People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

University of Ferhat Abbas Setif -1-

Thesis

to obtain the title of

PhD of LMD

Specialty : COMPUTER SCIENCE

Defended by

Adel GOT

Machine Learning using Multi-Objective Evolutionary Algorithms

Jury :

President :	Mohammed SAIDI	MCA	University of Ferhat Abbas Setif -1-
Advisors :	Abdelouahab MOUSSAOUI	Pr.	University of Ferhat Abbas Setif -1-
	Djaafar ZAOUACHE	MCA	University of Bordj Bou Arreridj
Examinators :	Sadik BESSOU	MCA	University of Ferhat Abbas Setif -1-
	Bilal SAOUD	MCA	University of Bouira

République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université de Ferhat Abbas Sétif -1-

Thèse

Présentée à la Faculté des Sciences
Département d'Informatique
En vue de l'obtention du diplôme de

Doctorat LMD

Option : INFORMATIQUE

Par

Adel GOT

Thème

Apprentissage Automatique par Algorithmes Evolutionnaires Multiobjectifs

Devant le jury composé de :

Président :	Mohammed SAIDI	MCA	Université de Ferhat Abbas Sétif -1-
Rapporteurs :	Abdelouahab MOUSSAOUI	Pr.	Université de Ferhat Abbas Sétif -1-
	Djaafar ZAOUACHE	MCA	Université de Bordj Bou Arreridj
Examinateurs :	Sadik BESSOU	MCA	Université de Ferhat Abbas Sétif -1-
	Bilal SAOUD	MCA	Université de Bouira

Acknowledgments

I would like to express my deep gratitude to my supervisors, Pr. Abdelouahab MOUSSAOUI, for his invaluable advice, time and support throughout this research work. This thesis would not have been possible without his encouragement, motivation, inspiration, and guidance.

I would like to express the appreciation to my co-supervisors, Dr. Djaafar ZAOUACHE, for his guidance, encouragement, and his constant support throughout the course of this work.

I would like to thank Dr. SAIDI MOHAMMED, Dr. Sadick BESSOU, and Dr. Bilal SAOUD for accepting to be members of the examination committee for this thesis and for taking time to read and review it. Their suggestions will be taken into account and will surely significantly improve the final version.

Last but not least, I wish to thank my mother for his love, encouragement and support.

Contents

List of Algorithms

List of Figures

List of Tables

CHAPTER 1

Introduction

Contents

1.1 Overview

The present chapter introduces this thesis. It starts with a brief overview in which the relationship between machine learning and optimization problem is clarified, consequently, some multiobjective optimization methods for machine learning will be presented. Then, it outlines the motivations, the major contributions and the organization of the thesis.

1.1.1 From machine learning towards optimization problem

Historically, the term of machine learning has been known as the development of computers (machines in broader sense) able to learn and reasone automatically. That is to say, able to improve without the intervention of human. This ability for self-learning is done through experience and analytical observation. In [Nguyen 2008] denoted "*Broadly speaking, machine learning is the process of finding and describing structural patterns in a supplied dataset*".

Machine learning algorithm is the tool that helps machines to evolve automatically and accomplish complex tasks that cannot done with classical algorithms. A machine learning algorithm can be defined as the process which takes as input a

dataset (also known as sample learning or examples) and provides as output the description of the knowledge that has been learnt. Generally speaking, machine learning algorithms can be classified into two main categories: unsupervised (often known as clustering) and supervised learning (often known as classification). The first category consists of finding patterns in the data above and beyond what would be considered pure unstructured noise [Ghahramani 2003]. In the second category, the aim is to create knowledge structures that predict and classfiy unknown information into predefined classes [Nguyen 2008]. Usually, the result of any classification algorithm can be viewed as a function (also called model) which approximate the mapping between the output and a new input dataset. So, the objective is to build a learning model that enable to a machine to predict the appropriate output of dataset without being explicitly programmed. However, the approximation quality can be affected by several factors. Indeed, the presence of irrelevant and redundant features results in poor accuracy of the model. Consequently, it is often desirable to identify the most informative features - a process known as feature selection (also known as dimensionality reduction).

Feature selection is considered as one of the major key in machine learning. The feature space usually contains a large number of features which can be divided into relevant/irrevelant features and redundant/non-redundant features. Irrelevant and redundant features are not useful as they affect the performance of a classification/learning algorithm [Russell 2003]. Therefore, feature selection technique is invoked to select the subset of relevant features in order to: decrease the dimensionality of the feature space, simplify the learning model, improve the classification performance and reduce the running time [Dash 1997]; [Liu 1998]. However, the high dimensionality of feature space imply high computational time. For n features, there are 2^n subsets possible. In addition, due to the absence of prior knowledge about the features and the complex interaction between them, the feature selection process is considered as an NP-hard problem.

Although there exist several feature subsets that can reduce the dimensionality of the search space, but the optimal subset is the one which contains smaller number of features and achieved better classification performance [Xue 2014b]. From broadly point of view, two main challenges are taken into account in order to select the optimal feature subset [Xue 2014a]. The first goal is to reduce as well as possible the size of feature space, and the second goal is to increase as well as possible the classification accuracy. Accordingly, feature selection is considered as a multiobjective optimization problem in which we attempt to minimize the number of features and maximize the classification performance. Further, these two criteria (number/performance) are often conflicting to each other and they

require simultaneous optimization.

Feature selection algorithms are usually grouped into two main categories: filter and wrapper approaches [Kohavi 1997]. Filter approaches use statistical measures to evaluate the quality of the selected feature subset, whereas wrapper approaches use a learning algorithm (or classifier) to evaluate the goodness of the selected features.

According to this brief state-of-the-art, we can say that the process of selecting an optimal feature subset is a crucial key to build hight-quality machine learning model. Additionally, we can say that the process of feature selection is performed through the resolution of a multi-objective optimization problem by using methods propre for performing simultaneous optimization. In this thesis, we have chosen metaheuristic approaches based on the collective intelligence of a set of individuals to extract knowledge and overcome the complex difficulty to find the optimal solution.

1.1.2 Multiobjective feature selection for classification

This section attempts to provides a brief survey of multiobjective feature selection for classification.

Multiobjective optimization problem (MOP) involves optimizing several conflicting objective functions and possesses a set of solutions rather than single solution. The aim of any resolution method is on the one hand, to find the tradeoff non-dominated solutions called Pareto solutions (known as Pareto front), and on the other hand, to obtain well distributed Pareto solutions in order to maintain the diversity in the search space [Deb 2014]. In the literature, many multiobjective optimization algorithms such as non-dominated sorting genetic algorithm (NSGA) [Srinivas 1994], non-dominated sorting genetic algorithm 2 (NSGA2) [Deb 2002a], non-dominated sorting particle swarm optimizer (NSPSO) [Li 2003], Multi-Objective Artificial Bee Colony (MOABC) [Akbari 2012] have been proposed. The effectiveness of these algorithms for solving any multiobjective problem has encouraged researchers to apply them for tackling feature selection problems.

