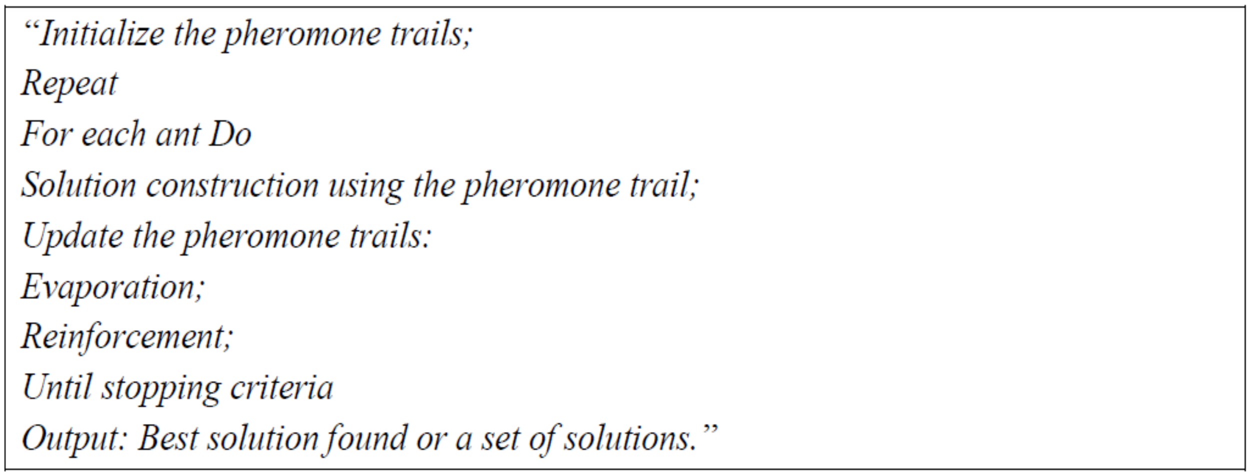
```
"Select the set of neighbourhood structures $N_k$ for $k = 1, . . ., k_{max}$.
$x = x_0$; // Generate the initial solution
$k = 1$;
While $k \le k_{max}$ Do
    Find the best neighbor $x'$ of $x$ in $N_k(x)$;
    If $f(x') < f(x)$ Then $x = x'$; $k = 1$;
    Otherwise $k = k + 1$;
Output: Best found solution"
```

Figure 2.11 Pseudo-code of VNS [52]

2.4.3 Ant Colony Optimization

Ant Colony Optimization (ACO) is a PB metaheuristic algorithm, it simulates the behavior of real ants when searching for the shortest path to a food source, which deposits pheromone as they travel. Each ant prefers to follow the path that is rich with pheromone [23]. ACO mimics this behavior by applying a simple communication mechanism of ant to find the shortest path between two points. As shown in Figure 2.12 the pseudo-code of ACO. Every repetition indicates a cycle of generated initial solutions. ACO iterates for a certain number of iterations. At every iteration, the pheromone value is updated, and the daemon action takes place if it is activated. and when a termination factor is reached the algorithm stops [39].

Because ACO aims to find the shortest path (minimum), it has been shown to be a useful method of fixing feature-selection, and it has been applied for feature selection problems [58], the procedure begins with the generation the same number of ants as equally the features in the dataset. Following this, the ants are randomly distributed across the graph. Each ant begins with a unique attribute and then performs a motion along the path. This is guided by measurement of probability and movements cease when the stopping requirement is achieved (i.e. with the uncovering of the best solution), the procedure will repeat again under a new pheromone, with a new population. This method had been applied on 13 UCI datasets and has been able to yield positive outcomes in a few datasets.



```
"Initialize the pheromone trails;
Repeat
For each ant Do
Solution construction using the pheromone trail;
Update the pheromone trails:
Evaporation;
Reinforcement;
Until stopping criteria
Output: Best solution found or a set of solutions."
```

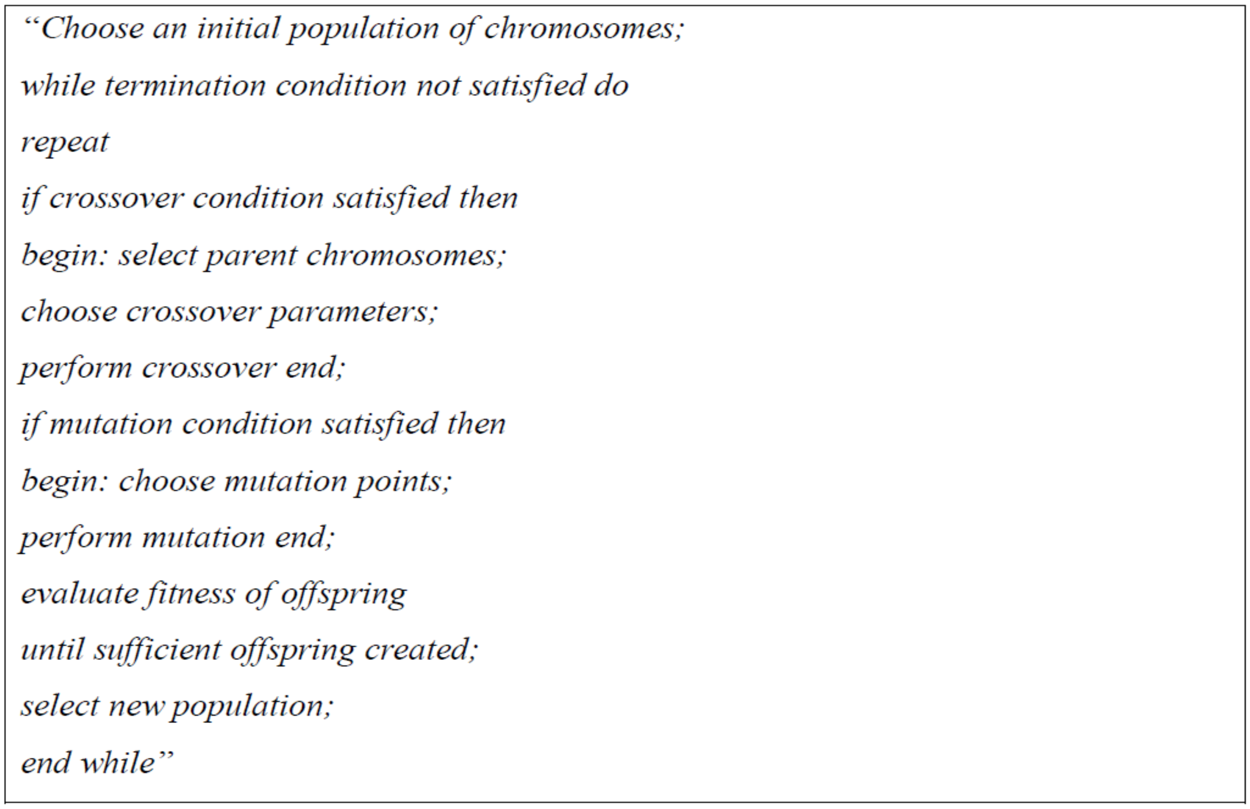
Figure 2.12 Pseudo-code of ACO [39]

2.4.4 Genetic Algorithm

GA is a PB metaheuristic algorithm that was designed to mirror biology, such as in animals and vegetation (i.e., the preferential breeding process, which involves the selection of the best genes for procreation) [51]. As explained in [39, 56], chromosomes, as they are commonly called in research on GA, are a group of strings. GA is also said to have the capacity to innovate a wide range of existing solutions to problems and that it must evolve through the use of search or disparity operators. GA can be divided into crossover operators (i.e., two parents producing at least one descendant) and mutation operators (i.e., changing the existing structure to a new structure). The GA uses concepts of the evolution process, which are analogous to evolutionary mechanisms in nature, such as selection, crossover, and mutation. However, GA is a stochastic algorithm and needs to be fine-tuned to achieve the best results [39]. Figure 2.13 shows the basic pseudo-code of the GA algorithm.

GA for FS problems has been applied using mathematical tools called the rough set theory [22], these tools have been employed in this method to calculate the dependence of degree to evaluate the quality of GA solution. The experiment results have shown that this GA did not manage to produce good results compared with other methods.

Another method has been proposed to enhance the performance of GA in solving FS problems [60]. This method exploits a competition strategy, in which the new selection and crossover systems are merged, and the expected result is an improved global search capability. To improve the quality of the search performed by the algorithm during the mutation process, a high mutation rate is suggested, which divides the chromosomes into groups that represent winners and losers. The next parents in the sequence are selected as a result of competitive selection for a crossover operation. Afterwards, the mutation takes place by means of a dynamic mutation operator. The study in question stated that the proposed technique could perform more rapidly and surpass other orthodox methods. However, the structure of the algorithm is highly complex [9].



```
"Choose an initial population of chromosomes;
while termination condition not satisfied do
repeat
if crossover condition satisfied then
begin: select parent chromosomes;
choose crossover parameters;
perform crossover end;
if mutation condition satisfied then
begin: choose mutation points;
perform mutation end;
evaluate fitness of offspring
until sufficient offspring created;
select new population;
end while"
```

Figure 2.13 Pseudo-code of GA [39]

2.4.5 Whale optimization algorithm

WOA [61] is a PB metaheuristic algorithm whose principal concept was inspired by the hunting method of humpback whales, which swim toward their prey in a spiral pattern and then form bubbles to restrain it [17, 20]. Similar to other metaheuristic algorithms, WOA consists of exploitation and exploration phases; the former mimics the act of encircling a prey and then attacking it with a spiral bubble net, and the latter reflects the random search for a prey [17, 20].

- The exploitation phase is simulated based on the whale's mechanism of encircling the prey (represented by the best solution found so far) and moving toward it, as modeled in Equations (2.1) and (2.2) [20]:

$$D = |C.X^* (t) - X(t)| \tag{2.1}$$

$$X(t+1) = X^* (t) - A.D \tag{2.2}$$

where *t* presents the current iteration; *X** and *X* indicate the best and the current whales (solutions), respectively; and A and C are coefficient vectors calculated as in Equations (2.3) and (2.4) [17, 20].

$$A = 2\, a.r - a \tag{2.3}$$

$$C = 2.r \tag{2.4}$$

where a decreases linearly from 2 to 0 through the iterations (to simulate the shrinking–encircling behavior as in Equation [2.5]) and r is a random vector in [0,1].

$$a = 2 - t\frac{2}{MaxIter} \tag{2.5}$$

where t is the iteration number, and $MaxIter$ is the maximum number of allowed iterations. The spiral-shaped path is obtained by calculating the distance between the solution (*X*) and the leading solution (X*). A spiral equation is then created between the current solution and the best (leading) solution as in Equation (2.6).

$$X(t+1) = D'.e^{bl}.\cos(2\pi l) + X^* (t) \tag{2.6}$$

where D' represents the distance between the *i*th search agent and D', b is a constant, and l is a random number in the interval [−1,1]. A 50% probability is used to select between the shrinking–encircling behavior and the spiral-shaped path as follows:

$$X(t+1) = \begin{cases} Shrinking\ Encircling\ using\ Equation\ (2), & p < 0.5 \\ Spiral\ Shaped\ Path\ using\ Equation\ (6), & p \geq 0.5 \end{cases} \tag{2.7}$$

where p is a random number in [0,1].

- In the exploration phase in WOA, a search agent is selected randomly from the population to update the positions of the current whales instead of updating their positions according to the location of the best solution so far; this approach can prevent the solutions from being trapped in the local optima [17, 20]. This process is modeled as in Equations (2.8) and (2.9).

$$D = |C.X_{rand} - X| \tag{2.8}$$

$$X(t+1) = X_{rand} - A.D \tag{2.9}$$

where the X_{rand} is a randomly selected search agent from the current population, A is a vector with random values less than -1 or greater than 1. The pseudo-code of WOA is shown in Figure 2.14. First, a random population of solutions is generated, then the fitness value for each solution is calculated by using an objective function. The best solution is determined, and the coefficients are updated in the next step. In the following phase, the solutions in the population are updated using Equation (2.2) or (2.6) depending on a random value (p). This process is repeated until a stopping constraint is met. Lastly, the algorithm returns the best solution [17, 20].

WOA has been exploited by numerous researchers in the arena of FS. First, a binary version of WOA was proposed for wrapper FS problems [17]. This method consists of two approaches: tournament roulette and evolutionary operators such as crossover, and it was tested on standard UCI benchmark datasets and compared with three algorithms, namely, PSO, GA, and ant lion optimizer; WOA performed better than the three algorithms. A hybrid between WOA and SA [62] was proposed for wrapper FS problems to achieve a balance between exploration and exploitation.

WOA was run first and was followed by SA to improve the exploitation, which aims to obtain better solutions. The results of this combination were better than the basic WOA.

```
Initialize the parameters
Generate a random population
Evaluate all solutions in the population
X* = the best solution
while (t < Max-Iteration) do
    for each solution do
        Update a, A, C, l, and p
        if (p < 0.5) then
            if (| A |< 1) then
            Update the position of the current solution by Eq (2.2)
            else if (| A |> 1) then
                Select a random solution from a population
                Update the position of X(t) by Eq (2.9)
            end if
        else if (p > 0.5) then
            Update the position of X(t) by Eq (2.6)
            end if
    end for
    Check if any solution goes beyond the boundaries of the search space.
    Evaluate all solutions in the populations
    Update X* if there is a better solution
    t = t + 1
end while
return X*
```

Figure 2.14 Pseudo-code of WOA [20]

2.4.6 Particle Swarm Optimization

PSO [15] is a PB metaheuristic algorithm inspired by the social behavior of bird flocking [63]. It uses a number of particles (candidate solutions) that fly within a search space to find the best solution, tracing the best location (best solution) in their paths. In other words, particles consider their own best solutions and the best solution obtained by the swarm thus far. Each particle in PSO needs to consider the current position; the current velocity; the distance to their personal best solution, $pbest$; and the distance to the global best solution, $gbest$, to modify its position [63]. PSO is mathematically modeled as follows:

$$v_i^{t+1} = wv_i^t + c_1 \times rand \times (pbest_i - x_i^t) + c_2 \times rand \times (gbest - x_i^t) \tag{2.10}$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \tag{2.11}$$

where v_i^t is the velocity of particle i at iteration t, w is a weighting function, c is an acceleration coefficient, rand is a random number between [0,1], x_i^t is the current position of particle i at iteration t, $pbest_i$ is the best solution that the *i-th* particle has obtained so far, and $gbest$ indicates the best solution the swarm has obtained so far. The first part of Equation (1), wv_i^t , provides exploration ability for PSO [63]. The second and third parts, $c_1 \times rand \times (pbest_i - x_i^t)$ and $c_2 \times rand \times (gbest - x_i^t)$, represent private thinking and collaboration of particles, respectively [63]. PSO begins by placing the particles randomly in a problem space. The velocities of particles are calculated using Equation (2.10) at each iteration. The position of particles can be calculated by using Equation (2.11) after the velocities have been defined. The process of changing particles' positions will continue until an end criterion is met [63].

A binary version of the PSO algorithm was presented for FS problems [63]. The effect of using five different updating strategies for w was investigated for evaluation purposes by using 12 UCI benchmark datasets. PSO obtained better results than other similar methods by means of average classification accuracy and average selection size. According to an extensive analysis of the results, updating strategies that gradually decrease the inertia weight parameter linearly and nonlinearly could improve the exploration and exploitation behaviors of PSO for FS tasks.

2.4.7 Gray Wolf Optimizer

GWO [64] is a PB metaheuristic algorithm that simulates the behaviors of gray wolves in hunting, searching, and encircling their prey. In accordance with the social hierarchy of the wolves' community, four types of wolves have different levels of dominance and leadership: alpha (α), beta (β), delta (δ), and omega (ω) [7]. In the GWO algorithm, the alpha, beta, and delta represent the first, second, and third best solutions, respectively. The remaining candidate solutions are considered as omega. While hunting, wolves encircle their prey in a manner that can be mathematically modeled as follows [7]:

$$\vec{D} = \left|\vec{C}.\vec{X}_p - \vec{X}(t)\right| \tag{2.12}$$

$$\vec{X}(t+1) = \left|\vec{X}_p(t) - \vec{A}.\vec{D}\right| \tag{2.13}$$

where $\vec{X}_p$ and $\vec{X}$ represent the positions of the prey and gray wolf, respectively, at an iteration (*t*). $\vec{A}$ and $\vec{C}$ are coefficient vectors that can be formulated as follows:

$$\vec{A} = \left|2\,\vec{a}\,.\overrightarrow{rand1} - \vec{a}\right| \tag{2.14}$$

$$\vec{C} = 2\,.\overrightarrow{rand2} \tag{2.15}$$

where $\overrightarrow{rand1}$ and $.\overrightarrow{rand2}$ are random vectors in [0,1]. $\vec{a}$ linearly decreases from 2 to 0 over iterations as follows:

$$a = 2 - t \times \frac{2}{iterations} \tag{2.16}$$

The parameter *iterations* determine the maximum number of iterations. The position of the best gray wolf is updated by adjusting the vectors $\vec{A}$ and $\vec{C}$. To mimic the hunting behavior, the alpha, beta, and delta are aware of the potential locations of the prey. Alpha, beta, and delta indicate the best three solutions, and the other wolves update their positions according to the best three solutions ($\vec{X}_1$, $\vec{X}_2$, *and* $\vec{X}_3$). This approach can be expressed as follows [7]:

$$\vec{X}(t+1) = (\vec{X}_1 + \vec{X}_2 + \vec{X}_3)/\,3 \tag{2.17}$$

$$\vec{X}_1 = \vec{X}_\propto - \vec{A}_1\,.\,(\vec{D}_\propto), (\vec{D}_\propto) = \left|\vec{C}\,.\,\vec{X}_\propto - \vec{X}\right| \tag{2.18}$$

$$\vec{X}_2 = \vec{X}_\beta - \vec{A}_2\,.\,(\vec{D}_\beta), (\vec{D}_\beta) = \left|\vec{C}_2\,.\,\vec{X}_\beta - \vec{X}\right| \tag{2.19}$$

$$\vec{X}_3 = \vec{X}_\delta - \vec{A}_3\,.\,(\vec{D}_\delta), (\vec{D}_\delta) = \left|\vec{C}_3\,.\,\vec{X}_\delta - \vec{X}\right| \tag{2.20}$$

When attacking the prey, each wolf updates its position between its current position and the position of the prey so that $|A| < 1$. In searching for the prey, the alpha, beta, and delta wolves move away from each other to search for the prey and then converge when they attack the prey. $\vec{A}_1$ can take random values less than -1 or greater than 1 to force the wolves to diverge from the prey [7]. Figure 2.15 shown the pseudo-code of GWO.

GWO has been applied for the selection of the optimal feature subset for classification purposes through a proposed binary version of the GWO for FS in the wrapper method [12]. The results of this method are compared against those of PSO and GA. The binary version of GWO has better search capability than PSO and GA.

Another work [7] proposed an improved GWO for solving wrapper FS problems. The algorithm is incorporated with a two-phase mutation operator, which was able to enhance the capability and efficacy of the algorithm. The first mutation phase is used to reduce the number of selected features considering the classification accuracy, and the second phase involved adding more features that increase the classification accuracy. The KNN classifier was used for training. The proposed method achieved good classification accuracy; however, it spent a long time during the evaluation process [7].

```
Initialize a population of n grey wolves $X_i, i = 1, \dots, n$
Initialize the parameters a, A, C
Calculate the fitness value of each grey wolf
Assign the best three grey wolves to $X_\alpha, X_\beta$, and $X_\delta$ respectively
Define t= 0
While (t < iterations)
  For each grey wolf
    Update the position of the current grey wolf using eq. (2.17)
  End for
  Update a, A, and C
  Calculate the fitness of all grey wolves
  Update the first three grey wolves $X_\alpha, X_\beta$, and $X_\delta$
  t++
End While
Return the best grey wolf $X_\alpha$
```

Figure 2.15 Pseudo-code of GWO [7]

2.4.8 Moth Flame Optimization

Moth Flame Optimization (MFO) [65] is a PB metaheuristic algorithm [66], it is mainly inspired by the concept of transverse orientation, a navigation method used by moths in nature. Moths fly at night by maintaining a fixed angle with respect to the moon, which is an effective way to travel in a straight line over long distances [65]. However, these insects become trapped in a deadly spiral path when they are in the presence of artificial lights. To perform optimization, this algorithm mathematically models this behavior where moths and flames are the main components [65]. The candidate solutions are moths, and the variables are the moths' positions in space. Therefore, moths can fly in 1D, 2D, 3D, or hyperdimensional space (of dimension *d*) with changing position vectors [66]. MFO is a PB metaheuristic algorithm; thus, the set of *n* moths is used as search agents in the problem space. Flames are the best *n* positions of moths that are obtained so far. Each moth searches around a flag (flame) and updates it in case they find a better solution. Flames are also *d* dimensional data points [65, 66]. Given a logarithmic spiral, a given moth updates its position with respect to a given flame [65, 66] as in Equation (2.21).

$$S(M_i, F_j) = D_i . e^{bt} . \cos(2\pi t) + F_j \tag{2.21}$$

where D_i means the Euclidian distance of the i^{th} moth for the j^{th} flame, *b* is a constant for defining the shape of the logarithmic spiral, M_i denotes the i^{th} moth, F_j is the j^{th} flame, and *t* is a random number in $[-1,1]$.

The next position of a moth is defined with respect to a flame, as shown in the above equation. In the spiral equation, *t* parameter defines how close the next position of the moth should be to the flame [65, 66]. Therefore, a hyper-ellipse can be assumed to be present around the flame in all directions and the next position of the moth would be found within this space. To further emphasize exploitation, the algorithm assumed that *t* is a random number in [*r*, 1] where *r* is linearly decreasing from -1 to -2 throughout the iteration; this is called convergence constant [66]. Through this approach, moths tend to exploit their corresponding flames more accurately in a way that is proportional to the number of iterations. This approach enhances the probability of converging to a global solution. Moreover, a given moth is required to update its position by using only one of the flames. The flames are sorted according to their fitness values during each iteration and after the flame list is updated. The moths then update their positions with respect to their corresponding flames [66]. To allow the best promising solutions to be exploited properly, the number of flames to be followed is decreased with respect to the iteration number, as shown in Equation (2.22).

$$N_{flames} = round\left(N - l \, . \, \frac{N-1}{T}\right) \tag{2.22}$$

where l is the current iteration number, N is the maximum number of flames, and T indicates the maximum number of iterations.

The MFO algorithm has been applied in the domain of machine learning to find optimal features. It has also been combined with the wrapper-based FS method [66] and compared with PSO and GA according to different evaluation criteria on 18 different datasets from the UCI Machine Learning Repository. Experiment results showed that MFO achieved significantly better performance than GA and PSO, which is a common result in wrapper-based FS. However, MFO is a time-consuming process [66].

```
input: n number of moths, N maximum number of flames, T number of iterations.
output: $f_{best}$ optimal flame position, $f(f_{best})$ fitness value for $f_{best}$.
Initialize a population of n flames positions randomly in the search space.
while Stopping criteria not met do
  Update the number of flames to be used $N_{flames}$ according to eq (2.22).
  Calculate the fitness of all the n moths.
  if first iteration then
    Sort the moths from best to worst and place the result in flame matrix.
  else
    Merge the population of past moths and flames.
    Sort the merged population from best to worst.
    Select the best N positions from the sorted merged population as the flames.
  end
  Calculate the convergence constant r.
  for each $Moth_i$ with $i \leq n$ do
    Calculate t as $t = (r - 1) * rand + 1$. With rand a random number drawn from uniform
    distribution in the range [0, 1].
    if $i \leq N$ then
      Update $Moth_i$ position according to $Flame_i$ using eq (2.21).
    else
      Update $Moth_i$ position according to $Flame_{N_{Flames}}$ using eq (2.21).
    end
  end for
```

Figure 2.16 Pseudo-code of MFO [66]

2.4.9 Multi Verse optimizer

Multi Verse Optimizer (MVO) [67] is a PB metaheuristic algorithm whose mathematical models and algorithm were designed based on three concepts from theories related to the multiverse in cosmology; these concepts are white hole, black hole, and wormhole [67]. In MVO, the white hole and black hole concepts are modeled to represent the exploration process in the search space, while the model of wormholes simulates the exploitation process. Each candidate solution generated by MVO is called a universe and represented a vector of real elements [68].

A binary version of the MVO algorithm called BMVO [68] has been proposed mainly for use with FS problems. BMVO proposed a new formulation over the original MVO, where the universe contains a vector of binary bits 0 or 1. A V-shaped transfer function was integrated into BMVO to map continuous values to probabilities. The second modification was performed on the update of the universe per each generation, where it was affected by the best universe that was found. The BMVO was compared with four other binary FS algorithms (BBAT, BPSO, BDA, and BGWO) using seven benchmark datasets based on classification accuracy and number of selected features. BMVO achieved better performance on most of the seven datasets compared with other FS algorithms.

2.4.10 Bat Algorithm

Bat Algorithm (BA) [24] is a PB metaheuristic algorithm that takes its inspiration from the nature of bats. The main characteristics of this algorithm are primarily influenced by the behavior of microbats [24]. Pulse rates and emission are the main two parameters that are used in this algorithm, and their values can be adjusted [69]. BA utilizes the frequency tuning method to expand the variety of solutions that exist in the population at the same time. It also uses automatic zooming to adjust the exploration and exploitation throughout the process by imitating the variation of the heartbeat outflow and the loudness of bats while they hunt prey. Microbats have a unique ability called echolocation, which enables them to find their prey and distinguish among different insects in total darkness. With these characteristics, the algorithm works efficiently and is able to initiate rapidly [69].

In BA, an artificial bat has position, velocity, and frequency vectors, which are updated throughout iterations as (2.23), (2.24), and (2.25) [69].

$$V_i(t+1) = V_i(t) + (X_i(t) - Gbest)F_i \tag{2.23}$$

$$X_i(t+1) = X_i(t) + V_i(t+1) \tag{2.24}$$

where $Gbest$ is the best solution attained so far, and F_i indicates the frequency of i^{th} bat, which is updated in each iteration as follows:

$$F_i = F_{min} + (F_{max} - F_{min})\beta \tag{2.25}$$

where β is a random number of a uniform distribution in [0, 1]. Equations (2.23) and (2.25) show that different frequencies enable artificial bats to have diverse inclinations to the best solution. These equations could guarantee the exploitability of BA. A random walk procedure is used to perform the exploitation as follows [69]:

$$X_{new} = X_{old} + \varepsilon \mathrm{A}^t \tag{2.26}$$

In this formula, ε is a random number in [-1,1], and A is the loudness of the sound emitted by bats to perform exploration process. BA can be considered a balanced combination of PSO and intensive local search. The balance between these techniques is controlled by the loudness (A) and pulse emission rate (r), which are updated as follows [69]:

$$A_i(t+1) = \alpha A_i(t) \tag{2.27}$$

$$r_i(t+1) = r_i(0)[1 - \exp(-\gamma t)] \tag{2.28}$$

where α and γ are constants. Eventually, A_i will equal zero, while the final value of r_i is $r_i(0)$. Loudness and rate are updated when the new solutions are improved to ensure that the bats are moving toward the best solutions [69].

The original version of this algorithm was appropriate for continuous problems but not for direct application to binary problems. A binary version [69] of this algorithm called binary bat algorithm (BBA) was thus developed to solve FS problems. BBA was compared with binary PSO and GA over 22 benchmark functions; BBA performed better than binary PSO and GA did on the majority of the benchmark functions for FS problems.

```
Initialize the bat population $X_i = (i = 1,2, \dots n)$ and $V_i$
Define pulse frequency $F_i$
Initialize pulse rates $r_i$ and the loudness $A_i$
while ($t <$ Max number of iterations)
  Generate new solutions by adjusting frequency,
  updating velocities and positions (equations (2.23), (2.24), (2.25))
  if ($rand > r_i$)
    Select a solution among the best solutions randomly
    Generate a local solution around the selected best solution
  end if
  Generate a new solution by flying randomly
  if ($rand < A_i$ and $f(x_i) < f(Gbest)$)
    Accept the new solutions
    Increase $r_i$ and reduce $A_i$
  end if
  Rank the bats and find the current Gbest
end while
```

Figure 2.17 Pseudo-code of BA [69]

Chapter 3

Research Methodology

In this chapter, we discuss our research methodology that is used in this work to solve the FS problems. The research methodology which is employed is illustrated in section 3.1, and a summary of this methodology is provided in section 3.2.

3.1 Research Methodology

The research methodology that is used in this work is shown in Figure 3.1. It is composed of six phases: the initial phase, the pre-processing phase, the construction phase, the improvement phase, the evaluation phase, and the (improvement and modification) phase.

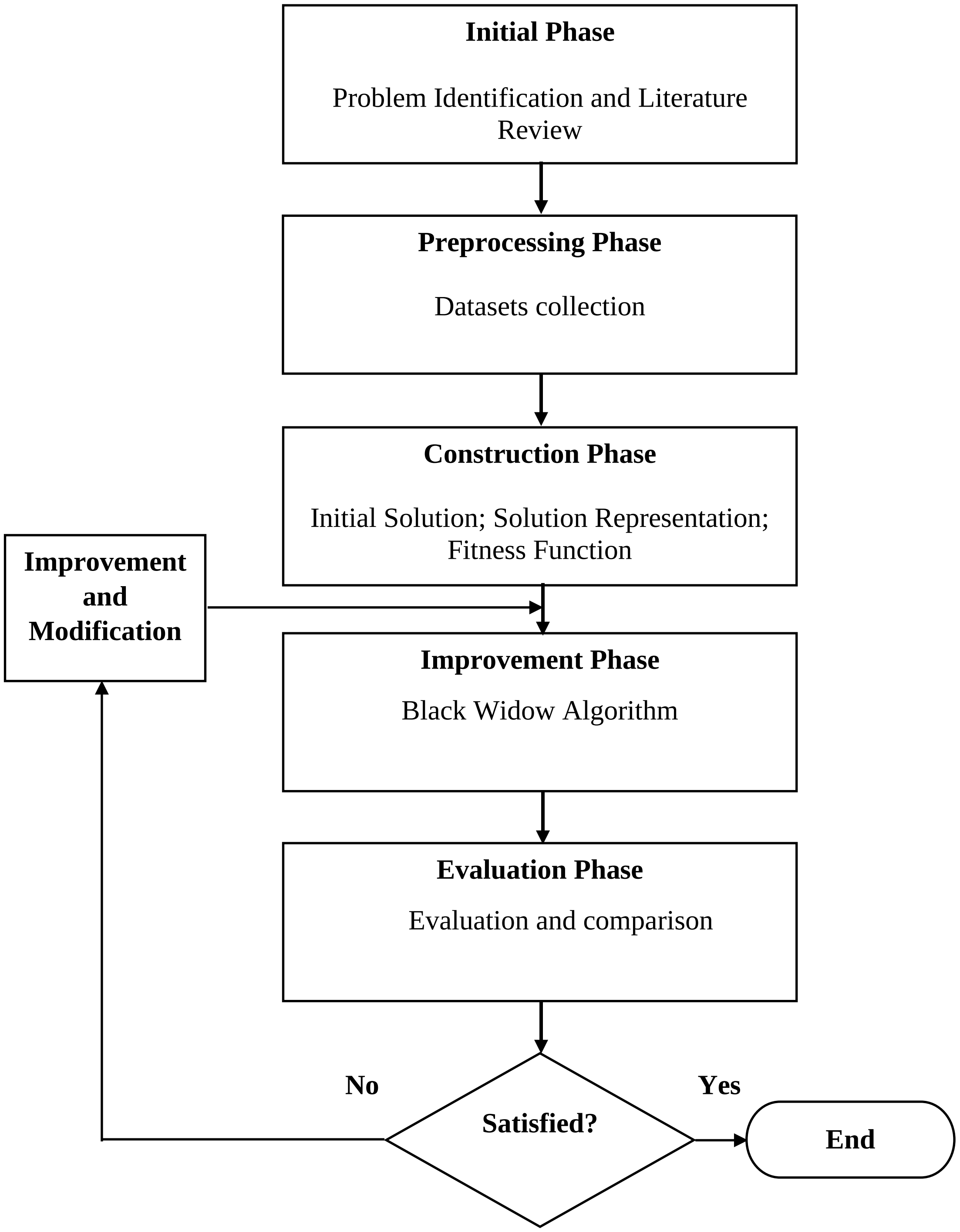

Initial Phase
Problem Identification and Literature Review
Preprocessing Phase
Datasets collection
Construction Phase
Initial Solution; Solution Representation; Fitness Function
Improvement and Modification
Improvement Phase
Black Widow Algorithm
Evaluation Phase
Evaluation and comparison
No
Satisfied?
Yes
End

Figure 3.1 Research methodology

3.1.1 Initial Phase

This phase is concerned with identifying the problem and the related works. The main objective is to understand in deep the problem formulation and the evaluation function and review the state-of-the-art methods that have been developed.

3.1.2 Preprocessing Phase

This phase is concerned with understanding and gathering information on the FS problems. In particular, this phase is focused on reading the problem instances, generating the auxiliary matrix and selecting the appropriate solution representation. Each solution is represented by a binary vector where “0” refers to the unselected feature and “1” refers to the selected feature. This structured format is used to transform the targeted original datasets into a structured format. The following subsection shows the datasets that have been used in the experiments.

- **Datasets**

Twenty-eight well-known datasets from the UCI (University of California Irvine) machine learning repository [70] have been used to investigate the performance and strength of our proposed methods. The datasets could be freely downloaded from the website https://archive.ics.uci.edu/ml/datasets.php. These datasets include real-valued attributes and have been adopted to compare all the FS methods equally [6, 7, 17]. A brief description of the datasets is displayed in Table 3.1: This shows the number of features, objects, classes, and the domain to which each of these datasets belongs.