In [Hamdani 2007], a wrapper feature selection algorithm using NSGA2 is developed. This algorithm attempts to minimize both the classification error and the number of features. K-Nearest Neighbours (KNN) algorithm was incorporated as classifier to evaluate the goodness of the selected feature subset. In [Xue 2012a], two binary multiobjective PSO frameworks were first developed. The first called

NSBPSO by using the NSPSO. The second called CMDBPSO by incorporating the ideas of crowding, mutation and dominance into PSO algorithm. Then, based on NSBPSO and CMDBPSO, four filter feature selection approaches are proposed which are: NSfsMI, NSfsE, CMDfsMI and CMDfsE. All these four approaches have the same goal which is: minimize the number of features and minimize the classification error rate. Since that they are filter-based approaches, mutual information and entropy are used as evaluation criteria. Compared against single-objective feature selection approaches, the results showed that the proposed multiobjective approaches perform better. Other multiobjective wrapper-based algorithms have been proposed by [Xue 2012b] called NSPSOFS and CMDPSOFS. Since that the two algorithms are wrapper approaches which need a classification algorithm to evaluate the performance of the selected subset, both NSPSOFS and CMDPSOFS utilize KNN algorithm (with K=5) as classifier. The results showed that CMDPSOFS algorithm can evolve a Pareto front which include a smaller number of features and achieve lower classification error rate. In [Hancer 2015], the MOABC algorithm is investigated to develop three filter-based feature selection approaches. Therefore, three filter evaluation criteria are used which are mutual information, fuzzy mutual information and new fuzzy mutual information proposed by the authors. The developed approaches are compared against three single-objective feature selection approaches and the results showed that the proposed methods perform better in terms of classification accuracy and especially in terms of the number of features. Recently, Grey Wolf Optimizer (GWO) [Mirjalili 2014] was investigated to propose new wrapper-based feature selection approach to improve the classification of cervix lesions [Sahoo 2017]. To achieve this, a non-dominated sorting based-GWO is developed to minimize the number of textural features and maximize the classification accuracy of cervix lesions. By the fact that the proposed algorithm (called NSGWO) is wrapper approach, Support Vector Machine (SVM) is employed as learning algorithm to evaluate the selected feature subset. Based on NSGA algorithm, [Waqas 2009] propose a new wrapper approach for feature selection. ID3 decision tree algorithm was employed as classifier to evaluate the fitness of each individual. In [Xue 2014a], two filter-based multiobjective feature selection approaches are proposed called MORSN and MORSE to obtain the set of non-dominated feature subsets. To achieve this goal, rough set (RS) [Pawlak 1995] is used to construct two measures: the first measure is to evaluate the classification performance and the second is to evaluate the number of features.

1.2 Motivations

After the presentation of the concept of feature selection, its high importance and its influencing in machine learning, and after showing the relationship between a feature selection problem and a multiobjective optimization problem, including the presentation of some feature selection algorithms. We summarize the main motivations of our work in the following points:

- The tremendous usefulness of machine learning techniques in todays society and for everyday life. Medical diagnostics, marketing and airport security are some examples in which machine learning has had remarkable success.

- Addressing feature selection problem as it is one of the most important task in machine learning. Feature selection is a complex problem in which it is too expensive to discover entire the feature space. Therefore, the search for an optimal feature subset is NP-hard problem for which we don't know any exact approach that can select only the needed features in acceptable running time.

- Addressing the feature interaction problem. Indeed, the difficulty of feature selection problem is not only due by the fact that the search space is very large, but also by the fact that the features are highly interacting between them. A given feature may be irrelevant, but when it is combining with other feature, it can becomes extremely relevant and very informative.

- Addressing feature selection as a multiobjective optimization problem. Although there exist some multiobjective appoaches for solving feature selection problem, but in the literature, there are rare approaches that handling feature selection as a multiobjective problem compared to those that handling it as single-objective problem.

- Taking advantage of metaheuristics based on the evolution of a population of solutions which offer a high ability to explore a large part of the srach space in reasonable computational time and overcome the problem of local optimal feature subset.

1.3 Major contributions

To solve feature selection problems using metaheuristics methods, our contribution is made in two phases:

- The first phase consists of developing a resolution approach as a new alternative to solve multiobjective optimization problems in a general manner.

- The second phase is to investigate the proposed approach for developing a new feature selection algorithm.

In the first contribution, we have proposed a new multiobjective optimization algorithm called GPAWOA (for Guided Population Archive Whale Optimization Algorithm). The proposed algorithm extend a recent single-objective algorithm called Whale Optimization Algorithm (WOA) in ordre to be able to deal with multiobjective optimization problem. The GPAWOA incorporates the Pareto optimality concept into the standard WOA and uses an additional population (archive) to guid the search towards the true Pareto front. In addition, the mechanism of crowding distance is adopted in two steps of the algorithm in order to obtain well-distributed solutions. The GPAWOA was evaluated using 12 well-known benchmark functions (5 bi-objective test functions and 7 three-objective test functions). Moreover, it was applied to four multiobjective engineering design problems. The exprimental results showed that the proposed GPAWOA is highly competitive compared against existing multiobjective optimization algorithms, being able to provides an excellent approximation of Pareto front in terms of convergence and diversity.

In the second contribution, we have applied the proposed GPAWOA for solving feature selection problem. This contribution consists of hybriding filter and wrapper models into a single system at the hope to benefits from the merits of each model. The proposed algorithm demonstrated its ability to be a good alternative for handling effectively the feature selection problems.

1.4 Thesis organization

The remainder of this thesis is organised as follows:

Chapter 2 gives a brief review of machine learning and presents some well-known classification algorithms. Also, it provides a short review of feature

selection problem, including the key steps of any feature selection approach and the main types of feature selection algorithms.

Chapter 3 provides a generality of optimization problems. It starts with the definition of the complexity theory and single-objective optimization problems. Then, it presents some exisitng resolution methods, in particular, the metaheuristics that use a population of solutions.

Chapter 4 presents the essential background and basic concepts of multiobjective optimization problems. It provides the essential notions and in particular the dominance Pareto concept. Also, it gives the main challenges of multiobjective problems. In addition, it reviews typical and popular resolution approaches.

Chapter 5 presents the proposed GPAWOA algorithm for solving multiobjective optimization problems. First, it gives a detailed description of GPAWOA algorithm, including the mechanisms used and the archiving strategy adopted in the algorithm and its pseudocode. Finally, it provides a summary of an extensive experimental evaluation.

Chapter 6 devoted to the description of the proposed FW-GPAWOA algorithm for feature selection. It provides how FW-GPAWOA treat discrete problems and how combining filter and wrapper approaches. Finally, it provides a summary of the comparative study.

Chapter 7 summaries the work and attractions overall conclusions of the thesis. Main research ideas and the contributions of the thesis are established as well. It also suggests some possible future research directions.

1.5 Academic publications

- Got Adel, Abdelouahab Moussaoui, and Djaafar Zouache. "A guided population archive whale optimization algorithm for solving multiobjective optimization problems." Expert Systems with Applications 141 (2020): 112972. https://doi.org/10.1016/j.eswa.2019.112972, impact factor: 5.452.

- Got Adel, Abdelouahab Moussaoui, and Djaafar Zouache. ”Hybrid filter-wrapper feature selection using Whale Optimization Algorithm: A Multi-Objective approach”. Expert Systems with Applications (Under review), impact factor: 5.452.

CHAPTER 2

Machine learning and feature selection

Contents

2.1 Introduction

Machine learning is a subfield of Artificial Intelligence (AI) which consists in building from a set of sample data, a learning model that allows to a machine to self-learning, that is to say, to analyze and reason in an autonomous manner with complex data. This ability to learn automatically from analytical observation of the sample data results in obtaining an intelligent artificial system able for learning and adapting with new datasets without being explicitly programmed.

Therefore, machine learning refers to the development of methods that enable to a machine to evolve automatically and accomplish tasks which are complex for classical algorithms.

In general, the process of a learning algorithm is divided into two phases [Aggarwal 2014]. The first phase (learning phase) consists to generate a prediction model ready to the implementation for the future situations. The second phase (deployment phase) consists of applying the generated model on the datasets to accomplish the desired tasks (usually is decision-making). Its clear that the quality of the final decision is strongly depends on the performance and the accuracy of the generated learning model. In addition, the input space (dataset) of a learning algorithm is often divided into a training space and testing space [Muñoz 2018]. The training space constitutes the set of examples used to provide the learning model, while the testing space is used to check the performance of the generated model.

Although there exist several types of machine learning, however, supervised and unsupervised learning that we will present in the following sections, are the most known types of machine learning and they are the most studied in the literature.

2.2 Supervised learning

Supervised learning consists in generating a typical prediction model connecting the predictor variables X and a target variable Y predetermined beforehand. A supervised learning algorithm takes as input a sample data (training set), where each data examples is assigned to a categorized variable called the target class (label). The goal is to build a learning model able to predict with high accuracy the target classes of new data examples (unseen datasets) that are not belong to the learning sample [Dougherty 1995].

Formally, given a training set D containing m example, where each example E_i is represented by the input x_i and its associated output target y_i :

$$D = \{E_i/E_i = (x_i, y_i), x_i \in X \ and \ y_i \in Y, i = 1, 2, 3, ..., m\} \tag{2.1}$$

The objective is to build a learning model called hypothesis function $h(x) = y$ with $x \notin X$, able to predict for any given input datum x, the value of the output target y. Depending on the nature of the output target Y, there are two types of supervised learning problems:

- **Supervised classification** if the target variable Y takes discrete values (binary, boolean, ect.). For a classification problem, Y corresponds to a finite number of classes used to find the model that classifiy the unseen datasets.

- **Regression problem** if the output target Y is continuous value. There are several regression forms, the simplest is the linear regression. It tries to fit data with the best hyperplane which goes through the points. Therefore, the learning model will be a simple straight line.

2.2.1 K-nearest neighbor algorithm (KNN)

The k-nearest neighbor algorithm (KNN) is considered to be one of the simplest supervised classification algorithms [Indyk 1998]. Its principle is to assign the input data to the most represented class among its k-nearest neighbors in terms of distance as shown in figure 2.1.

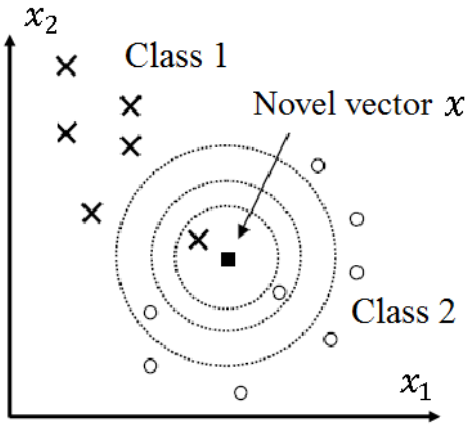

Figure 2.1: K-Nearest Neighbor classification algorithm

Given a new unlabeled vector x, the KNN algorithm consists in finding the set E containing k-closest classes to x, and by majority vote, it determines the most frequent class in E and assigns it the new vector x.

The choice of the optimal k is important for better classification. Indeed,

choosing smaller k can be less immunized against the noise in the learning sample, a large value of k makes it difficult to define the limits between classes.

There are many variants of this algorithm depending on the distance calculation function used [Chomboon 2015], usually the most used is the Euclidean Distance. The major advantage of the k-nearest neighbor method is its simplicity and ease of implementation. On the other hand, the high cost when searching for neighbors is its main drawback, especially in large databases. In addition, the choice of k-value might be not obvious.

2.2.2 Support Vector Machine (SVM)

Support Vector Machine method is based on the theoretical work of Vanpik and Cortes [Cortes 1995]; [Boser 1992]. SVMs are usually used for classification models and applied to linearly and non-linearly separable problems. The objective is to find the optimal separating hyperplane among an infinity of hyperplanes, which linearly separates the learning sample by maximizing the margin (in terms of distance) between the separating hyperplane and the closest points to the limit of separation (see figure 2.2). These points are called support vectors.

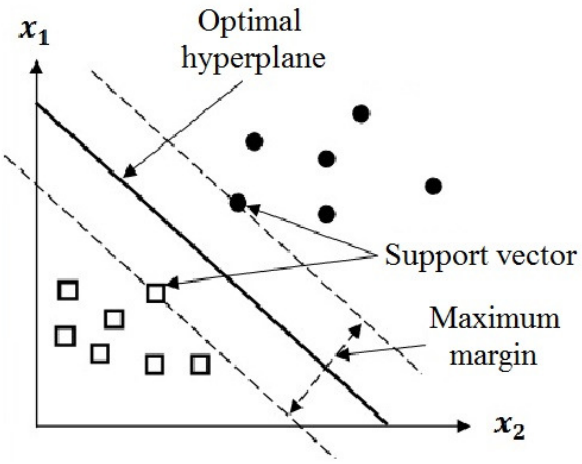

Figure 2.2: Optimal hyperplane for Support Vector Machine with two classes

SVMs are based on two concepts which are : the maximum margin and the kernel function. In the linearly separable case, the separating hyperplane is a simple straight line defined by the function h:

$$h(x) = \sum_{i=1}^{n} w_i.x_i + b \tag{2.2}$$

where, w is the normal vector of the hyperplane and b is the displacement from the origin.

As we have seen previously, the optimal hyperplane is the one which maximizes the minimum distance between the separation limit and the support vectors. This distance is given as follows:

$$d(x_i, h) = \frac{y_i.h(x_i)}{\|w\|} \tag{2.3}$$

with, y_i is the corresponding class of any point x_i.

Maximizing the distance d amounts to minimize the quantity $\|w\|$, which leads, after certain mathematical operations, to the following optimization problem:

$$\begin{cases} min\frac{\|w\|^2}{2} \\ y_i.h(x_i) \geq 1, i = 1, 2, 3, ..., n \end{cases} \tag{2.4}$$

This problem can be solved using the lagrange multipliers method. Therefore, the hyperplane function for a new datum x is written as follows:

$$h(x) = \sum_{k=1}^{n} a_k.y_k.(x.x_k) + b \tag{2.5}$$

where a_k are the lagrange multipliers.

In the non-linearly separable case where there is no hyperplane able to separates the classes, SVM uses the so-called $kernel\ function$ to transform the initial space into a higher dimension space ϕ in which there is a possibility to perform a linear separation. Accordingly, the function of the separating hyperplane is defined in the new space ϕ and it is given by:

$$h(x) = \sum_{k=1}^{n} a_k.y_k.K(x, x_k) + b \tag{2.6}$$

$K(x, x_k)$ is the kernel function. There are many kernel functions such as polynomial, gaussian and RBF kernel functions [Schölkopf 2002]. The SVMs have shown their ability to provide excellent performance in classification problems, and they have been employed in several areas such as the pattern recognition [Byun 2002] and medical diagnostics [Pimple 2016]; [Sweilam 2010].

2.3 Unsupervised learning

Unlike the supervised learning where the database is labeled, unsupervised learning is used when the learning sample contains only raw data (unlabeled examples). That is to say, the target classes including their numbers are not defined a priori. Consequently, this type of problem is often more difficult compared to supervised problem.

An unsupervised algorithm consists in finding patterns or clusters that group the dataset based on certain similarity between the data. Therefore, the goal is to discover the target classes that maximize the intra-class similarity and minimize the inter-class similarity [Jain 2000]. Various tasks are associated to this type of learning, in particular, the clustering which involves grouping the dataset into several classes called $clusters$, where each $cluster$ is a collection of similar objects. Usually, this similarity criterion is expressed in terms of distance function.