Table 3.1 Datasets description

No.	Datasets	Features	Objects	Classes	Domain
1	Breastcancer	9	699	2	Medical
2	BreastEW	30	569	2	Medical
3	CongressEW	16	435	2	Politics
4	Exactly	13	1000	2	Medical
5	Exactly2	13	1000	2	Medical
6	HeartEW	13	270	5	Medical
7	IonosphereEW	34	351	2	Electronic
8	Lymphography	18	148	4	Medical
9	M-of-n	13	1000	2	Medical
10	PenglungEW	325	73	2	Medical
11	SonarEW	60	208	2	Medical
12	SpectEW	22	267	2	Medical
13	Tic-tac-toe	9	958	2	Game
14	Vote	16	300	2	Politics
15	WaveformEW	40	5000	3	Physical
16	Zoo	16	101	7	Artificial
17	Colon	2000	62	2	Medical
18	Parkinsons	22	195	2	Medical
19	Lungcancer	21	226	2	Medical
20	Leukemia	7129	72	2	Medical
21	Dermatology	34	366	6	Medical
22	Semeion	256	1593	10	Handwriting
23	Satellite	36	5100	2	Physical
24	Spambase	57	4601	2	Computer
25	Segment	19	2310	7	Images
26	Credit	20	1000	2	Business
27	KrvskpEW	36	3196	2	Game
28	Plants-100	64	1599	100	Agriculture

3.1.3 Construction Phase

In this phase, the concentration is on finding the initial solution by employing a random constructive heuristic in order to formulate a random initial solution of the Black Widow Optimization algorithm [19] (BWO) which is tested in this study.

3.1.3.1 Solution Representation

Due to the binary nature of FS problems, we adopted the binary representation to represent each solution [9, 16, 22]. In this kind of representation, a solution is indicated in a one-dimensional vector. The length of the vector is dependant on the number of features of the original dataset, for example, if S features are contained in the dataset, the solution length is S. The cell value in the vector is indicated by ‘1’ or ‘0’. The value ‘1’ indicates that the corresponding feature is chosen while ‘0’ indicates that the corresponding feature is not chosen.

The construction of the initial solution is generating randomly, i.e., a ‘1’ or ‘0’ value is assigned randomly for each cell in the vector. Figure 3.2 indicates the representation of the solution (with length=S) as four features are chosen.

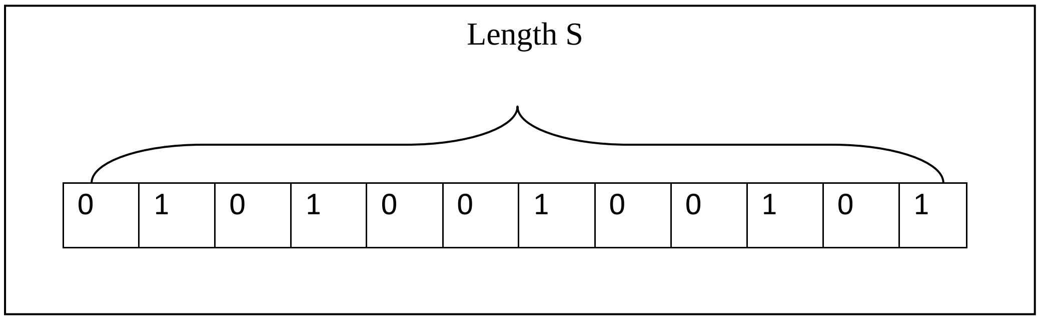


Figure 3.2 A solution representation

3.1.3.2 Fitness Function

FS can be considered as a multi-objective optimization problem where two contradictory objectives are to be achieved: a minimal number of selected features and the highest classification accuracy. The smaller is the number of features in the solution and the higher the classification accuracy, the better the solution. Each solution is evaluated according to the proposed fitness function, which depends on the classifier to get the classification accuracy of the solution, and on the number of selected features in the solution generated by the search algorithm [7, 14].

The fitness function based on the FS wrapper method is shown in the equation below:

$$f = \alpha\gamma_R\,(D) + \beta\,\frac{|R|}{|C|} \qquad (3.1)$$

Where $\gamma_R\,(D)$ represents the classification error rate of a given classifier, $|R|$ is the cardinality of the selected subset, $|C|$ is the total number of the original features in the dataset, and α, β are two weight parameters corresponding to the importance of classification quality and subset length, $\alpha \in [0,1]$ and $\beta = (1 - \alpha)$ [12, 16, 20].

3.1.4 Improvement Phase

This phase is concerned with improving the solutions that are generated in the construction phase. In this thesis, we propose a Black Widow Optimization [19] (BWO) algorithm in the improvement phase. BWO is a nature-inspired algorithm that mimics the Black Widow's life cycle in nature. BWO is a population-based-metaheuristic algorithm that operates on a population of solutions and aims to iteratively improve them for a certain number of iterations. We selected BWO in this thesis due to its success in solving engineering problems, has few parameters and is easy to understand and implement [19]. More details about BWO implementation for the FS problems are given in the next chapter (Chapter 4).

3.1.5 Evaluation Phase

In this phase, we evaluate the performance of the proposed method by testing it using the twenty-eight FS datasets provided by UCI [70]. The results are compared with up-to-date FS algorithms based on two criteria: classification accuracy and features selected. A calculation of the classification accuracy and the features selected were carried out by taking the average accuracy, taking the average number of features selected for the optimum solution of the proposed method and running it a number of times [6, 7, 17].

3.2.6 Improvement and Modification Phase

In the previous phase, the performance of our proposed method will be compared to various up-to-date FS algorithms. In this phase, if the performance of our proposed method is not competitive with the compared methods in some cases, an improvement and modification upon the proposed method could be conducted. In this thesis, we found that, based on the experimental results, the performance of the proposed algorithm needed some improvements. Therefore, we further enhanced the exploitation process of the proposed algorithm by combining it with a local search algorithm to maximize the performance and to achieve a good quality solution as explained in next Chapters (4 and 5).

3.2 Summary

This chapter presented the research methodology that we were used in this thesis which has six different phases (the initial phase, the pre-processing phase, the construction phase, the improvement phase, the evaluation phase, and the improvement and modification phase). The first phase is concerned with problem identification and studying the state-of-the-art methods that were proposed for the FS problems. The data collection is presented in second phase. Phase three discussed how to construct the initial solution, the solution representation, and the fitness function. The proposed method for the FS problems is discussed in the fourth phase which is concerned with how to improve the solution constructed in previous phase. In the fifth phase, the performance of our proposed method is compared with up-to-date FS algorithms. Finally, in the sixth phase we further investigated the performance of our proposed method and decided whether to further improve it or not.

Chapter 4

Black Widow Optimization Algorithm for Feature Selection

In this chapter, a new metaheuristic algorithm, known as the Black Widow Optimization (BWO), has been selected to solve the FS problems, due to its great results when applied to a range of daunting design problems in the field of engineering [19]. Therefore, we are proposing a modified Binary Black Widow Optimizing (BBWO) algorithm to deal with the FS problems. The structure of this chapter as follows: a brief description of the BWO algorithm and its proposed uses for FS (BBWO) are explained in Sections 4.1 and 4.2 respectively. Next, the experiments for BBWO are presented in Section 4.3, which contain: (i) the implementation setup (ii) the evaluation criteria and parameters setting (iii) the results and discussions. Then, the summary of this chapter is presented in Section 4.4.

4.1 The Black Widow Optimization Algorithm

The BWO is a population metaheuristic algorithm recently proposed with the intention of optimizing engineering design [19]. The BWO process was inspired, in essence, by the singular mating behaviour exhibited by black widow spiders, a process that includes an exclusive stage: cannibalism. Because of the operators involved in the process, the BWO is considered as one of the evolutionary algorithms (EA) [19]. BWO and GA share a similar component structure, which proves it as an important evolutionary method. The BWO, compared to other EA, in certain criteria mimics the natural evolution process, and the most notable of the lot are selection, reproduction, and mutation [19]. The diverse nature of these criteria distinguishes between the various EA. Nevertheless, BWO inherits the extraordinary mating habits of black widow spiders. However, there are a few dissimilarities that distinguish this algorithm from other EA, and this helps to enhance the results in an analysis of complicated problems [19]. The BWO technique is inspired by Darwin's natural selection theory, which is defined as generational descent accompanied by modification, and introduced the concept of species being subtly adjusted over time and new species arising as a result [19]. The BWO approach is designed to deliver rapid convergence and to avoid local optima, and it is, therefore, particularly appropriate for solving several kinds of optimization problems that involve a number of local optima, because BWO maintains equilibrium between the exploration and exploitation stages [19].

4.2 The Proposed BBWO Algorithm for FS

The BWO is a simple but effective metaheuristics algorithm [19] and is utilized here for better solution efficiency and dependability while discovering the most prolific solutions to the FS problems. In this section, the pseudo-code of the proposed method (BBWO) is displayed in Figure 4.1. The main steps of our recommended algorithm for use in this research in pursuit of the FS problems are presented in a number of the following steps.

```
Set the parameters value:
population size ($N_{pop}$) = 20; number of iterations ($max$Iteration) = 10; number of features ($N_f$) = dimension size; procreate rate ($P_r$) = 0.6; mutation rate ($M_r$) = 0.4
# Initialization process
Generating the initial population of solutions randomly ($N_{pop} \times N_f$). Each solution represents one widow, which is indicated in one-dimension vector $1 \times N_f$ (as explained in Sec. 4.2.2)
Calculate the fitness value for each solution using Eq. (3.1)
Evaluate all solutions in the population based on their fitness value and save the them in pop1
Set the best solution in the population as $W^*$
based on $P_r$ calculate the number of reproductions $N_r$
based on $M_r$ calculate the number of mutations $N_m$
Define $I$=0
while $I < max$Iteration do
  #Procreate and cannibalism processes
  for $i$ = 1 to $(N_r/2)$ do
    Randomly select two solutions $w_1$, $w_2$ as parents from pop1
    Generate two children $c_1$, $c_2$ using Eq. (4.3)
    Transformation $c_1$, $c_2$ to binary nature using Eq. (4.1 and 4.2) (as explained in sec 4.2.4)
    Calculate the fitness value of $c_1$, $c_2$ using Eq. (3.1)
    Destroy the father $w_1$ or $w_2$ based on their fitness value (cannibalism process)
    Remove one of the two children $c_1$ or $c_2$ based on their fitness value (sibling cannibalism)
    Save the remaining solutions in pop2
  end for
  #Mutation process
  for $i$ = 1 to $N_m$ do
    Randomly select a solution from pop1
    Apply the mutation process on the selected solution (as explained in Figure 4.5)
    Save the result (the new solution) in pop3
  end for
  #Update the population
  Update the population = pop2+pop3
  Evaluate all solutions in the population using Eq. (3.1)
  Update $W^*$ if there is a better solution
  $I$= $I$+1
end while
returning the best solution $W^*$
```

Figure 4.1 Pseudo-code of BBWO

4.2.1 Solutions Representation

In the BWO algorithm, the possible solution to every problem has been envisioned in terms of the attributes of the black widow spider. To solve the optimization problem, the structure should be viewed as an array in a N_{var} dimensional optimization problem. A widow is an array describing the solution of the problem, and this array can, in turn, be defined as: $w = [x_1, x_{2,}, x_{3,} \dots, x_{N_{var}}]$. In this study, the proposed BBWO algorithm uses the binary representation to represent a population of solutions (N_{pop}). Each solution represents a single widow. In binary representation, a solution is shown by a one-dimensional vector. The length of the vector varies in accordance with the feature number of the original dataset: for example, if S features are contained in the dataset, this means that the solution length is S. The cell value in the vector is indicated by a '1' or a '0'. The value '1' indicates that the corresponding feature is selected, whereas '0' indicates that the feature is not selected. Since BBWO operates on a population of solutions, the population is represented by an array, where each row represents one candidate solution. Assume that the number of features is N_f. and the population size is $|N_{pop}|$, the array size will be $N_f \times |N_{pop}|$. An example of this representation is presented in Chapter 3 (Figure 3.2).

4.2.2 Initialization

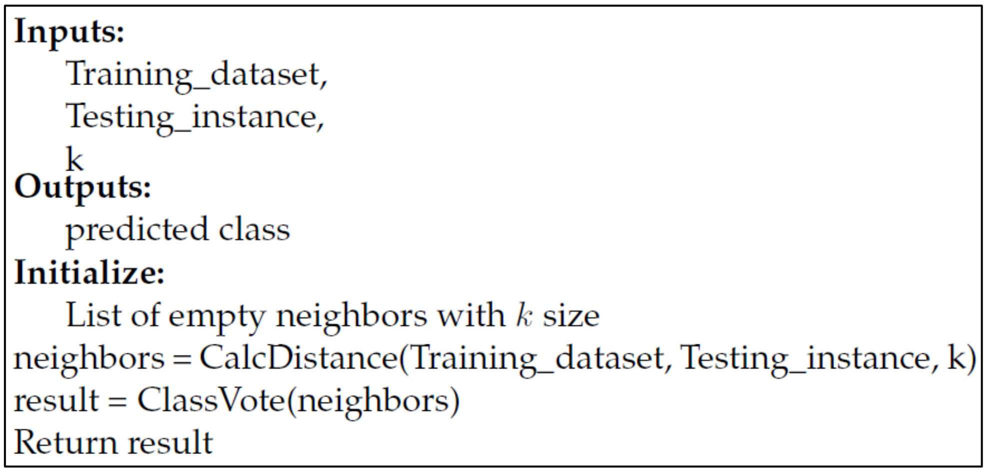
The population of solutions offered by BBWO for the FS problems is randomly generated by assigning to each cell of the solution a value of either "0" or "1". The process begins by initializing the population-size and the number of features. The algorithm then arbitrarily assigns either '0' or '1' by looping through each solution in the population. This process is repeated until all solutions in the population have been initialized. The population generation procedure is presented in Figure 4.2 below.

```
Set population size, $|\mathrm{N_{pop}}|$
Set the number of features, $\mathrm{N_f}$
$X_j^i = \theta$
For i=1 to $|\mathrm{N_{pop}}|$ do
    For j=1 to $\mathrm{N_f}$ do
        $X_j^i$ = select a randomly number either "0" or "1"
        Add $X_j^i$ to $|\mathrm{N_{pop}}|$
    End for
End for
```

Figure 4.2 Population Generation procedure

4.2.3 Fitness Function and Evaluation Method

After initializing the population of solutions, we assign to each solution (widow) a fitness value, which is represented the quality of the solution. In this thesis, the fitness value of each solution is calculated using the fitness function of the wrapper method, which is explained in equation 3.1 in Chapter 3. The wrapper method is preferred over the filter and the embedded methods as the evaluation technique because of its higher accuracy and because it is highly recommended by many researchers [7, 9, 12]. Thus, we used the wrapper method to determine the fitness value and to foster a balance between the number of selected features in each solution (the minimum) and the classification accuracy (maximum). The FS wrapper method uses the classification performance (accuracy) of a classifier to evaluate the solutions, in particular, we used the KNN classifier as a learning algorithm stage to assess the accuracy of the solutions generated by the proposed algorithm (BBWO). KNN is a supervised machine learning algorithm that is used for both classification and regression problems. It calculates the distances between the testing sample and the samples in the training dataset based on specific metrics like Euclidean distance, then it sorts the calculated distances in ascending order and picks the first k neighbours. The final step is to predict the response based on neighbours voting, where each neighbour votes for its class feature, then takes the majority vote as the prediction [46, 71]. Figure 4.3 shows the pseudo-code of the KNN algorithm.



```
Inputs:
    Training_dataset,
    Testing_instance,
    k
Outputs:
    predicted class
Initialize:
    List of empty neighbors with $k$ size
neighbors = CalcDistance(Training_dataset, Testing_instance, k)
result = ClassVote(neighbors)
Return result
```

Figure 4.3 Pseudo-code of KNN algorithm

4.2.4 Transformation Function

The positions of the search agents generated from the standard BWO are continuous values. This cannot, therefore, be directly applied to our problem because it contradicts the binary nature of the FS on selection or non-selection (0 or 1). The sigmoidal function in Eq. (4.1) and (4.2), which is considered a form of the transformation function, is used in our proposed method as a part of the reproduction process to convert any continuous value to a binary equivalent. The performance of the transformation function has been investigated and adopted by many researchers [7, 12, 16].

$$z_{s_w} = \frac{1}{1+e^{-z_w}} \tag{4.1}$$

$$z_{binary} = \begin{cases} 0, & if\ rand < z_{s_w} \\ 1, & if\ rand \geq z_{s_w} \end{cases} \tag{4.2}$$

Where each of z_{s_w} is a continuous value (feature) in the search agent for the S-shaped function, specifically in the solution *w* at dimension *d* ($w = 1, \dots, d$), *rand* is a random number drawn from the uniform distribution $\in [0,1]$. z_{binary} value and can be 0 or 1 in accordance with the value of a *rand* in comparison with the values of z_{s_w}, where e is a mathematical constant known as Euler's number.

4.2.5 Reproduction Process

To bring forth the new generation, the procreation process begins and parents (in pairs) are selected randomly to perform the procreating steps by mating. An array known as Alpha should also be generated to complete the further reproduction. Offspring c_1 and c_2 will be produced by taking α with the following equation in which w_1 and w_2 are parents]19].

$$\begin{cases} c_1 = \alpha \times w_1 + (1-\alpha) \times w_2 \\ \\ c_2 = \alpha \times w_2 + (1-\alpha) \times w_1 \end{cases} \tag{4.3}$$

This process is repeated for all pairs, where no repetition of randomly selected parents should take place. Lastly, the children and maternal parents are added to an array and sorted in accordance with their fitness value. The following Figure 4.4 is an assuming example of the procreate process for a child *y1*.

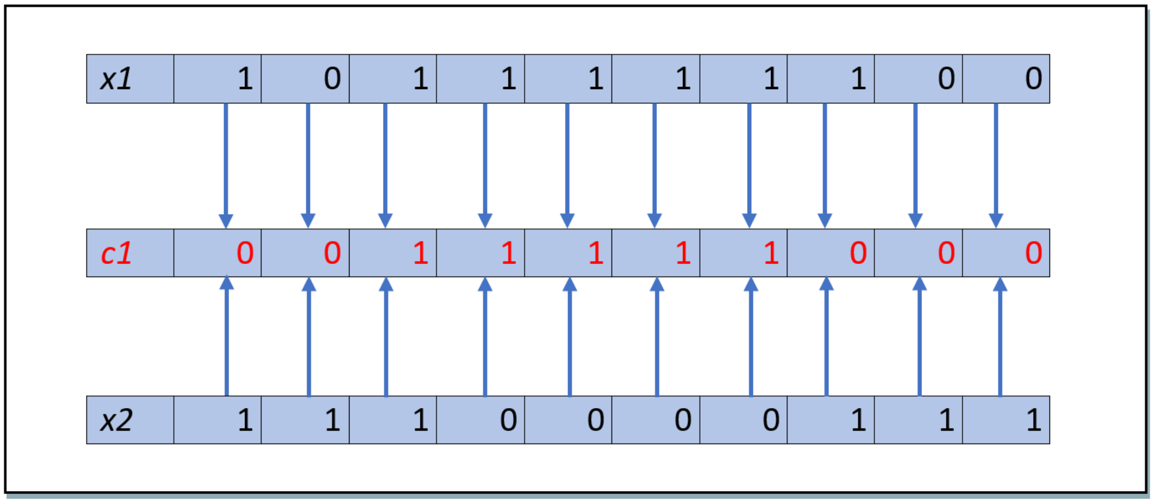


Figure 4.4 An assuming example of the procreate process

4.2.6 Cannibalism Process

Cannibalism can be classified into three kinds: sexual cannibalism where the husband gets eaten by his black widow during or after mating, sibling-cannibalism where the weaker sibling spiders get eaten by stronger siblings and the last kind where the mother gets eaten by her strongest baby [19]. The proposed method (BBWO) determines the weak or strong spiders by calculating and evaluating their fitness values. Therefore, the best solutions (surviving spiders) from the reproduction process will be selected and stored in a variable pop2.

4.2.7 Mutation Process

The procedure of mutations begins by randomly selecting a number of solutions (widows) from the *pop1* population which will be mutated individually. Two cells from each selected solution (widow) are randomly exchanged, and the new mutation solutions will be kept in *pop3*. An example in Figure 4.5 explains the mutation structure for an individual solution.

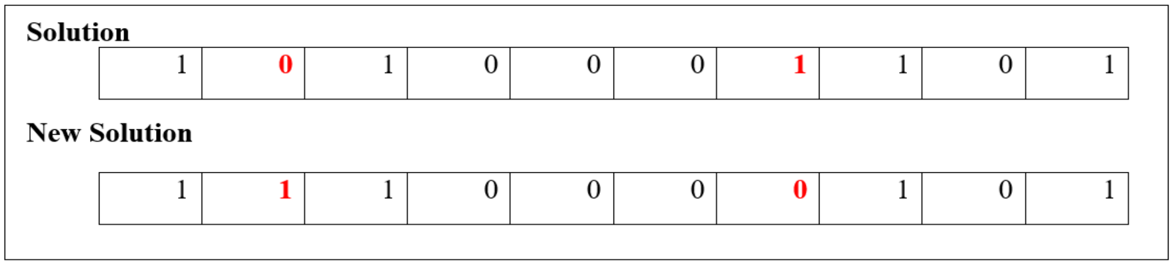


Figure 4.5 Mutation structures

4.2.8 New Population Generation

The new population can finally be generated as a combination of *pop2* and *pop3*, which will then be evaluated to return the optimal solution (W^*) of values bearing the N dimension. The BWO algorithm contains some of the parameters with which exceptional results can be achieved. These involve the cannibalism rate, the procreation rate (P_r) and the mutation rate (M_r) [19]. The proposed BBWO method determines the cannibalism rate in accordance with the fitness values Eq. (3.1), whereas the same parameter rates P_r and M_r of the standard BWO [19] have been used in the BBWO process.

4.2.9 The Flowchart of BBWO

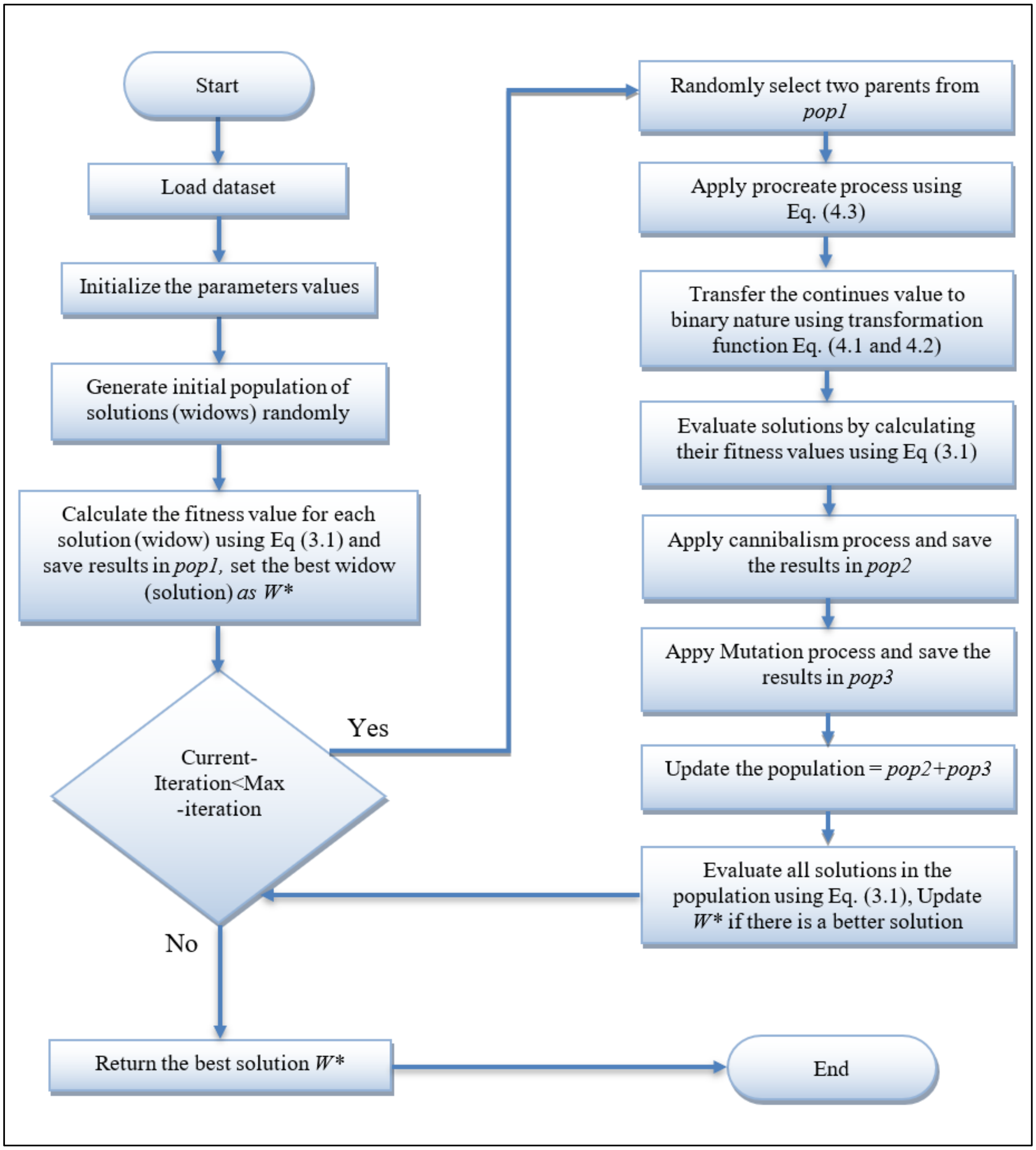

Start
Load dataset
Initialize the parameters values
Generate initial population of solutions (widows) randomly
Calculate the fitness value for each solution (widow) using Eq (3.1) and save results in *pop1,* set the best widow (solution) *as W**
Current-Iteration<Max -iteration
Yes
No
Return the best solution *W**
End
Randomly select two parents from *pop1*
Apply procreate process using Eq. (4.3)
Transfer the continues value to binary nature using transformation function Eq. (4.1 and 4.2)
Evaluate solutions by calculating their fitness values using Eq (3.1)
Apply cannibalism process and save the results in *pop2*
Appy Mutation process and save the results in *pop3*
Update the population = *pop2*+*pop3*
Evaluate all solutions in the population using Eq. (3.1), Update *W** if there is a better solution

Figure 4.6 Flowchart of BBWO

4.2.10 The Implementation Steps of BBWO

Step 1: Select the dataset.

Step 2: Split the dataset into testing and training partitions.

Step 3: Selecting the BBWO initial parameters: $N_{pop}, max\text{Iteration}$, N_f, P_r, M_r.

Step 4: Generating the initial population of solutions randomly $N_{pop} \times N_f$. Each solution represents one widow, which is indicated in a one-dimension vector $1 \times N_f$.

Step 5: Calculating the fitness function for the initial population using Eq. (3.1), save the results in *pop1,* set the best widow (solution) as *W**.

Step 6: Reproduction a new generation (procreate and cannibalism) processes, by selecting two parents from *pop1*, to generate *C* children using Eq. (4.3), transformation *C* to binary nature using Eq. (4.1 and 4.2), calculate the fitness values for C, remove the father, and some of the children, then save the reaming solutions (surviving spiders) into *pop2.*

Step 7: Apply the mutation process by selecting a number of solutions from *pop1*, then select two positions randomly in each solution and swap them and save the new solutions into *pop3.*

Step 8: Update the population = (*pop2+pop3*).

Step 9: Evaluate the population using Eq. (3.1), and update *W** if there is a better solution.

Step 10: Checking the criteria of convergence so, if $max\text{Iteration}$or is met. Then, the algorithm will stop, and return the best solution *W**, otherwise, it will return to Step 6.

4.3 Experiments

4.3.1 Implementation Setup

Python programming was used to implement the proposed method (BBWO), and the work was carried out via a Windows 10 64-bit operating system, Core i5 processor, operating at 1.8 GHz and with 8 GB of RAM. A wrapper approach based on a KNN classifier (where *K*=5 [6, 17] is used to evaluate the fitness value of the selected feature subsets generated by BBWO. One of the most used supervised learning algorithms; KNN classifier, that classifies the test set based on the distance function with respect to the training set [7]. Well-known twenty-eight datasets from the UCI machine learning repository [6, 7, 17, 70] have been used to investigate the performance and the strength of our proposed method as presented in Table 3.1 (Chapter 3). The dataset is randomly split into 80% for the training set and 20% for the test set, as recommended in real-world datasets [4].

4.3.2 Evaluation Criteria and Parameters Setting

The performance of our proposed method (BBWO) is compared with the six up-to-date binary FS algorithms (BPSO [63], BMVO [68], BGWO [7, 12], BMFO [66], BWOA [17], BBAT [69]) based on the two evaluation criteria: classification accuracy, and feature selected. A calculation of the classification accuracy and the feature selected were carried out by taking the average accuracy and the average number of features selected for the optimum solution of the proposed algorithms, run on a number of independent runs. To ensure an impartial comparison and a correct evaluation between our proposed method and other FS algorithms, we reimplemented the six FS algorithms

(BPSO, BMVO**,** BGWO, BMFO, BWOA, BBAT) using the same parameters values (population-size, iterations, runs, K (KNN classifier), α, β) as illustrated in Table 4.1 , and same transformation function as explained in section 4.2.4. While the numerical results for all FS algorithms (BBWO, BPSO, BMVO**,** BGWO, BMFO, BWOA, BBAT) accepted exactly as we tested using the twenty-eight popular UCI datasets as shown in Table 3.1 in (Chapter 3). Note, the pseudo-codes of the six FS algorithms are available as an open source at [2, 36, 76, 77].

Table 4.1. Parameters values

Parameters Name	Value
Population-size	20
No. of iterations	10
Number of independent runs	20
K (*KNN* classifier)	5
Dimension-size	No. of features
Number of iterations for hill climbing	20
pr	0.6
mr	0.4
α	0.99
β	0.01

4.3.3 Experiment Results and Discussion of BBWO

In this section the experiment results of our proposed method (BBWO) presented in Table 4.2, this shows (the dataset name, number of features in each dataset, the BBWO results of the classification accuracy and feature selected). Following, the results and discussion between BBWO and the six FS algorithms (BPSO, BMVO**,** BGWO, BMFO, BWOA, BBAT), based on the two evaluation criteria: the classification accuracy (maximizing), and feature selected (minimizing), as illustrated in Table 4.3 and Table 4.4 respectively.

Table 4.2 Experiment results for the BBWO

Datasets Name	Number of features	Classification Accuracy	Feature selected
Breastcancer	9	0.97	3.00
BreastEW	30	0.94	12.25
CongressEW	16	0.95	4.60
Exactly	13	0.91	3.75
Exactly2	13	0.77	3.65
HeartEW	13	0.84	3.80
IonosphereEW	34	0.88	13.75
Lymphography	18	0.85	6.80
M-of-n	13	0.95	7.00
PenglungEW	325	0.90	150.75
SonarEW	60	0.86	24.40
SpectEW	22	0.81	8.50
Tic-tac-toe	9	0.80	3.80
Vote	16	0.93	4.05
WaveformEW	40	0.88	20.60
Zoo	16	0.92	5.05
Parkinsons	22	0.89	7.70
Lungcancer	21	0.90	8.40
Colon	2000	0.87	959.20
Leukemia	7129	0.86	3531.90
Dermatology	34	0.97	15.70
Semeion	256	0.93	130.00
Satellite	36	0.99	12.20
Spambase	57	0.92	28.60
Segment	19	0.96	8.70
Credit	20	0.79	7.60
KrvskpEW	36	0.95	19.00
Plants-100	64	0.80	33.50

Table 4.3 Comparison BBWO with all algorithms based on the classification accuracy

Datasets Name	BBWO	BPSO	BMVO	BGWO	BMFO	BWOA	BBAT
Breastcancer	**0.97**	0.96	**0.97**	0.96	**0.97**	**0.97**	0.96
BreastEW	0.94	0.94	0.94	**0.95**	0.94	0.93	0.94
CongressEW	**0.95**	0.92	**0.95**	**0.95**	**0.95**	**0.95**	0.94
Exactly	**0.91**	0.76	0.89	0.74	0.90	**0.91**	0.73
Exactly2	**0.77**	**0.77**	0.76	0.75	0.76	0.74	0.74
HeartEW	0.84	0.81	**0.85**	0.84	**0.85**	**0.85**	0.82
IonosphereEW	**0.88**	0.86	**0.88**	**0.88**	**0.88**	**0.88**	**0.88**
Lymphography	**0.85**	0.82	0.84	0.82	**0.85**	0.83	0.81
M-of-n	0.95	0.83	**0.99**	0.88	0.98	0.98	0.81
PenglungEW	**0.90**	0.87	0.89	0.89	0.89	0.88	0.88
SonarEW	0.86	0.86	**0.87**	0.86	0.86	**0.87**	0.86
SpectEW	0.81	0.81	0.81	**0.82**	**0.82**	0.81	0.81
Tic-tac-toe	0.80	0.74	0.81	0.78	**0.82**	0.81	0.76
Vote	0.93	0.91	**0.94**	**0.94**	**0.94**	**0.94**	0.93
WaveformEW	**0.88**	0.86	**0.88**	0.87	**0.88**	**0.88**	0.83
Zoo	**0.92**	0.89	0.89	0.88	0.88	0.90	0.89
Parkinsons	**0.90**	0.88	0.89	0.86	0.89	0.89	0.88
Lungcancer	0.90	0.88	**0.91**	0.90	**0.91**	**0.91**	0.90
Colon	0.87	0.86	**0.89**	0.87	0.87	0.87	0.87
Leukemia	**0.86**	0.83	**0.86**	0.85	**0.86**	**0.86**	0.85
Dermatology	**0.97**	0.89	0.96	0.95	**0.97**	**0.97**	0.92
Semeion	**0.93**	0.92	**0.93**	0.92	**0.93**	**0.93**	0.92
Satellite	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**
Spambase	**0.93**	0.88	**0.93**	0.92	**0.93**	**0.93**	0.89
Segment	**0.96**	0.94	**0.96**	**0.96**	**0.96**	**0.96**	0.94
Credit	**0.79**	0.76	**0.79**	0.77	0.78	**0.79**	0.78
KrvskpEW	0.95	0.90	0.96	0.95	**0.97**	**0.97**	0.87
Plants-100	**0.80**	0.78	0.79	0.78	**0.80**	0.79	0.77
Average	0.8932	0.8614	0.8935	0.8760	0.8939	0.8925	0.8632
Rank	3	7	2	5	1	4	6