2.3.1 k-means algorithm

The k-means method proposed in [MacQueen 1967] is one of the most popular clustering algorithms in the current usage as it is relatively fast and simple to understand and easy to deploy in practice. As illustrated in figure 2.3, k-means algorithm consists of partitioning the dataset into K clusters, such that objects within the same cluster are similar as well as possible, while the objects from different clusters are dissimilar as well as possible. In k-means algorithm, each cluster is represented by its center $(centroids)$ that corresponds to the mean of points assigned to the cluster. The k-means algorithm is given in the following steps [Selim 1984] :

1. Determine the number K of clusters C_i.

2. Initialize randomly from the data points, K centroids (denoted w_i).

3. Assign each data point to the closest cluster (centroid).

4. Update the cluster centroids w_i so that the cost function is to minimize:

$$V = \sum_{i=1}^{k} \sum_{w_i \in C_i} D(x_i, w_i)^2 \tag{2.7}$$

with, x_i is the point belongs to the cluster having the centroid w_i and $D(x_i, w_i)$ is the Euclidean distance between w_i and x_i.

5. If the partitioning does not converge, return to 3.

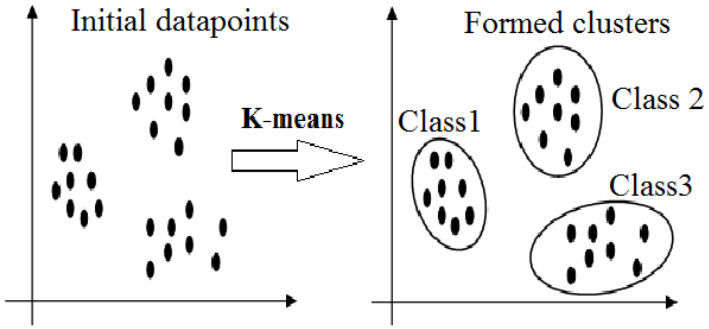

Figure 2.3: Example of clustering using K-means algorithm

The good accuracy and efficiency of the K-means algorithm motivate researchers to propose several variants such as K-medians [Juan 2000], K-medoids [Park 2009] which applied different techniques to calculate the centroids of clusters.

2.3.2 Hierarchical clustering algorithm (HCA)

Hierarchical clustering algorithm (also known as hierarchical cluster analysis) is a classical clustering algorithm which consists for grouping two individuals or two groups (clusters) of individuals that are closest together in terms of distance until all objects are grouped into a single cluster [Ward Jr 1963]. The result of this type of algorithm is usually represented as a hierarchical tree (or dendrogram). In order to obtain the desired classes, a division at a certain level of the dendrogram must be performed (see figure 2.4).

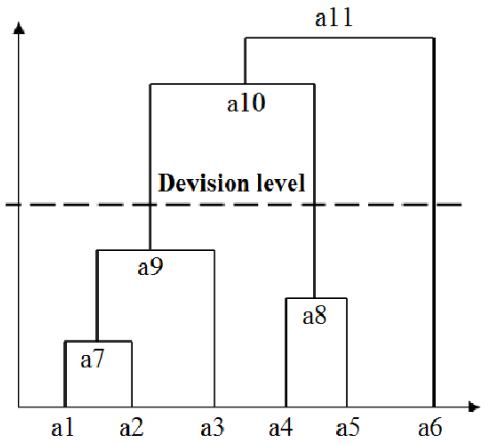

Figure 2.4: Dendrogram for the Hierarchical Cluster Analysis (HCA)

Let $A = \{a1, a2, ..., an\}$ be the set of individuals to be grouped, HCA algorithm takes place in the following way:

1. Initialize the clusters C_i where each cluster contains a single individual of A.

2. Combine the nearest clusters C_i and C_j to form a new cluster C_k.

3. Calculate the distance between C_k and the clusters C_y with $y \neq i, j$.

4. Repeat 2 and 3 until the aggregation of all individuals of A into a single cluster.

2.4 Dimensionality reduction based on feature selection

In practice, the performance of any learning algorithm does not depend only to its design and its manner to deal with the set of data. Indeed, several factors can affect the quality of the algorithm, including the representation and quality of the dataset. High dimensionality of data, the existence of irrelevant and redundant data (features) are some factors which could decrease the algorithm performance [Dash 1997]. Consequently, it is often necessary to reduce the dimensionality of the sample learning (dataset) in order to remove the irrelevant and redundant

features. Generally, the dimensionality reduction strategies are grouped into two categories :

- Reduction by feature selection, which consists to select the most relevant features from the dataset.

- Reduction by feature extraction, which attempts to transform the initial space to lower dimensional space. Principal Component Analysis (PCA) is one of the typical methods for dimensionality reduction based on feature extraction [Jolliffe 2016].

In our thesis, we're interested on the reduction through the use of the feature selection technique. For this reason, a brief overview on the feature selection problem will be presented in the following sections.

2.5 Feature selection problem

Feature Selection is one of the core concepts in machine learning which try to select the most informative (relevant) variables or attributes from highly dimensional data. It is a technique widely used to reduce the size of the datasets while building the subset containing the features deemed relevant. The impact of the feature selection is not limited only to the dimensionality reduction, but extends to the construction of an efficient and simplest learning model. Hence, improving the performance of the system and optimizing the running time [Langley 1994]. The feature selection problem is the process of finding, from the original features, the optimal subset that is sufficient for solving successfully a classification problem. The optimal feature subset is the minimum subset that can provide the better classification accuracy, which makes feature selection a multiobjective problem [Xue 2012b]; [Oliveira 2002].

2.6 General procedure of feature selection algorithm

According to [Dash 1997], a typical feature selection algorithm includes the four key steps illustrated in figure 2.5. The search process starts by generating the can-

didate feature subsets which will be evaluated by using a certain evaluation criterion. These two steps are repeated until a given stopping criterion is satisfied. Finally, a result validation step is applied to validate the selected feature subset.

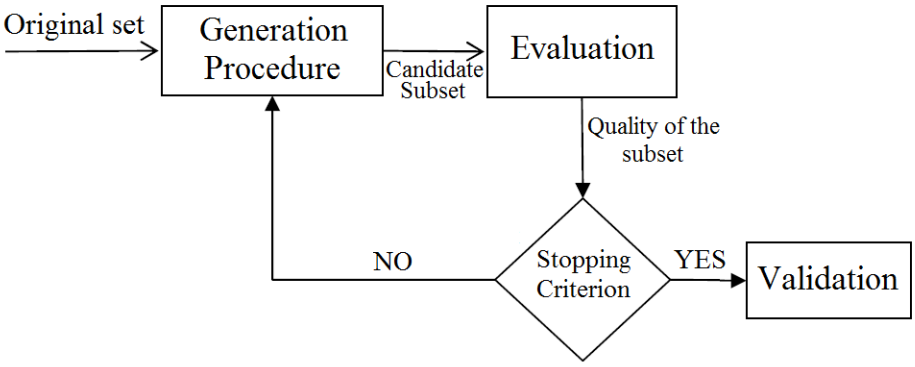

Figure 2.5: Four key steps of feature selection algorithm [Dash 1997]

2.6.1 Generation procedure

The generation procedure (also called search procedure) consists of creating a subset of candidate features from the initial set of features. However, for a given set containing N features, there are 2^N possible subsets, which makes an exhaustive search very expensive even impossible for a large number of features. Therefore, three search strategies have been proposed [Liu 2005]:

- **Complete search**
  This strategy can provides the guarantee of finding the best feature subset according to the employed evaluation criterion. An exhaustive search is absolutely complete while a non-exhaustive search can be complete. This is possible by using different heuristic functions to reduce the search space without compromising the chances of finding the optimal subset by applying backtracking process (Branch and bound for example) allowing to go back if the selection is going in the wrong generation direction.

- **Sequential search**
  Also called heuristic search, the principle of sequential search is to add or delete, iteratively, one or more features. It begins either with an empty set

and then the features are added one at a time until a stopping criterion is reached (sequential forward selection SFS), either by the original set and then eliminates features one at a time (sequential backward selection SBS). A third search strategy (sequential stepwise selection) consists for combining the previous strategies while removing or adding the features from the selected subset.