Table 4.4 Comparison BBWO with all algorithms based on the features selected

Datasets Name	BBWO	BPSO	BMVO	BGWO	BMFO	BWOA	BBAT
Breastcancer	**3.00**	3.40	4.55	5.15	4.35	4.60	3.45
BreastEW	12.25	11.40	**10.95**	13.55	13.20	12.12	13.15
CongressEW	4.60	5.20	4.25	5.95	5.40	**4.20**	5.55
Exactly	**3.75**	5.30	7.25	7.05	7.05	6.65	5.80
Exactly2	3.65	3.95	2.33	5.40	3.15	**2.10**	3.50
HeartEW	3.80	4.25	3.45	4.15	3.70	**3.45**	4.65
IonosphereEW	13.75	14.35	13.35	15.90	15.00	**12.55**	16.55
Lymphography	6.80	7.60	6.66	7.56	7.35	**6.35**	7.55
M-of-n	7.00	**5.50**	7.22	8.00	6.75	7.35	6.15
PenglungEW	151.75	154.80	152.35	155.20	152.45	**146.35**	156.85
SonarEW	24.40	27.05	25.55	28.05	28.95	**23.85**	27.00
SpectEW	8.50	8.95	8.22	10.15	**8.20**	8.75	9.85
Tic-tac-toe	**3.80**	4.20	4.55	4.55	4.41	4.05	4.28
Vote	**4.05**	5.05	4.95	6.35	5.85	4.50	6.15
WaveformEW	20.60	22.00	22.15	21.45	21.35	**19.45**	20.25
Zoo	**5.05**	5.59	6.35	6.65	6.13	5.75	6.50
Parkinsons	**7.70**	8.00	8.45	9.15	9.10	8.20	9.25
Lungcancer	**7.00**	7.25	8.66	9.35	8.90	8.05	8.95
Colon	980.22	961.65	963.25	965.55	962.15	**943.55**	963.35
Leukemia	3531.90	3555.82	3571.85	3535.85	3534.55	**3511.35**	3513.50
Dermatology	**14.70**	16.85	15.95	16.70	16.60	16.45	16.50
Semeion	130.00	131.85	127.00	128.6	131.60	126.80	126.70
Satellite	**9.20**	13.01	10.55	12.40	11.40	10.10	12.45
Spambase	28.60	29.77	26.55	30.50	**26.25**	26.50	27.25
Segment	**7.70**	8.72	9.25	9.90	9.95	8.90	9.60
Credit	**7.22**	8.41	7.95	8.50	8.30	7.72	8.75
KrvskpEW	19.00	19.73	19.81	21.30	18.56	**17.92**	18.45
Plants-100	**32.50**	35.55	33.15	33.80	33.32	34.15	35.50
Average	180.44	181.61	181.66	181.66	180.85	178.27	180.26
Rank	3	5	6	6	4	**1**	2

From the results given in Table 4.3, the proposed method (BBWO) is comparable with the six FS algorithms (BPSO, BMVO**,** BGWO, BMFO, BWOA, BBAT) based on the classification accuracy (Maximizing) of the twenty-eight datasets, and we can make the following conclusion:

- BBWO produced better results than BPSO in 23 datasets (Breastcancer, CongressEW, Exactly, HeartEW, IonosphereEW, Lymphography, M-of-n, PenglungEW, Tic-tac-toe, Vote, WaveformEW, Zoo, Parkinsons, Lungcancer, Colon, Leukemia, Dermatology, Semeion, Spambase, Segment, Credit, KrvskpEW, Plants-100), and same results in 5 datasets (BreastEW, Exactly2, SonarEW, SpectEW, Satellite).
- BBWO obtained better results than BMVO in 8 datasets (Exactly, Exactly2, Lymphography, PenglungEW, Zoo, Parkinsons, Dermatology, Plants-100), and same results in 12 datasets (Breastcancer, BreastEW, CongressEW, IonosphereEW, SpectEW, WaveformEW, Leukemia, Semeion, Satellite, Spambase, Segment, Credit), and worse results in 8 datasets (HeartEW, M-of-n, SonarEW, Tic-tac-toe, Vote, Lungcancer, Colon, KrvskpEW).
- BBWO obtained better results than BGWO in 16 datasets (Breastcancer, Exactly, Exactly2, Lymphography, M-of-n, PenglungEW, Tic-tac-toe, WaveformEW, Zoo, Parkinsons, Leukemia, Dermatology, Semeion, Spambase, Credit, Plants-100), and same results in 9 datasets (CongressEW, HeartEW, IonosphereEW, SonarEW, Lungcancer, Colon, Satellite, Segment, KrvskpEW), and worse results in 3 datasets (BreastEW, SpectEW, Vote).
- BBWO obtained better results than BMFO in 6 datasets (Exactly, Exactly2, PenglungEW, Zoo, Parkinsons, Credit), and same results in 15 datasets (Breastcancer, BreastEW, CongressEW, IonosphereEW, Lymphography, SonarEW, WaveformEW, Colon,

Leukemia, Dermatology, Semeion, Satellite, Spambase, Segment, Plants-100), and worse results in 7 datasets (HeartEW, M-of-n, SpectEW, Tic-tac-toe, Lungcancer, KrvskpEW).

- BBWO obtained better results than BWOA in 7 datasets (BreastEW, Exactly2, Lymphography, PenglungEW, Zoo, Parkinsons, Plants-100), and same results in 14 datasets (Breastcancer, CongressEW, Exactly, IonosphereEW, SpectEW, WaveformEW, Colon, Leukemia, Dermatology, Semeion, Satellite, Spambase, Segment, Credit), and worse results in 7 datasets (HeartEW, M-of-n, SonarEW, Tic-tac-toe, Vote, Lungcancer, KrvskpEW).
- BBWO produced better results than BBAT in 20 datasets (Breastcancer, CongressEW, Exactly, Exactly2, HeartEW, Lymphography, M-of-n, PenglungEW, Tic-tac-toe, WaveformEW, Zoo, Parkinsons, Leukemia, Dermatology, Semeion, Spambase, Segment, Credit, KrvskpEW, Plants-100), and same results in 8 datasets (BreastEW, IonosphereEW, SonarEW, SpectEW, Vote, Lungcancer, Colon, Satellite).

With regard to the second criterion, the average number of features selected (Minimizing) as presented in Table 4.4, the performance of the proposed BBWO is as follows:

- BBWO produced better results than BPSO in all 25 datasets (Breastcancer, CongressEW, Exactly, Exactly2, HeartEW, IonosphereEW, Lymphography, PenglungEW, SonarEW, SpectEW, Tic-tac-toe, Vote, WaveformEW, Zoo, Parkinsons, Lungcancer, Leukemia, Dermatology, Semeion, Satellite, Spambase, Segment, Credit, KrvskpEW, Plants-100), and worse results in 3 of them (BreastEW, M-of-n, Colon).
- BBWO obtained better results than BMVO in 19 datasets (Breastcancer, Exactly, M-of-n, PenglungEW, SonarEW, Tic-tac-toe, Vote, WaveformEW, Zoo, Parkinsons, Lungcancer, Leukemia, Dermatology, Satellite , Segment, Credit, KrvskpEW, Plants-100), and worse

results in 9 datasets (BreastEW, CongressEW, Exactly2, HeartEW, IonosphereEW, Lymphography, SpectEW**,** Semeion, Spambase).

- BBWO obtained better results than BGWO in all 26 datasets (Breastcancer, BreastEW, CongressEW, Exactly, Exactly2, HeartEW, IonosphereEW, Lymphography, M-of-n, PenglungEW, SonarEW, SpectEW, Tic-tac-toe, Vote, WaveformEW, Zoo, Parkinsons, Lungcancer, Leukemia, Dermatology, Satellite, Spambase, Segment, Credit, KrvskpEW, Plants-100), and worse results in 2 datasets (Semeion, Colon).
- BBWO obtained better results than BMFO in 21 datasets (Breastcancer, BreastEW, CongressEW, Exactly, IonosphereEW, Lymphography, PenglungEW, SonarEW, Tic-tac-toe, Vote, WaveformEW, Zoo, Parkinsons, Lungcancer, Leukemia, Dermatology, Semeion, Satellite, Segment, Credit, , Plants-100), and worse results in 7 datasets (Exactly2**,** HeartEW, M-of-n, SpectEW, Colon, Spambase, KrvskpEW).
- BBWO obtained better results than BWOA in 14 datasets (Breastcancer, Exactly, M-of-n, SpectEW, Tic-tac-toe, Vote, Zoo, Parkinsons, Lungcancer, Dermatology, Satellite, Segment, Credit, Plants-100), and worse results in 14 datasets (BreastEW, CongressEW, Exactly2, HeartEW, IonosphereEW, Lymphography, PenglungEW, SonarEW, WaveformEW, Colon, Leukemia, Semeion, Spambase, KrvskpEW).
- BBWO produced better results than BBAT in 20 datasets (Breastcancer, BreastEW, CongressEW, Exactly, HeartEW, IonosphereEW, Lymphography, PenglungEW, SonarEW, SpectEW, Tic-tac-toe, Vote, Zoo, Parkinsons, Lungcancer, Dermatology, Satellite, Segment, Credit, Plants-100), and worse results in 8 datasets (Exactly2, M-of-n, WaveformEW, Colon, Leukemia, Semeion, Spambase, KrvskpEW).

Overall, the comparison between the BBWO and the six FS algorithms (BPSO, BMVO, BGWO, BMFO, BWOA, BBAT), based on the classification accuracy (Maximizing) and feature selected (minimizing) of the twenty-eight datasets are presented. From the best results are highlighted in bold in Table 4.3, and Table 4.4, we can conclude that BBWO performance is produced competitive results in terms of the average of the total classification accuracy of all datasets in comparison with other six FS algorithms as illustrated in Figure 4.7. Moreover, The BBWO shows the efficiency by minimizing the feature selected as clarified by Figure 4.8, with an impressive rank of three out of seven based on the average of the total feature selected of all datasets in comparison with the six FS algorithms. Here, the results based on the two evaluation criteria (classification accuracy and feature selected) proved that BBWO is a competitive method for solving the FS problems and its performance is better or same as many of existing algorithms.

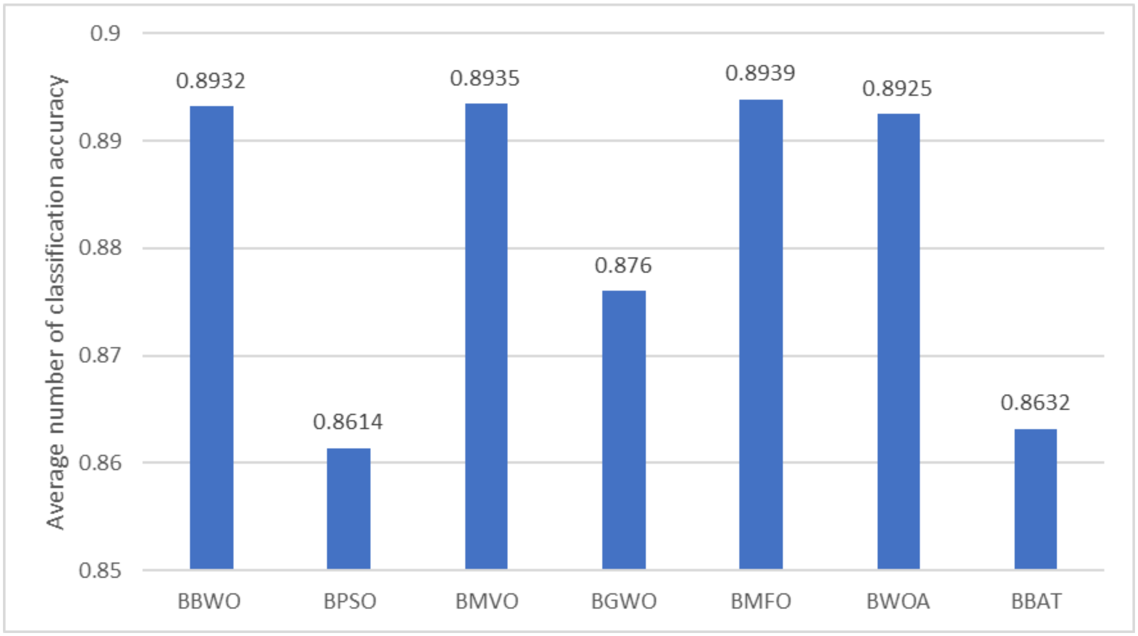


Figure 4.7 Average number of classification accuracy of all algorithms (Maximizing)

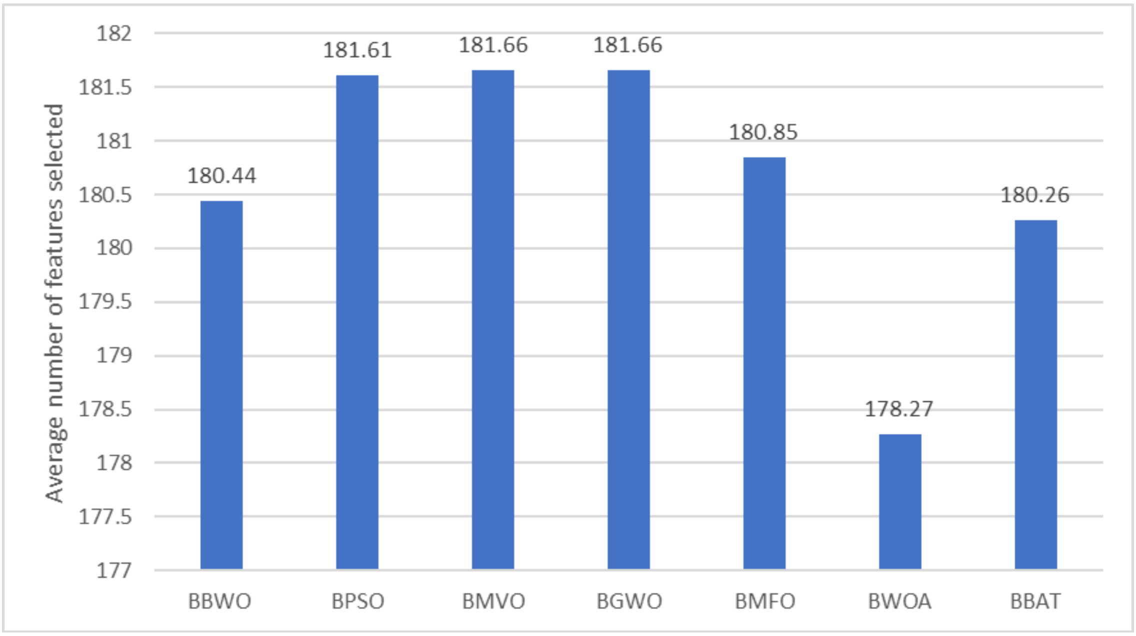


Figure 4.8 Average number of features selected of all algorithms (Minimizing)

4.4 Summary

This chapter presented a Black Widow Optimization (BWO) algorithm for FS problems. The BWO is a recent nature inspired method that mimics the nature of a black widow's life cycle in solving optimization problems. In this chapter, a modified Binary Black Widow Optimization (BBWO) algorithm is adopted with suitable solution representation to deal with FS problems which use a binary representation. Furthermore, the BBWO main steps that are responsible for generating a new solution are defined to tackle the problem. The performance of the BBWO has been tested on twenty-eight UCI benchmark datasets and from a comparison of the results, it has been proven that this method is capable of producing results as good as other up-to-date FS methods (BPSO, BMVO, BGWO, BMFO, BWOA, BBAT). However, BBWO cannot beat all FS methods on all tested datasets. It should be noted that none of the compared methods managed to beat all existing methods on all tested datasets. The main reason why BBWO did not outperformed all existing methods on many datasets is due to the nature of the metaheuristic algorithms (MAs), i.e., the fact that they come upon the problem of slow convergence because of the use of a population of solutions and lack of local exploitation [4]. The two primary factors that impact the overall performance of any population-based-metaheuristic algorithm are exploration (diversification) and exploitation (intensification). As we mentioned earlier, the exploration technique diversifies the solutions within the population, so that the search space is explored globally, whereas exploitation specializes in the neighbour’s area of a current accurate solution. Exploration thru randomization lets in the solutions to avoid being trapped on the local optima and increases the population diversity. But exploitation lets in the searching procedure to converge into an optimal solution. Having the right balance between these two parts leads to an increase in global optimality [4].