- **Random search**
  As its name indicates, the random search (also called stochastic search) consists of randomly generating a finite number of candidate features. The methods based on such strategy (such as genetic algorithm) are able for escaping from local optima subset and reducing the running time.

2.6.2 Evaluation procedure

In this step, an evaluation criterion is defined in order to measure the quality of each features subset generated in the previous step. Obviously, the choice of the evaluation function is crucial to obtain an optimal subset. The generated subset using a given evaluation criteria may be not the same using another evaluation criteria [Xue 2014b]. Additionally, according to the used evaluation criterion, the feature selection algorithms can be classified into two main categories : Filters and Wrappers.

- **Filter approach**
  In filter methods, the evaluation is realized without involving any mining algorithm, which means that the evaluation is independent to the used classifier. Several measures criterion are proposed to evaluate the goodness of the candidate features. Consistency, mutual information, correlation criteria and rough set theory are usually the most employed filter functions in the literature [Dash 2003]; [Chandrashekar 2014].

  Filter algorithms are known as fast to compute and easy to understand. However, the ignorance of features dependency and the absence of interaction with the classifier are the main disadvantages of filtering methods.

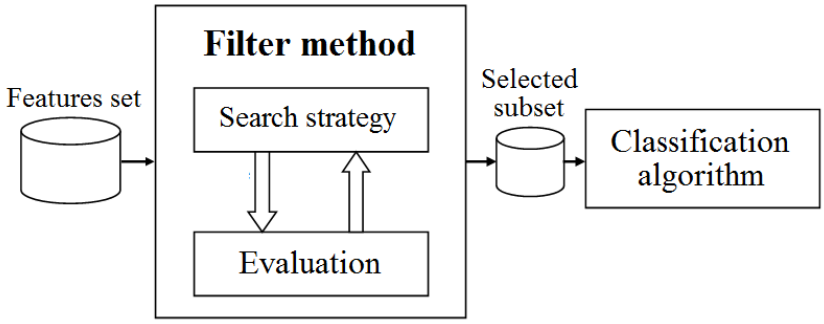

Figure 2.6: Overall schematic of filter method

- **Wrapper approach**
  In contrast of the filter methods which try to find the optimal feature subset independently of the classification algorithm, the wrapper methods integrate the classifier to search and evaluate the candidate feature subset, which results in increasing the chance to obtain a better subset.

  In general, the use of wrapper methods make it possible to obtain a high accuracy comparing to filter methods thanks to the well interaction of the selected features with the classifier. However, the big shortcoming of these methods is they require high computational costs even when using simple classification algorithm.

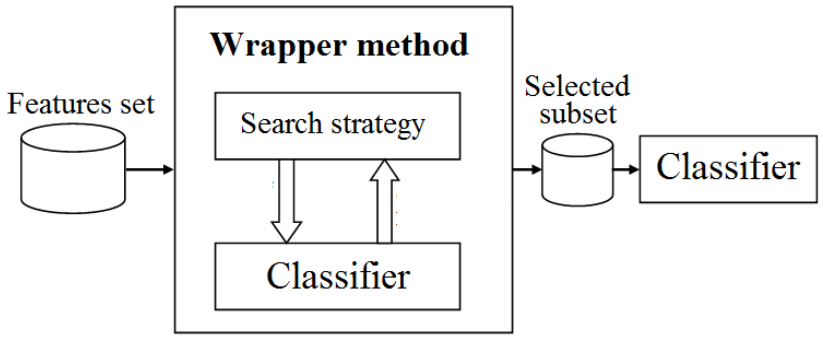

Figure 2.7: Overall schematic of wrapper method

2.6.3 Stopping criteria

Generally, the number of features to be selected is unknown a priori, which makes the definition of a good stopping criterion a crucial step while ensuring the selection of the needed features. In fact, a poor choice of the stopping criterion can force the selection process to stop while the optimal subset is still far away. In practice, the choice of stopping criterion depends on the search strategy and the evaluation function. Among the most used stopping criteria in the literature, we can cite :

- Predefined number of features to be selected (knowing that this task is not often achievable as mentioned previously).
- Predefined number of iterations is reached (this type of criterion can reduce the computational costs but the result is not necessarily optimal).
- Addition or deletion of any feature does not result in a better subset.
- An optimal subset is obtained (for example, given error rate is allowable).

2.6.4 Validation

The result validation is considered as the fourth and the final step in the feature selection process. It depends on the nature of handled dataset, artificial or real. A straightforward way for validation is to directly measure the quality of the obtained features subset. In this case, prior knowledge about the relevant features is necessary. The database benchmarks are typically used to directly validate the obtained results.

In real-world applications, generally there is no prior knowledge about the relevant features. In this case, the validation process utilize some indirect methods by observing the change of the performance indicator (such as classification error rate) with the change of features subset [Liu 2005]. Cross-validation technique might be also used to assess the performance of the selected feature subset.

2.7 Chapter summary

In this chapter, we have presented the basic concepts of machine learning and its main types; supervised and unsupervised learning, as well as their operating principles and some algorithms for each type.

We have also presented one of the important related task to machine learning

which is the dimensionality reduction, in particular, the reduction based on the feature selection. Consequently, a brief overview for feature selection is given in which we have presented the main steps to be taken into account when designing any feature selection algorithm. In addition, feature selection approaches, namely: filter and wrapper algorithms were presented, including the advantages and disadvantages of each model.

CHAPTER 3

Generalities on the optimization problems

Contents

3.1 Introduction

Currently, the optimization has become an important multidisciplinary field that can be used in image processing, engineering design, machine learning, bioinformatics, economics, etc. The researchers, decision-makers and engineers are often faced against high-dimensional problems that might requires an exhaustive

search, which is not always feasible in a reasonable time.

An optimization problem can be defined as the act of finding the best result among the set of all feasible solutions. However, most of these problems are NP-hard where there is no exact resolution method that can find the best solution in reasonable computation time. Without claiming to be an exhaustive presentation, we will present in this chapter the main optimization concepts. Hence, some resolution methods will be presented, in particular, the so-called "metaheuristics".

3.2 Complexity theory

For a given problem, there are several algorithms that can be used for solving it, and the user is faced with the following question: among this multiplicity of algorithms, which algorithm is performing better ?. The straightforward way to answer this question is to compare the running time of each algorithm. However, this comparison is unfair because it is affected by several factors such as the characteristics of the machine in which the algorithms are implemented. Therefore, other approach independent of any software and hardware resources is necessary to perform rigorous and an equitable comparison.

The complexity of an algorithm can be defined as the estimation of necessary number of elementary operations performed by this algorithm to solve a given problem. The complexity theory consists [Papadimitriou 2003]; [Arora 2009] of formally studying the difficulty of a decision problem whose the answer is of binary type (yes/no, true/false, 0/1, ect). Furthermore, some studies [Cook 1971]; [Karp 1972] have shown that there is a fundamental classification of optimization problems based on the complexity of the algorithms used for solving them.

Définition 3.1 ***(P class)*** *We distinguish by P class the problems that can be solved efficiently by an algorithm of polynomial complexity, that is to say in $O(N^p)$ where N is the size of the problem and p is a constant.*

Définition 3.2 ***(NP class)*** *The NP class involves the problems that can be solved by a non-deterministic algorithm in polynomial time. Moreover, the correctness of a claimed solution can be checked in polynomial time. From this definition, it is clear that the inclusion $P \subset NP$ is valid, but the reverse was never been affirmed.*

Définition 3.3 ***(NP-complete class)*** *This class contains NP problems that are considered more difficult than any other NP problem. A problem Q known to be NP-complete if it is in NP and if any NP problem can be reduced to Q in polynomial time. Among the well-known NP-complete problems we can cite: the boolean satisfiability (SAT) and knapsack problem.*

3.3 Optimization problems

Each optimization problem is characterized by its search space, an objective function and a set of constraints:

- The search space is the set of the feasible solutions and it is defined by a finite set of decision variables, each of them with a finite domain.