Chapter 5

An Improved BBWO Algorithm for Feature Selection

The results presented in the previous chapter showed that the proposed BBWO produced very good results and, in some cases, competitive to the best-known results. However, the results also revealed that the BBWO faces the problem of slow convergence due to the use of a population of solutions and lack of local exploitation. To further improve the exploitation process and enhance the algorithm performance, in this chapter, we are proposing an improvement to the BBWO for feature selection (denoted as IBBWO).

The structure of this chapter as following: the improved BBWO for feature selection (IBBWO) is presented in Section 5.1. The experimental results and discussions for IBBWO are presented in Section 5.2. The summary of the chapter is presented in Section 5.3.

5.1 IBBWO

The BWO is a population metaheuristic algorithm, that aims to enhance the quality of solution based on the exploration process [19]. While in the local search metaheuristics algorithms the enhancement of the solution quality is based on the exploitation process. Having the right balance between exploration and exploitation leads to an increase in global optimality [4]. Therefore, our improvement method (IBBWO) aims to increase the exploitation process to the BBWO by incorporating it with a local metaheuristic algorithm based on the hill-climbing algorithm (HCA). Thus, to further maximize the performance of BBWO.

HCA is a well-known simple local search algorithm [72]. It has been tested on various problems and shown to be an effective and efficient method that can produce good results. HCA takes an initial solution as an input and then keeps modifying it in order to get a better solution in terms of the fitness value for a fixed number of iterations [72]. In this thesis, the HCA takes place after each iteration of BBWO is completed, it is called to improve the best solution (W*) in BBWO. That is, at each iteration of the BBWO, the best solution in the population (W*) is further improved for a predefined number of iterations (defined by the user). As shown in the IBBWO pseudo-code (Figure 5.1) which combines the BBWO and HCA. The best solution (W*) of the BBWO is used as an initial solution for the HCA. The solution is modified by selecting one feature randomly, and then flip the value of the feature, i.e., if the feature value is “0” then change it to “1” (which indicate adding one feature) and if it is “1” then change it to “0” (which mean delete one feature), as explained of the flip neighbourhood operator in Figure 5.2. Then, if the fitness value of the modified solution is better than the initial one, it will replace with the old one. Otherwise, discard the new solution. Next, update the HCA iteration counter and check if the maximum number of iterations of the HCA is reached, then stop HCA, update BBWO best solution (W*) and check the criteria condition of BBWO so, if maxIterationor of BBWO is met. Then, the algorithm will stop, and return the best solution W*, otherwise, starts a new iteration for the BBWO.

```
Set the parameters value:
population size ($N_{pop}$) = 20; number of iterations ($max$Iteration) = 10; number of features ($N_f$) = dimension size; procreate rate ($P_r$) = 0.6; mutation rate ($M_r$) = 0.4
# Initialization process
Generating the initial population of solutions randomly ($N_{pop} \times N_f$). Each solution represents one widow, which is indicated in one-dimension vector $1 \times N_f$ (as explained in Sec. 4.2.2)
Calculate the fitness value for each solution using Eq. (3.1)
Evaluate all solutions in the population based on their fitness value and save the them in pop1
Set the best solution in the population as $W$*
based on $P_r$ calculate the number of reproductions $N_r$
based on $M_r$ calculate the number of mutations $N_m$
Define $I$=0
while $I$ < $max$Iteration do
  #Procreate and cannibalism processes
  for $i$ = 1 to $(N_r/2)$ do
    Randomly select two solutions $w_1$, $w_2$ as parents from pop1
    Generate two children $c_1$, $c_2$ using Eq. (4.3)
    Transformation $c_1$, $c_2$ to binary nature using Eq. (4.1 and 4.2) (as explained in sec 4.2.4)
    Calculate the fitness value of $c_1$, $c_2$ using Eq. (3.1)
    Destroy the father $w_1$ or $w_2$ based on their fitness value (cannibalism process)
    Remove one of the two children $c_1$ or $c_2$ based on their fitness value (sibling cannibalism)
    Save the remaining solutions in pop2
  end for
  #Mutation process
  for $i$ = 1 to $N_m$ do
    Randomly select a solution from pop1
    Apply the mutation process on the selected solution (as explained in Figure 4.5)
    Save the result (the new solution) in pop3
  end for
  #Update the population
  Update the population = pop2+pop3
  Evaluate all solutions in the population using Eq. (3.1)
  Update W* if there is a better solution
```

Continue…

```
...Continue

  #Hill climbing algorithm
  #Set the new solution = the best solution
  $S^*$=$W$*
  #Set the fitness value of $S^*$ = fitness values of $W$*
  F($S^*$) = F($W$*)
  $H$=1
  While $H$ < $Hmax$Iteration do
    S=$S^*$
    Randomly select one feature (i) in $S^*$, i=1, 2..., $N_f$
    if $S_i^*$=0, then $S_i^*$=1; else $S_i^*$=0
    if F($S^*$) < F($W$*) then $W$*=$S^*$; else $S^*$=S
    $H$=$H$+1
  end while
  $I$= $I$+1
end while
returning the best solution $W$*
```

Figure 5.1 Pseudo-code of IBBWO

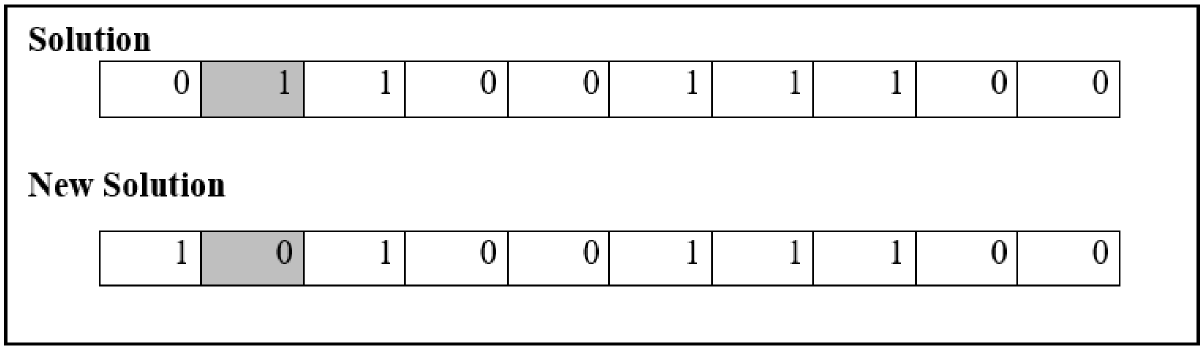


Figure 5.2 Flip neighbourhood operator

5.2 Experiments Results of IBBWO

Same as the BBWO method, the proposed method (IBBWO) was tested on twenty-eight UCI datasets using the same implementation steps and parameter settings as explained in the previous chapter (Chapter 4). In order to evaluate the performance of our proposed method (IBBWO) that obtained by combining the BBWO with HCA. We first compare the results of the IBBWO with BBWO as presented in section 5.2.1. Next, we compare IBBWO with the six up-to-date FS algorithms (BPSO, BMVO**,** BGWO, BMFO, BWOA, BBAT) as presented in section 5.2.2.

5.2.1 Results and Discussion 1

The following Table 5.1 shows the comparison between our two proposed methods: IBBWO and BBWO based on the two evaluation criteria: the classification accuracy (maximizing), and feature selected (minimizing).

Table 5.1 Comparison between (IBBWO and BBWO)

Datasets	Classification accuracy		Feature selected	
	IBBWO	BBWO	IBBWO	BBWO
Breastcancer	**0.98**	0.97	3.00	3.00
BreastEW	**0.95**	0.94	**4.60**	12.25
CongressEW	0.95	0.95	**1.50**	4.60
Exactly	**1**	0.91	5.25	**3.75**
Exactly2	0.77	0.77	**2.00**	3.65
HeartEW	**0.85**	0.84	**2.55**	3.80
IonosphereEW	**0.90**	0.88	**9.45**	13.75
Lymphography	0.85	0.85	**4.05**	6.80
M-of-n	**1**	0.95	**5.75**	7.00
PenglungEW	0.90	0.90	**100.85**	151.75
SonarEW	**0.87**	0.86	**14.75**	24.40
SpectEW	**0.82**	0.81	**6.50**	8.50
Tic-tac-toe	**0.82**	0.80	**4.05**	3.80
Vote	**0.95**	0.93	**1.55**	4.05
WaveformEW	0.88	0.88	**19.80**	20.60
Zoo	0.92	0.92	**4.60**	5.05
Parkinsons	0.90	0.90	**2.65**	7.70
Lungcancer	**0.92**	0.90	**4.70**	7.00
Colon	**0.89**	0.87	**888.35**	980.22
Leukemia	0.86	0.86	**3499.75**	3531.90
Dermatology	0.97	0.97	**11.25**	14.70
Semeion	**0.94**	0.93	**120.00**	130.00
Satellite	0.99	0.99	**4.40**	9.20
Spambase	0.93	0.93	**21.10**	28.60
Segment	0.96	0.96	**6.60**	7.70
Credit	0.79	0.79	**6.00**	7.22
KrvskpEW	**0.97**	0.95	**15.80**	19.00
Plants-100	**0.81**	0.80	**32.20**	32.50
Average	0.9050	0.8932	171.53	180.44
Rank	**1**	2	**1**	2

In Table 5.1, the best classification accuracy results are highlighted in bold. It can be seen that IBBWO outperforms BBWO in 15 datasets (Breastcancer, BreastEW, Exactly, HeartEW, IonosphereEW M-of-n, SonarEW, SpectEW, Tic-tac-toe, Vote, Lungcancer, Colon, Semeion, KrvskpEW, Plants-100), and same results in 13 datasets **(**CongressEW, Exactly2, Lymphography, PenglungEW, WaveformEW, Zoo, Parkinsons, Leukemia, Dermatology, Satellite, Spambase, Segment, Credit). As illustrated in Figure 5.3. The IBBWO has obtained better performance in terms of the average total classification accuracy of all datasets in comparison with the BBWO. These results proved that IBBWO is more efficient than BBWO in terms of maximizing classification accuracy.

With regards of the second criteria: the features selected; the results show that the IBBWO obtained better results than BBWO in 26 datasets (BreastEW, CongressEW, Exactly2, HeartEW, IonosphereEW, Lymphography, M-of-n, PenglungEW, SonarEW, SpectEW, Tic-tac-toe, Vote, WaveformEW, Zoo, Parkinsons, Lungcancer, Colon, Leukemia, Dermatology, Semeion, Satellite ,Spambase, Segment, Credit, KrvskpEW, Plants-100), same result in 1 dataset (Breastcancer), and worse result in 1 dataset (Exactly). The IBBWO shows the efficiency by minimizing the average of the total feature selected of all datasets in comparison with BBWO. The result is shown in Figure 5.4.

Overall, The capability of the proposed methods (BBWO and IBBWO) had been tested on twenty-eight regular benchmark datasets and from a comparison of the results it has been proven that (IBBWO) algorithm is capable of producing very good results and that it is proven better than the original method (BBWO) on most tested datasets for solving FS problems. This means that the IBBWO (BBWO with HCA) improves the performance of the BBWO and obtained very good results.

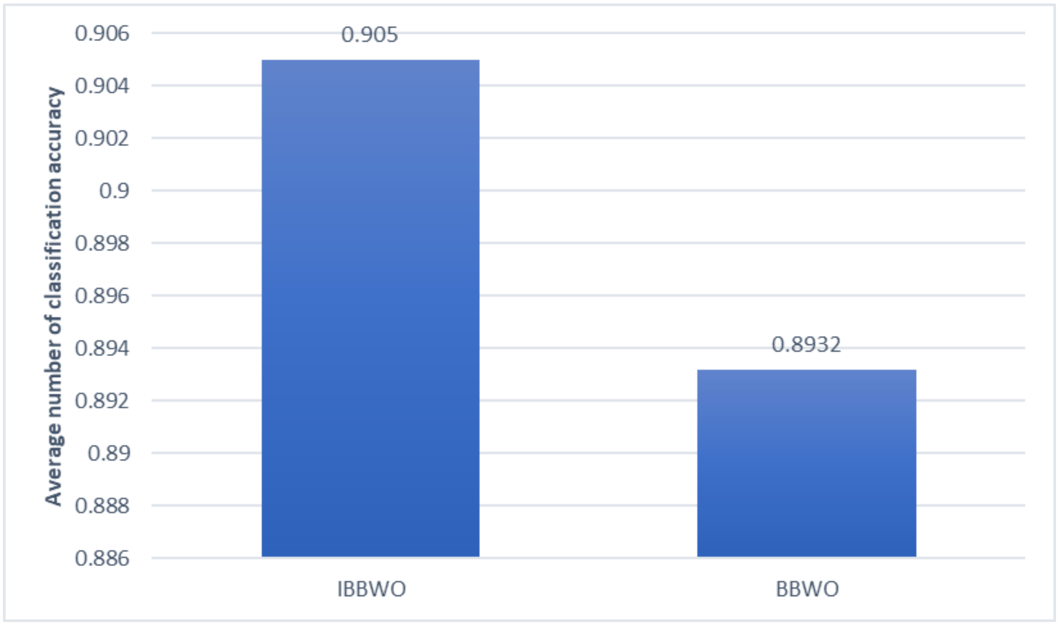


Figure 5.3 Average number of classification accuracy of (IBBWO, BBWO) (Maximizing)

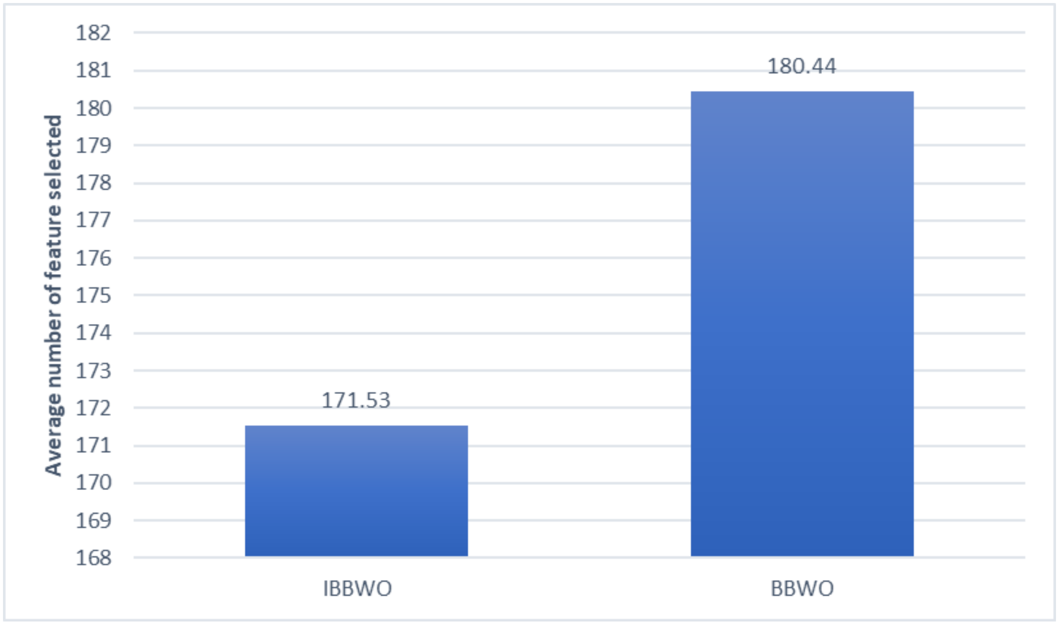


Figure 5.4 Average number of features selected of (IBBWO, BBWO) (Minimizing)

5.2.2 Results and Discussion 2

Same as results and comparisons of BBWO in Chapter 4, we compared IBBWO with the six up-to-date FS algorithms (BPSO, BMVO**,** BGWO, BMFO, BWOA, BBAT) based on the two evaluation criteria: the classification accuracy (maximizing), and feature selected (minimizing), as illustrated in Table 5.2 and Table 5.3 respectively.

Table 5.2 Comparison IBBWO with all algorithms based on the classification accuracy

Datasets Name	IBBWO	BPSO	BMVO	BGWO	BMFO	BWOA	BBAT
Breastcancer	**0.98**	0.96	0.97	0.96	0.97	0.97	0.96
BreastEW	**0.95**	0.94	0.94	0.95	0.94	0.93	0.94
CongressEW	**0.95**	0.92	0.95	0.95	0.95	0.95	0.94
Exactly	**1**	0.76	0.89	0.74	0.90	0.91	0.73
Exactly2	**0.77**	0.77	0.76	0.75	0.76	0.74	0.74
HeartEW	**0.85**	0.81	0.85	0.84	0.85	0.85	0.82
IonosphereEW	**0.90**	0.86	0.88	0.88	0.88	0.88	0.88
Lymphography	**0.85**	0.82	0.84	0.82	0.85	0.83	0.81
M-of-n	**1**	0.83	0.99	0.88	0.98	0.98	0.81
PenglungEW	**0.90**	0.87	0.89	0.89	0.89	0.88	0.88
SonarEW	**0.87**	0.86	0.87	0.86	0.86	0.87	0.86
SpectEW	**0.82**	0.81	0.81	0.82	0.82	0.81	0.81
Tic-tac-toe	**0.82**	0.74	0.81	0.78	0.82	0.81	0.76
Vote	**0.95**	0.91	0.94	0.94	0.94	0.94	0.93
WaveformEW	**0.88**	0.86	0.88	0.87	0.88	0.88	0.83
Zoo	**0.92**	0.89	0.89	0.88	0.88	0.90	0.89
Parkinsons	**0.90**	0.88	0.89	0.86	0.89	0.89	0.88
Lungcancer	**0.92**	0.88	0.91	0.90	0.91	0.91	0.90
Colon	**0.89**	0.86	0.89	0.87	0.87	0.87	0.87
Leukemia	**0.86**	0.83	0.86	0.85	0.86	0.86	0.85
Dermatology	**0.97**	0.89	0.96	0.95	0.97	0.97	0.92
Semeion	**0.94**	0.92	0.93	0.92	0.93	0.93	0.92
Satellite	0.99	0.99	0.99	0.99	0.99	0.99	0.99
Spambase	**0.93**	0.88	0.93	0.92	0.93	0.93	0.89
Segment	**0.96**	0.94	0.96	0.96	0.96	0.96	0.94
Credit	**0.79**	0.76	0.79	0.77	0.78	0.79	0.78
KrvskpEW	**0.97**	0.90	0.96	0.95	0.97	0.97	0.87
Plants-100	**0.81**	0.78	0.79	0.78	0.80	0.79	0.77
Average	0.9050	0.8614	0.8935	0.8760	0.8939	0.8925	0.8632
Rank	**1**	7	3	5	2	4	6

Table 5.3 Comparison IBBWO all algorithms based on the features selected

Datasets	IBBWO	BPSO	BMVO	BGWO	BMFO	BWOA	BBAT
Breastcancer	**3.00**	3.40	4.55	5.15	4.35	4.60	3.45
BreastEW	**4.60**	11.40	10.95	13.55	13.20	12.12	13.15
CongressEW	**1.50**	5.20	4.25	5.95	5.40	4.20	5.55
Exactly	**5.25**	5.30	7.25	7.05	7.05	6.65	5.80
Exactly2	**2.00**	3.95	2.33	5.40	3.15	2.10	3.50
HeartEW	**2.55**	4.25	3.45	4.15	3.70	3.45	4.65
IonosphereEW	**9.45**	14.35	13.35	15.90	15.00	12.55	16.55
Lymphography	**4.05**	7.60	6.66	7.56	7.35	6.35	7.55
M-of-n	5.75	**5.50**	7.22	8.00	6.75	7.35	6.15
PenglungEW	**100.85**	154.80	152.35	155.20	152.45	146.35	156.85
SonarEW	**14.75**	27.05	25.55	28.05	28.95	23.85	27.00
SpectEW	**6.50**	8.95	8.22	10.15	8.20	8.75	9.85
Tic-tac-toe	**4.05**	4.20	4.55	4.55	4.41	4.05	4.28
Vote	**1.55**	5.05	4.95	6.35	5.85	4.50	6.15
WaveformEW	19.80	22.00	22.15	21.45	21.35	**19.45**	20.25
Zoo	**4.60**	5.59	6.35	6.65	6.13	5.75	6.50
Parkinsons	**2.65**	8.00	8.45	9.15	9.10	8.20	9.25
Lungcancer	**4.70**	7.25	8.66	9.35	8.90	8.05	8.95
Colon	**888.35**	961.65	963.25	965.55	962.15	943.55	963.35
Leukemia	**3499.75**	3555.82	3571.85	3535.85	3534.55	3511.35	3513.50
Dermatology	**11.25**	16.85	15.95	16.70	16.60	16.45	16.50
Semeion	**120.00**	131.85	127.00	128.6	131.60	126.80	126.70
Satellite	**4.40**	13.01	10.55	12.40	11.40	10.10	12.45
Spambase	**21.10**	29.77	26.55	30.50	26.25	26.50	27.25
Segment	**6.60**	8.72	9.25	9.90	9.95	8.90	9.60
Credit	**6.00**	8.41	7.95	8.50	8.30	7.72	8.75
KrvskpEW	**15.80**	19.73	19.81	21.30	18.56	17.92	18.45
Plants-100	**32.20**	35.55	33.15	33.80	33.32	34.15	35.50
Average	171.53	181.61	181.66	181.66	180.85	178.27	180.26
Rank	**1**	5	6	6	4	2	3

From the results given in Table 4.10, the proposed method (IBBWO) is comparable with the six FS (BPSO, BMVO**,** BGWO, BMFO, BWOA, BBAT) algorithms based on the classification accuracy of the twenty-eight datasets. We can make the following conclusion:

- IBBWO produced better results than BPSO in 26 datasets (Breastcancer, BreastEW, CongressEW, Exactly, HeartEW, IonosphereEW, Lymphography, M-of-n, PenglungEW, SonarEW, SpectEW, Tic-tac-toe, Vote, WaveformEW, Zoo, Parkinsons, Lungcancer, Colon, Leukemia, Dermatology, Semeion, Spambase, Segment, Credit, KrvskpEW, Plants-100), and same results in 2 datasets (Exactly2, Satellite).
- IBBWO produced better results than BMVO in 18 datasets (Breastcancer, BreastEW, Exactly, Exactly2, IonosphereEW, Lymphography, M-of-n, PenglungEW, SpectEW, Tic-tac-toe, Vote, Zoo, Parkinsons, Lungcancer, Dermatology, Semeion, KrvskpEW, Plants-100), and same results in 10 datasets (CongressEW, HeartEW, SonarEW, WaveformEW, Colon, Leukemia, Satellite ,Spambase, Segment, Credit).
- IBBWO produced better results than BGWO in 23 datasets (Breastcancer, Exactly, Exactly2, HeartEW, IonosphereEW, Lymphography, M-of-n, PenglungEW, SonarEW, Tic-tac-toe, Vote, WaveformEW, Zoo, Parkinsons, Lungcancer, Colon, Leukemia, Dermatology, Semeion, Spambase, Credit, KrvskpEW, Plants-100), and same results in 5 datasets (BreastEW, CongressEW, SpectEW, Satellite, Segment).
- IBBWO produced better results than BMFO in 16 datasets (Breastcancer, BreastEW, Exactly, Exactly2, IonosphereEW, M-of-n, PenglungEW, SonarEW, Vote, Zoo, Parkinsons, Lungcancer, Colon, Semeion, Credit, Plants-100), and same results in 12 datasets (CongressEW, HeartEW, Lymphography, SpectEW, Tic-tac-toe, WaveformEW, Leukemia, Dermatology, Satellite ,Spambase, Segment, KrvskpEW).

- IBBWO produced better results than BWOA in 17 datasets (Breastcancer, BreastEW, Exactly, Exactly2, IonosphereEW, Lymphography, M-of-n, PenglungEW, SpectEW, Tic-tac-toe, Vote, Zoo, Parkinsons, Lungcancer, Colon, Semeion, Plants-100), and same results in 11 datasets (CongressEW, HeartEW, SonarEW**,** WaveformEW, Leukemia, Dermatology, Satellite, Spambase, Segment, Credit, KrvskpEW).
- IBBWO produced better results than BBAT in 27 datasets (Breastcancer, BreastEW, CongressEW, Exactly, Exactly2, HeartEW, IonosphereEW, Lymphography, M-of-n, PenglungEW, SonarEW, SpectEW, Tic-tac-toe, Vote, WaveformEW, Zoo, Parkinsons, Lungcancer, Colon, Leukemia, Dermatology, Semeion, Spambase, Segment, Credit, KrvskpEW, Plants-100), and same result in 1 dataset (Satellite).

With regard to the second criterion, the average number of features selected (Minimizing) as presented in Table 5.3, and from the best results are highlighted in bold, we can see the performance of the proposed IBBWO outperform all the six FS algorithms (BPSO, BMVO**,** BGWO, BMFO, BWOA, BBAT) in term of the minimizing the number of features selected of all twenty-eight tested datasets.

Overall, the result showed that the proposed IBBWO is better than other algorithms in many datasets and has obtained the best performance in terms of the average of the total classification accuracy, and minimizing the feature selected in comparison with the six up-to-date FS algorithms (BPSO, BMVO**,** BGWO, BMFO, BWOA, BBAT) of all twenty-eight datasets as illustrated in Figure 5.5, Figure 5.6. This indicates that the IBBWO is a good and effective algorithm for solving the FS problems.

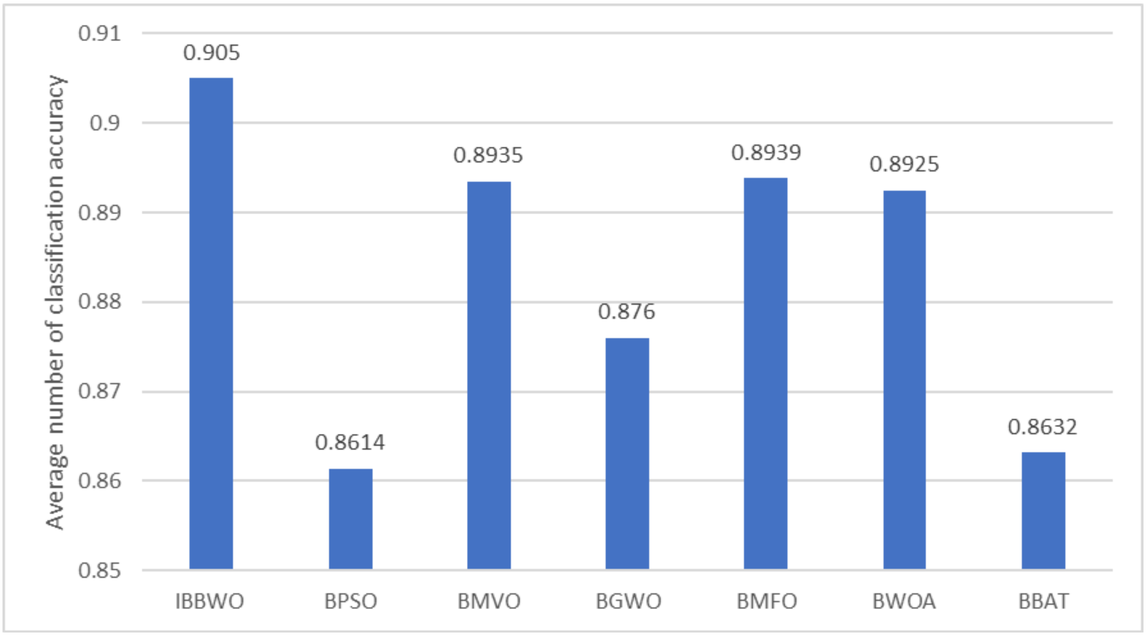


Figure 5.5 Average number of classification accuracy of all algorithms (Maximizing)

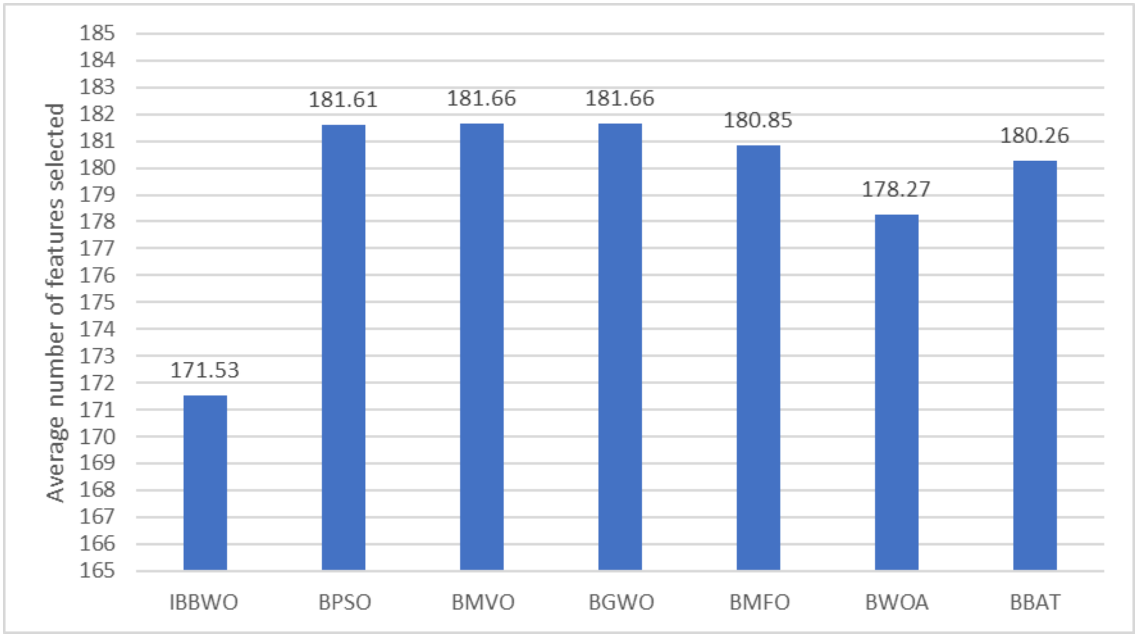


Figure 5.6 Average number of features selected of all algorithms (Minimizing)

5.3 Summary

This chapter proposed an improvement to the BBWO, that aims at enhancing the exploitation of BBWO by focusing the search on a certain area in the solution space. The proposed method (IBBWO) is combined the BBWO with HCA, and it was applied to the public dataset provided by UCI. The results were then compared with the basic BBWO and the six up-to-date FS algorithms (BPSO, BMVO, BGWO, BMFO, BWOA, BBAT). The results showed that IBBWO produced better results than the basic BBWO and the six FS algorithms on many datasets, which is proved that IBBWO is one of the most powerful FS algorithms.

Chapter 6

Conclusion and Future Work

The key points of the suggested strategies from this thesis shall now be summarized in this chapter. Firstly, the summary of the research is presented in Section 6.1, and the conclusion is presented in Section 6.2. Followed by the contributions in section 6.3. Finally, opportunities for further research will be proposed in Section 6.4.

6.1 Research Summary

This research was written in the hope of generating effective strategies for FS problems in terms of producing good quality solutions. The research commenced with the overview of the primary research objective (to investigate feature selection problems using the modified a Binary Black Widow Optimization (BBWO) algorithm and to generate a combined hill-climbing algorithm with BBWO to improve the usefulness of the BBWO, which were formed in order to formulate methods for discovering the minimal and most useful features, while retaining enough information). In the second chapter of this thesis, the FS problems were described, along with the literature review. Following this, the chosen methodology for the study was outlined in Chapter 3, and the newly generated a Black Widow Optimization algorithm for FS problems was introduced in Chapter 4. This was applied in order to evaluate the usefulness of solutions using the wrapper FS method based on the KNN classifier. Additionally, an improvement to the BBWO for FS problems (IBBWO) is presented in Chapter 5.

6.2 Conclusion

FS is acknowledged to be a particularly daunting challenge in data mining and has been classified as an NP-hard problem. Recently, many metaheuristic algorithms (MAs) have been applied to enhance FS and thus minimize the number of features while also maintaining a track record of highly accurate results. The idea of the BWO as an optimization algorithm was derived from nature, in essence, by the singular mating behaviour exhibited by black widow spiders, a process that includes an exclusive stage in the natural world: cannibalism [19]. To investigate the performance and suitability of the BWO for FS problems, we proposed the modified Binary Black Widow Optimization (BBWO). As a part of this research, the performance of the BBWO was tested on twenty-eight datasets from the UCI repository and the results were compared with six up-to-date FS algorithms (BPSO, BMVO**,** BGWO, BMFO, BWOA, BBAT). The results showed that BBWO produced very good results, and in some cases, were competitive with the best-known results. The promising results of the BBWO in terms of performance gave us confidence in suggesting improvements to the BBWO by combining it with HCA to further maximize the performance. This improvement method (IBBWO) was also tested on twenty-eight datasets from the UCI repository. The results are then compared with the basic BBWO and the six up-to-date FS algorithms. Results showed that the (IBBWO) produced better results than the basic BBWO and up-to-date FS algorithms.

Overall, the findings of this study have revealed that the combined method proposed in this thesis (IBBWO) shows the greatest flexibility in terms of potential solutions when compared to previous FS methods and has the capacity to achieve the most useful outcomes on most FS datasets.

6.3 Contributions

In this thesis, the most important contributions are described below.

1- We have investigated the ability of the Black Widow Optimization algorithm to solve FS problems. The results showed that it produced good results, and, in some cases, these results were competitive with the best-known results. However, the results also revealed that the algorithm faced the problem of slow convergence due to the use of a population of solutions and lack of local exploitation.