- The objective function represents the goal to be reached (minimizing the cost, the error rate, or maximizing the profit, the measure, etc.) according to the problem context.

- The constraints are the set of conditions that should be respected during the optimization process. They are often of inequality or equality equations and they allow to boundary the search space.

An optimization problem is defined as the act of searching within the search space, the solution that minimizes (or maximizes) the objective function while satisfying the constraints. This solution is called optimal solution or the global optimum. However, for a given optimization problem, a candidate solution might be optimal for a part of search space but not for all the search space: we talk about local optima and global optima. These two notions are illustrated in figure 3.1.

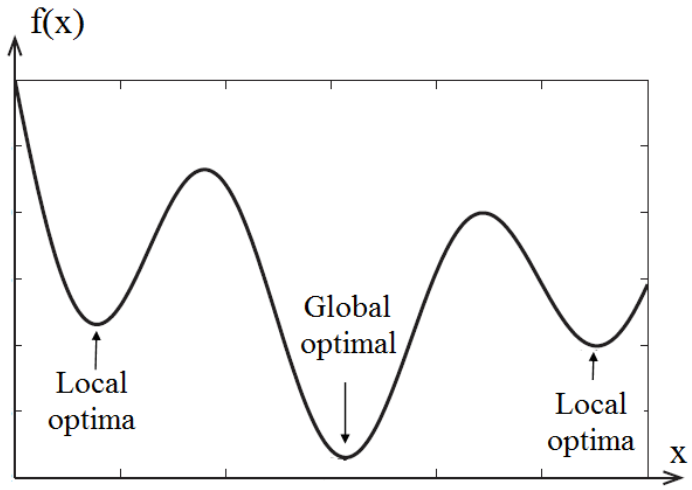

Figure 3.1: Illustration of local optimum and global optimum

3.3.1 Mathematical formulation

Mathematically, an optimization problem is formulated as follows:

$$\begin{cases} min\ f(x) \\ g_i(x) \leq 0, i = 1, 2, 3, ..., m \\ h_j(x) = 0, j = 1, 2, 3, ..., k \\ x^L \leq x \leq x^U \end{cases} \tag{3.1}$$

where $x \in S$ is the vector of decision variables and S is the search space, $f(x)$ is the objective function to optimize, g_i and h_j are, respectively, the inequality and equality constraints. x^L and x^U are, respectively, lower and upper bounds of decision variables. The aim is to find the point x^* that satisfy the constraints g_i, h_j and $\forall x \in S\ ,\ f(x^*) \leq f(x)$ [Schoen 1991].

3.3.2 Resolution of optimization problems

The main objective when solving optimization problems is finding the global optima rather than the local optima. In order to achieve this goal, an exploitation strategy of the search space must be defined [Chen 2009]. Additionally, due to the existence of several local optima, an exploration strategy is necessary to overcome the local optimality problem [Xu 2014]. The exploration attempts to visit a large

part of the search space, while the exploitation express the ability of the resolution method to focus in the regions that probably containing an optimal solution. According to this description, we can say that the performance of such resolution method depends on the manner to deal with the exploration/exploitation dilemma.

In the literature, there exist several solution methods which are usually categorized into exact and approximate (metaheuristics) approaches [Talbi 2009]. In the following, some existing methods will be presented.

3.4 Optimization exact methods

The exact approaches include all algorithms which are guaranteed to find an optimal solution and to prove its optimality for every instance of the combinatorial optimization problem [Puchinger 2005]. Generally, they take the form of exhaustive search. These methods have an exponential computational complexity, indeed, the time to solve the problem grow exponentially with its size.

3.4.1 Linear programming

Linear programming (also called linear optimization) is a decision mathematical support tool which is dedicated for solving optimization problems whose the objective function and the constraints are in the form of a system of linear equations.

The general form of a linear program (LP) is given as:

$$\begin{cases} min\ f(x) = \sum_{i=1}^{n} c_i x_i \\ Ax \geq b \\ x_i \geq 0, x \in R^n, A \in R^{m*n}, x \in R^m \end{cases} \tag{3.2}$$

with, c_i are the coefficients of the objective function $f(x)$, A and b are the matrix and vector, respectively, of constraints. x is the vector of decision variables. The solution of a linear program is a values assignment to the decision variables of the problem. The solution is feasible if it satisfy all the constraints. One of the well-known optimization algorithms in linear programming is the simplex algorithm [Bartels 1969].

3.4.2 Branch and Bound (B and B)

Branch and Bound [Land 1960]; [Lawler 1966] is a technique usually used for solving combinatorial optimization problems. It consists of an enumeration systematic non-exhaustive of the solutions space while eliminating the subsets of solutions that does not lead to the desired solution. Its operating principle is inspired by the classic computer science paradigm "Divide to conquer" and it is based on two concepts:

- **Branching** which is the process of spawning subproblems from the initial problem while building a "decision tree" in which the nodes represent the sub-problems, each one is associated to a reduced solutions space.

- **Bounding** which attempts to guide the search tree by using the so-called "lower" and "upper" bounds. It consists to maintain the current subset as a potential search space, or pruning it if its associated bound is worse than the best known solution. The classical bounding strategies are; either browsing the decision tree in depth first (Deep-First Search), or in width first (Breadth-First Search), or handle the best subset not yet treated (Best-First Search).

Branch and Bound algorithm can obtains an optimal solution and enumerates the search space in acceptable complexity if the problem is of limited size. Due to its efficiency and simplicity, several variants have been proposed such as Branch and Cut [Padberg 1991] and Branch and Price [Barnhart 1998].

As mentioned above, the computational cost of exact approaches increase exponentially with the problem size. Thus, due to the "combinatorial explosion" nature of the real-world problems, such approach is not efficient and even not applicable. In this case, the so-called "metaheuristics" represent an interesting alternative which aimed at providing high-quality solution (instead of guaranteeing a global optimum solution) in acceptable computing time [Gogna 2013]. Based on a stochastic search, the metaheuristics progress iteratively by alternating effectively between exploration and exploitation techniques in order to converge as well as possible towards the optimal solution. Many criteria may be used to categorized the metaheuristic, the classification of metaheuristics, which differentiates between Single solution-based and Population-based, is often taken to be a fundamental distinction in the literature [BoussaïD 2013].

3.5 Single solution based metaheuristics

Single solution methods manipulate a single point of the search space throughout the progression of the algorithm. Also called trajectory methods, they start with a random initial solution and try to, iteratively, evolve and improve the current configuration in it local neighborhood while constructing a trajectory path within the search space. Tabu Search and Simulated Annealing are considered as the most popular single-based algorithms in the literature.

3.5.1 Tabu Search

Tabu search proposed by Glover in 1986 [Glover 1986] is a local search method that inspired by the human memory. In fact, Tabu Search (TS) utilizes an adaptive memory called $tabu\ list$ in which the promising regions already visited are listed. This mechanism allows to prevent the search process to perform backwards moves to the solutions previously explored. The principle of Tabu search is the use of the neighborhood notion that consists in replacing the current solution by the best solution among all of its neighbors. However, this strategy may lead to a premature convergence. Therefore, Tabu search permits moves that deteriorate the objective function in ordre to overcome local optima.