2- We have proposed an improvement to the BBWO. The proposed method combined the BBWO with HCA in order to improve the exploitation process of the BBWO. The experiment results showed that IBBWO can produce better results than the basic BBWO and up-to-date FS algorithms.

3- We introduced two competitive methods (BBWO and IBBWO) that obtained good quality solutions compared to up-to-date FS algorithms.

6.4 Future Works

Our future works are as follows:

1- In this thesis, we improve the basic BBWO by combining it with HCA in order to improve the exploitation process. We would also like to combine the proposed method with other metaheuristics algorithms to further maximize the performance.

2- The proposed algorithms showed high accuracy relative to the other latest FS algorithms when we tested them on various UCI datasets. We would also like to verify the precision of the data by determining the deviation of data points from each other.

3- The proposed algorithms can also be further improved by using parameter adaptation schemes or investigating different population generation methods.

4- The BBWO could be applied to various other areas of study to solve many other real-world problems such as text mining, clustering, image processing, and routing problems.

References

[1] Sun, Lin, Lanyingying Wang, Weiping Ding, Yuhua Qian, and Jiucheng Xu. "Feature Selection Using Fuzzy Neighborhood Entropy-Based Uncertainty Measures for Fuzzy Neighborhood Multigranulation Rough Sets." *IEEE Transactions on Fuzzy Systems* (2020).

[2] Aljarah, Ibrahim, Maria Habib, Hossam Faris, Nailah Al-Madi, Ali Asghar Heidari, Majdi Mafarja, Mohamed Abd Elaziz, and Seyedali Mirjalili. "A dynamic locality multi-objective salp swarm algorithm for feature selection." *Computers & Industrial Engineering* 147 (2020): 106628.

[3] Venkatesh, B., and J. Anuradha. "A review of feature selection and its methods." *Cybernetics and Information Technologies* 19, no. 1 (2019): 3-26.

[4] Hammouri, Abdelaziz I., Majdi Mafarja, Mohammed Azmi Al-Betar, Mohammed A. Awadallah, and Iyad Abu-Doush. "An improved Dragonfly Algorithm for feature selection." *Knowledge-Based Systems* 203 (2020): 106131.

[5] Arora, Sankalap, and Priyanka Anand. "Binary butterfly optimization approaches for feature selection." *Expert Systems with Applications* 116 (2019): 147-160.

[6] Alweshah, Mohammed, Saleh Alkhalaileh, Dheeb Albashish, Majdi Mafarja, Qusay Bsoul, and Osama Dorgham. "A hybrid mine blast algorithm for feature selection problems." *Soft Computing* (2020): 1-18.

[7] Abdel-Basset, Mohamed, Doaa El-Shahat, Ibrahim El-henawy, Victor Hugo C. de Albuquerque, and Seyedali Mirjalili. "A new fusion of grey wolf optimizer algorithm with a two-phase mutation for feature selection." *Expert Systems with Applications* 139 (2020): 112824.

[8] Sharma, Manik, and Prableen Kaur. "A Comprehensive Analysis of Nature-Inspired Meta-Heuristic Techniques for Feature Selection Problem." *Archives of Computational Methods in Engineering* (2020): 1-25.

[9] Too, Jingwei, and Seyedali Mirjalili. "A Hyper Learning Binary Dragonfly Algorithm for Feature Selection: A COVID-19 Case Study." *Knowledge-Based Systems* (2020): 106553.

[10] Han, Jiawei, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.

[11] Liu, Huan, and Rudy Setiono. "Chi2: Feature selection and discretization of numeric attributes." In *Proceedings of 7th IEEE International Conference on Tools with Artificial Intelligence*, pp. 388-391. IEEE, 1995.

[12] Emary, Eid, Hossam M. Zawbaa, and Aboul Ella Hassanien. "Binary grey wolf optimization approaches for feature selection." *Neurocomputing* 172 (2016): 371-381.

[13] Neggaz, Nabil, Essam H. Houssein, and Kashif Hussain. "An efficient henry gas solubility optimization for feature selection." *Expert Systems with Applications* (2020): 113364.

[14] Taradeh, Mohammad, Majdi Mafarja, Ali Asghar Heidari, Hossam Faris, Ibrahim Aljarah, Seyedali Mirjalili, and Hamido Fujita. "An evolutionary gravitational search-based feature selection." *Information Sciences* 497 (2019): 219-239.

[15] Kennedy, James, and Russell Eberhart. "Particle swarm optimization." In *Proceedings of ICNN'95-international conference on neural networks*, vol. 4, pp. 1942-1948. IEEE, 1995.

[16] Emary, Eid, Hossam M. Zawbaa, and Aboul Ella Hassanien. "Binary ant lion approaches for feature selection." *Neurocomputing* 213 (2016): 54-65.

[17] Mafarja, Majdi, and Seyedali Mirjalili. "Whale optimization approaches for wrapper feature selection." *Applied Soft Computing* 62 (2018): 441-453.

[18] Tawhid, Mohamed A., and Abdelmonem M. Ibrahim. "Feature selection based on rough set approach, wrapper approach, and binary whale optimization algorithm." *International Journal of Machine Learning and Cybernetics* 11, no. 3 (2020): 573-602.

[19] Hayyolalam, Vahideh, and Ali Asghar Pourhaji Kazem. "Black widow optimization algorithm: A novel meta-heuristic approach for solving engineering optimization problems." *Engineering Applications of Artificial Intelligence* 87 (2020): 103249.

[20] Mafarja, Majdi, Iyad Jaber, Sobhi Ahmed, and Thaer Thaher. "Whale optimisation algorithm for high-dimensional small-instance feature selection." *International Journal of Parallel, Emergent and Distributed Systems* (2019): 1-17.

[21] Shukla, Alok Kumar. "Multi-population adaptive genetic algorithm for selection of microarray biomarkers." *Neural Computing and Applications* (2019): 1-22.

[22] Jensen, Richard, and Qiang Shen. "Semantics-preserving dimensionality reduction: rough and fuzzy-rough-based approaches." *IEEE Transactions on knowledge and data engineering* 16, no. 12 (2004): 1457-1471.

[23] Dorigo, Marco, Mauro Birattari, and Thomas Stutzle. "Ant colony optimization." *IEEE computational intelligence magazine* 1, no. 4 (2006): 28-39.

[24] Yang, Xin-She. "A new metaheuristic bat-inspired algorithm." In *Nature inspired cooperative strategies for optimization (NICSO 2010)*, pp. 65-74. Springer, Berlin, Heidelberg, 2010.

[25] Fayyad, Usama, Gregory Piatetsky-Shapiro, and Padhraic Smyth. "From data mining to knowledge discovery in databases." *AI magazine* 17, no. 3 (1996): 37-37.

[26] Maimon, Oded, and Lior Rokach, eds. "Data mining and knowledge discovery handbook." (2005).

[27] Fayyad, Usama. "Data mining and knowledge discovery in databases: implications for scientific databases." In *Proceedings. Ninth International Conference on Scientific and Statistical Database Management (Cat. No. 97TB100150)*, pp. 2-11. IEEE, 1997.

[28] Tsai, Chih-Fong, and Yu-Chi Chen. "The optimal combination of feature selection and data discretization: An empirical study." *Information Sciences* 505 (2019): 282-293.

[29] Stańczyk, Urszula, and Beata Zielosko. "Heuristic-based feature selection for rough set approach." *International Journal of Approximate Reasoning* 125 (2020): 187-202.

[30] Mafarja, Majdi, and Salwani Abdullah. "A fuzzy record-to-record travel algorithm for solving rough set attribute reduction." *International Journal of Systems Science* 46, no. 3 (2015): 503-512.

[31] Bhattacharyya, Trinav, Bitanu Chatterjee, Pawan Kumar Singh, Jin Hee Yoon, Zong Woo Geem, and Ram Sarkar. "Mayfly in harmony: A new hybrid meta-heuristic feature selection algorithm." *IEEE Access* 8 (2020): 195929-195945.

[32] Sreejith, S., H. Khanna Nehemiah, and A. Kannan. "Clinical data classification using an enhanced SMOTE and chaotic evolutionary feature selection." *Computers in Biology and Medicine* 126 (2020): 103991.

[33] Sayed, Gehad Ismail, Ashraf Darwish, and Aboul Ella Hassanien. "A new chaotic whale optimization algorithm for features selection." *Journal of classification* 35, no. 2 (2018): 300-344.

[34] Rong, Miao, Dunwei Gong, and Xiaozhi Gao. "Feature selection and its use in big data: challenges, methods, and trends." *IEEE Access* 7 (2019): 19709-19725.

[35] Dash, Manoranjan, and Huan Liu. "Feature selection for classification." *Intelligent data analysis* 1, no. 1-4 (1997): 131-156.

[36] Khurmaa, Ruba Abu, Ibrahim Aljarah, and Ahmad Sharieh. "An intelligent feature selection approach based on moth flame optimization for medical diagnosis." *Neural Computing and Applications* (2020): 1-40.

[37] Mafarja, Majdi, Asma Qasem, Ali Asghar Heidari, Ibrahim Aljarah, Hossam Faris, and Seyedali Mirjalili. "Efficient hybrid nature-inspired binary optimizers for feature selection." *Cognitive Computation* 12, no. 1 (2020): 150-175.

[38] Liu, Huan, and Hiroshi Motoda. *Feature selection for knowledge discovery and data mining*. Vol. 454. Springer Science & Business Media, 2012.

[39] Talbi, El-Ghazali. *Metaheuristics: from design to implementation*. Vol. 74. John Wiley & Sons, 2009.

[40] Dokeroglu, Tansel, Ender Sevinc, Tayfun Kucukyilmaz, and Ahmet Cosar. "A survey on new generation metaheuristic algorithms." *Computers & Industrial Engineering* 137 (2019): 106040.

[41] Chandrashekar, Girish, and Ferat Sahin. "A survey on feature selection methods." *Computers & Electrical Engineering* 40, no. 1 (2014): 16-28.

[42] Binitha, S., and S. Siva Sathya. "A survey of bio inspired optimization algorithms." *International journal of soft computing and engineering* 2, no. 2 (2012): 137-151.

[43] El Aboudi, Naoual, and Laila Benhlima. "Review on wrapper feature selection approaches." In *2016 International Conference on Engineering & MIS (ICEMIS)*, pp. 1-5. IEEE, 2016.

[44] Bolón-Canedo, Verónica, Noelia Sánchez-Maroño, and Amparo Alonso-Betanzos. "A review of feature selection methods on synthetic data." *Knowledge and information systems* 34, no. 3 (2013): 483-519.

[45] Brijain, Mr, R. Patel, M. R. Kushik, and K. Rana. "A survey on decision tree algorithm for classification." (2014).

[46] Shaban, Warda M., Asmaa H. Rabie, Ahmed I. Saleh, and M. A. Abo-Elsoud. "A new COVID-19 Patients Detection Strategy (CPDS) based on hybrid feature selection and enhanced KNN classifier." *Knowledge-Based Systems* 205 (2020): 106270.

[47] Polat, Huseyin, Homay Danaei Mehr, and Aydin Cetin. "Diagnosis of chronic kidney disease based on support vector machine by feature selection methods." *Journal of medical systems* 41, no. 4 (2017): 55.

[48] Dy, Jennifer G. "Unsupervised feature selection." *Computational methods of feature selection* (2008): 19-39.

[49] Xin-She, Y. "An introduction with metaheuristic applications." *Engineering Optimization* (2010).

[50] Pham, Duc, and Dervis Karaboga. *Intelligent optimisation techniques: genetic algorithms, tabu search, simulated annealing and neural networks*. Springer Science & Business Media, 2012.

[51] Whitley, Darrell. "A genetic algorithm tutorial." *Statistics and computing* 4, no. 2 (1994): 65-85.

[52] Mladenović, Nenad, and Pierre Hansen. "Variable neighborhood search." *Computers & operations research* 24, no. 11 (1997): 1097-1100.

[53] Kirkpatrick, Scott, C. Daniel Gelatt, and Mario P. Vecchi. "Optimization by simulated annealing." *science* 220, no. 4598 (1983): 671-680.

[54] Kuo, Yiyo. "Using simulated annealing to minimize fuel consumption for the time-dependent vehicle routing problem." *Computers & Industrial Engineering* 59, no. 1 (2010): 157-165.

[55] Zhang, Defu, Yongkai Liu, Rym M'Hallah, and Stephen CH Leung. "A simulated annealing with a new neighborhood structure-based algorithm for high school timetabling problems." *European Journal of Operational Research* 203, no. 3 (2010): 550-558.

[56] Gendreau, Michel, and Jean-Yves Potvin, eds. *Handbook of metaheuristics*. Vol. 2. New York: Springer, 2010.

[57] Jihad, SaifKifah, and Salwani Abdullah. "Investigating composite neighbourhood structure for attribute reduction in rough set theory." In *2010 10th International Conference on Intelligent Systems Design and Applications*, pp. 1183-1188. IEEE, 2010.

[58] Ke, Liangjun, Zuren Feng, and Zhigang Ren. "An efficient ant colony optimization approach to attribute reduction in rough set theory." *Pattern Recognition Letters* 29, no. 9 (2008): 1351-1357.

[59] Hoos, Holger H., and Thomas Stützle. *Stochastic local search: Foundations and applications*. Elsevier, 2004.

[60] Too, Jingwei, and Abdul Rahim Abdullah. "A new and fast rival genetic algorithm for feature selection." *The Journal of Supercomputing* (2020): 1-31.

[61] Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in engineering software*, *95*, 51-67.

[62] Mafarja, Majdi M., and Seyedali Mirjalili. "Hybrid whale optimization algorithm with simulated annealing for feature selection." *Neurocomputing* 260 (2017): 302-312.

[63] Mafarja, Majdi, Radi Jarrar, Sobhi Ahmad, and Ahmed A. Abusnaina. "Feature selection using binary particle swarm optimization with time varying inertia weight strategies." In *Proceedings of the 2nd International Conference on Future Networks and Distributed Systems*, pp. 1-9. 2018.

[64] Mirjalili, Seyedali, Seyed Mohammad Mirjalili, and Andrew Lewis. "Grey wolf optimizer." *Advances in engineering software* 69 (2014): 46-61.

[65] Mirjalili, Seyedali. "Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm." *Knowledge-based systems* 89 (2015): 228-249.

[66] Zawbaa, Hossam M., Eid Emary, Bazil Parv, and Marwa Sharawi. "Feature selection approach based on moth-flame optimization algorithm." In *2016 IEEE congress on evolutionary computation (CEC)*, pp. 4612-4617. IEEE, 2016.

[67] Mirjalili, Seyedali, Seyed Mohammad Mirjalili, and Abdolreza Hatamlou. "Multi-verse optimizer: a nature-inspired algorithm for global optimization." *Neural Computing and Applications* 27, no. 2 (2016): 495-513.

[68] Al-Madi, Nailah, Hossam Faris, and Seyedali Mirjalili. "Binary multi-verse optimization algorithm for global optimization and discrete problems." *International Journal of Machine Learning and Cybernetics* 10, no. 12 (2019): 3445-3465.

[69] Mirjalili, Seyedali, Seyed Mohammad Mirjalili, and Xin-She Yang. "Binary bat algorithm." *Neural Computing and Applications* 25, no. 3 (2014): 663-681.

[70] https://archive.ics.uci.edu/ml/datasets.php

[71] Ayyad, Sarah M., Ahmed I. Saleh, and Labib M. Labib. "Gene expression cancer classification using modified K-Nearest Neighbors technique." *Biosystems* 176 (2019): 41-51.

[72] Dordaie, N. and Navimipour, N.J., 2018. A hybrid particle swarm optimization and hill climbing algorithm for task scheduling in the cloud environments. *ICT Express*, *4*(4), pp.199-202.

[73] Dueck, Gunter. "New optimization heuristics: The great deluge algorithm and the record-to-record travel." *Journal of Computational physics* 104, no. 1 (1993): 86-92.

[74] Glover, Fred, and Manuel Laguna. "Tabu search." In *Handbook of combinatorial optimization*, pp. 2093-2229. Springer, Boston, MA, 1998.

[75] Martí, R., Pardalos, P. M., & Resende, M. G. C. (2018). *Handbook of Heuristics* (1st ed. 2018.). Springer International Publishing. https://doi.org/10.1007/978-3-319-07124-4

[76] Mafarja, Majdi, Ibrahim Aljarah, Ali Asghar Heidari, Hossam Faris, Philippe Fournier-Viger, Xiaodong Li, and Seyedali Mirjalili. "Binary dragonfly optimization for feature selection using time-varying transfer functions." *Knowledge-Based Systems* 161 (2018): 185-204.

[77] Faris, Hossam, Ibrahim Aljarah, Seyedali Mirjalili, Pedro A. Castillo, and Juan Julián Merelo Guervós. "EvoloPy: An Open-source Nature-inspired Optimization Framework in Python." In *IJCCI (ECTA)*, pp. 171-177. 2016.