Algorithm 3.1: Tabu Search algorithm

```
Initialize the tabu list T = ∅
Choose an initial solution s
V(s) : neighborhood of the current solution s
S_best = s
while t < iter_max do
    Choose the best solution s' from V(s)
    if f(s') < f(S_best) then
        S_best=s'
    end
    Update the tabu list T
    s = s'
    t=t+1
end
```

3.5.2 Simulated Annealing

Simulated Annealing (SA) is one of the simplest and effective method, it is based on the physical annealing process used by metallurgists [Kirkpatrick 1983]; [Černỳ 1985]. Annealing aims to obtain homogeneous and well structured materials by alternating slow cooling and heating cycles until reaching the thermodynamic equilibrium. A metal in the liquid state (at high-temperature) is a metal whose the atoms that compose it are unstable. The cooling act (lowering of temperature) enable to stabilize these atoms while minimizing their energies of displacement. However, the sharp cooling may unbalance the atoms. Hence, the lowering of temperature must be done in a regular and careful manner. This physical phenomenon has been perfectly applied in Simulated Annealing algorithm to solve optimization problems where: the temperature becomes a control parameter and the energy becomes the objective function to be optimized [BoussaïD 2013].

Algorithm 3.2: Simulated Annealing algorithm

```
1  s ← initial solution
2  N(s) : neighborhood of the current solution s
3  S_best = s
4  T ← maxTemperature
5  while t > minTemperature do
6      iter ← 0
7      while iter < MaxIter do
8          Select a random solution s' from N(s)
9          Δ = f(s') − f(s)
10         if Δ < 0 then
11             s = s'
12             if f(s') < f(S_best) then
13                 S_best = s'
14             end
15         else
16             if rand(0,1) < e^(−Δ/T) then
17                 s = s'
18             end
19         end
20         iter = iter + 1
21     end
22     T ← T × (1 − α)
23 end
```

At the start of the algorithm, the T parameter is initialized to a high value and the search begins with a random initial solution. At each iteration, another solution s' is chosen from the neighborhood of the current solution s. If this solution improves the objective function, it will be accepted. Otherwise, it will be accepted but with a certain probability P which depends on T parameter. Indeed, P is higher whenever T is higher. Initially, T parameter is high enough in order to favored the exploration of the search space, thereby escaping from local optima. During the iterations, T parameter decreases and consequently, the probability P of accepting a degradation within the solution quality decreases. This mechanism improve the ability of the algorithm to focus on promesing regions while accepting only the configurations that enhance the objective function in order to converge to the optimal solution.

Due to its high ability to provides an excellent approximation of the global optima, Simulated Annealing algorithm has been successfully applied in several academic optimization problems [Elmohamed 1997]; [Malek 1989]; [Drexl 1988] and it has a broad range of real applications [Suman 2006].

3.6 Population based metaheuristics

Unlike single solution-based methods, population-based metaheuristics manipulate a set of solutions called *population* and they are usually inspired by natural phenomena. These methods represent a powerful tool for solving different types of optimization problems thanks to their ability for performing global search. Additionally, using a population of solutions increase their ability to explore very large part of the search space in reasonable running time. In the following paragraphs, some population-based algorithms will be presented.

3.6.1 Genetic Algorithm (GA)

Genetic Algorithms (GAs) are considered as the well-known population-based metaheuristics belonging to the larger class of evolutionary algorithms (EAs). The origins of GAs dates back to the early 1970s in Holland's work [Holland 1975] by inspiring the process of natural evolution of living things and genetics. Genetic algorithms simulate the Darwins theory of evolution "survival of the fittest": fitter organisms have a better chance of surviving and to produce offsprings to the next generation.

Without having delved too much into the genetic aspect, the best individuals (parents) of the current population are selected to produce offsprings by applying two genetic operators known as $crossover$ and $mutation$. An individual (chromosomes) is composed of a sequence of genes. The crossover allows to create new childrens by exchanging the genes of two parents, while the mutation is a random pertubation on the genes of the borne children. Finally, the fittest children's replace the weakest parents to generate new population, hopefully, better than the original parental population.

Despite the field of genetic algorithms was established by John Holland. However, their mathematical formulations for solving optimization problems have been introduced by David Goldberg [Goldberg 1989].

Algorithm 3.3 provides the operating principle of GA. The population of individuals (solutions) are maintained within the search space. For each individual is applied a fitness score, that is, the evaluation of the objective function. At each iteration, the best solutions are chosen by using a given selection strategy such as roulette wheel selection, rank selection and tournament selection (for more details on selection schemes, you can see [Goldberg 1991]). Once the solutions are selected, crossover and mutation operators are applied with a certain probability in order to create an intermediate population. Mutation operators are mostly used to increase the exploration ability, whereas crossover operators are widely used to lead the population towards promesing points in the search space (i.e, increase the exploiatation ability of the algorithm). Finally, the fittest solutions in the intermediate population are selected to replace the weakest solutions in the current population in order to reproduce a new population better than the old population.

Algorithm 3.3: Pseudocode of a Genetic Algorithm

```
1  Generate an initial random population P of individuals
2  gen = 0
3  while gen < GenMax do
4  |   Evaluate the fitness of each individual
5  |   Q=Selection(P)
6  |   IntPop=Crossover(Q)
7  |   PopChild=Mutation(IntPop)
8  |   Evaluate the fitness of PopChild
9  |   new_Pop=Replacement(worst(P) with best(PopChild))
10 |   gen = gen + 1
11 end
```

Genetic algorithms have become an essential tool to solve many optimization problems and they have been successfully applied in several fields [Chaiyaratana 1997]. However, the choice of some parameters such as the rate of crossover and mutation should be defined carefully as they affect the algorithm's performance.

3.6.2 Particle swarm optimization (PSO)

Particle Swarm Optimization (PSO) proposed in [Kennedy 1995] is an evolutionary optimization technique that mimic the collective intelligent behavior observed in swarms of bird flocking or fish schooling. The swarm incorporates a set of particles that possess a local memory and attempt to improve their current positions in order to attain a common goal. Each particle moves according to its velocity by using its own knowledge and the information shared by its teammates. Indeed, each particle utilize its previous experiences (its best position obtained so far) and the best position reached by the whole swarm to update both its current position and velocity to gets the best position within the swarm.

The general principle of PSO (as shown in algorithm 3.4) is to deal with population of particles (solutions) and improve it iteratively in order to reach the global optima by always following the best position found so far. A particle is characterized by its current position in the search space and its velocity. Over the iterations, each particle updates its own velocity according to its current position, its best position achieved so far as well as the best position obtained by entire population.

$$v_i(t+1) = wv_i(t) + c_1 r_1(pBest(t) - x_i(t)) + c_2 r_2(gBest(t) - x_i(t)) \tag{3.3}$$

where, $x_i(t)$ and $v_i(t)$ represent, respectively, the current position and velocity of the particle i. $pBest(t)$ is the best personal position of this particle and $gBest(t)$ is the global best position. c_1 and c_2 are the acceleration coefficients. r_1 and r_2 are random numbers between 0 and 1. w is an inertia weight introduced to balance between global and local search and t is the current iteration. Therefore, the particle i updates its current position as follows:

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{3.4}$$

One of the strengths of PSO is the simplicity of the movements rules which make it easy to understand and to program. Unlike the genetic algorithms where

the evolution is based on competition between individuals, the PSO is based on a cooperative evolution in which only the best informative individuals can share its knowledge, which allow to converge towards high quality solutions within a few iterations. Moreover, PSO has less computation efforts compared to GAs. However, PSO usually suffers from the premature convergence problem and being trapped in local optima.

Algorithm 3.4: Particle Swarm Optimization (PSO)

```
Initialize particles population x_i and volicity v_i
Initialize pBest and gBest
while t < iter_max do
    for each particle x_i do
        Update v_i using equation 3.3
        Update x_i using equation 3.4
        Calculate the fitness f(x_i)
        if f(x_i) < f(pBest) then
            pBest[i] = x_i
        end
        if f(x_i) < f(gBest) then
            gBest = x_i
        end
    end
    t = t + 1
end
```

3.6.3 Ant colony algorithm (ACO)

The biologists have demonstrated that the colony of insects is not a "blind" and "stupid" world as some people thought, but it is an impressive world in which insects form an intelligent systems. By observation of an ant colony, the ants can easily find the shortest path from the nest to the food source.

Such behavior is possible via the communication between the ants. Indeed, it has been found that the ant deposits, along its path, a chemical substance called $pheromone$ that can be detected by other ants. In addition, it has been observed that the ants prefer to move towards the path which possess higher pheromone intensity. As the pheromone evaporates over the time, it is clear that the long path will be less marked by pheromone and the shortest path will be more marked. Since that the ants are attracted by highest rate of pheromone, the shortest path has a greater chance to be followed by the whole colony.

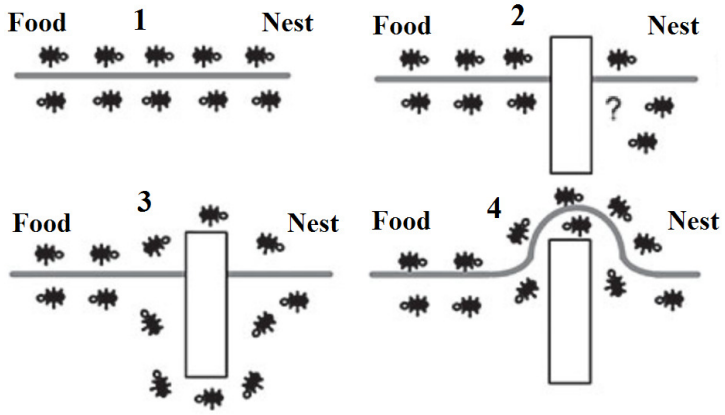

Figure 3.2: Shortest path finding capability of ant colony

Ant Colony Optimization (ACO) is inspired by these observations and it was originally designed for solving traveling salesman problem [Dorigo 1991]; [Dorigo 2006]. ACO deal with a set of agents called $artificial\ ants$. Each ant moves from city to another with probability P_{ij}^k given as follow:

$$P_{ij}^k(t) = \begin{cases} \frac{(\tau_{ij}(t))^\alpha . (\eta_{ij})^\beta}{\sum_{I \in J_i^k} (\tau_{iI}(t))^\alpha . (\eta_{iI})^\beta} \ \ if \ \ j \in J_i^k \\ 0 \quad otherwise \end{cases} \tag{3.5}$$

with, α and β are two parameters which controle the influence of the pheromone. $\tau_{ij}(t)$ is the intensity of pheromone in the path connecting city i and city j. $\eta_{ij} = \frac{1}{d_{ij}}$ is the visibility (d_{ij} is the distance). J_i^k represent the cities not yet visited by the ant k. After a full turn, each ant deposits a quantity of pheromone $\Delta\eta_{ij}^k(t)$ which depends on the quality of the found solution:

$$\Delta\eta_{ij}^k(t) = \begin{cases} \frac{Q}{L^k(t)} \ \ if \ \ (i,j) \in T^k(t) \\ 0 \quad otherwise \end{cases} \tag{3.6}$$

where, $T^k(t)$ is the tour performed by the ant k at iteration t, $L^k(t)$ is the length of $T^k(t)$ and Q is a parameter setting. As well the pheromone is updated to simulate the evaporation behavior:

$$\tau_{ij}(t+1) = (1-\rho).\tau_{ij}(t) + \Delta\tau_{ij}(t) \tag{3.7}$$

where, $\rho \in]0,1]$ is the pheromone evaporation coefficient.

3.6.4 Whale Optimization Algorithm (WOA)

One of the recent evolutionary computation techniques is the Whale Optimization Algorithm (WOA) developed in 2016 by Mirjalili in [Mirjalili 2016a]. WOA is inspired by the bubble-net hunting strategy of humpback whales. These latter attack small fish close to the surface of ocean by creating distinct air bubbles along a circle in a ”9”-shaped path as illustrated in figure 3.3. These air bubbles act as barriers to prevent fish from escaping. Morever, it has been observed that the behavior of bubbles is performed in conjunction with another mechanism so-called encircling of prey. In this mechanism, the whales dive down and then begin to create bubbles and swim toward the surface in a spiral shape around the prey within a shrinking circle.

Similar to previous algorithms, WOA uses a random population of agents

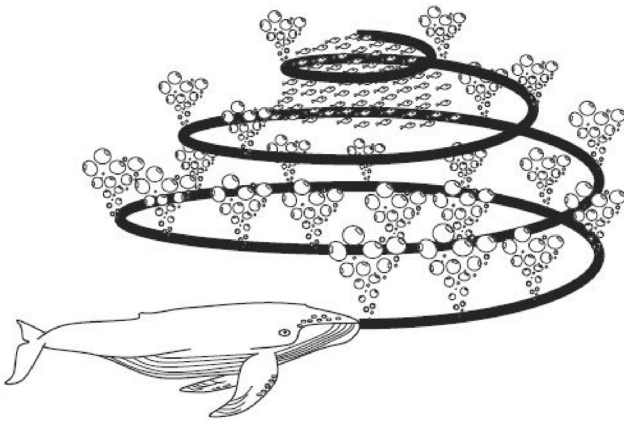

Figure 3.3: Bubble-net hunting behavior of humpback whales [Mirjalili 2016a]

(solutions) and uses three mechanisms to improve the position of the candidate solutions. The encircling mechanism and the spiral model are incorporated for ensuring the exploitation phase, whereas the searching for prey mechanism maintains the exploration phase. The mathematical formulas of WOA algorithm are described in the following paragraphs:

- **Searching and encircling prey**
  The whales attempt to hunt prey by encircling them. Hence, they update their positions with respect to the best hunting agent as follow:

$$D = |C.X^*(t) - X(t)| \tag{3.8}$$

$$X(t+1) = X^*(t) - A.D \tag{3.9}$$

where, $X(t)$ the current position and $X^*(t)$ is the best solution obtained so far. A and C are two vectors given as:

$$A = 2a.r - a \tag{3.10}$$

$$C = 2.r \tag{3.11}$$

$r \in [0, 1]$ and a linearly decrease from 2 to 0 during iterations. If $|A| < 1$, the algorithm promotes the exploitation phase. If $|A| \geq 1$, the exploration phase is favored to perform a global search. Thereby, whales update randomly their positions according to the following equations:

$$D = |C.X_{rand} - X(t)| \tag{3.12}$$

$$X(t+1) = X_{rand} - A.D \tag{3.13}$$

X_{rand} is a point in the search space chosen randomly.

- **Spiral model**
  To imitate the helix-shaped movement, the whales update their positions by using a spiral equation as follow:

$$X(t+1) = X^*(t) + D.e^{bk}.\cos(2\pi.k) \tag{3.14}$$

$$D = |X^*(t) - X(t)| \tag{3.15}$$

D is the distance between the current position and the target prey. k is a random number between -1 and 1 and b is a constant to define the spiral shape.

Whales swim around the prey in a shrinking circle along a spiral-shaped path, this behavior can be formulated mathematically as follows:

$$X(t+1) = \begin{cases} X^*(t) - A.D & if(p < 0.5) \\ X^*(t) + D.e^{bk}.\cos(2\pi.k) & if(p \geq 0.5) \end{cases} \tag{3.16}$$

where, p is a random number between 0 and 1, which indicates that there is probability of 50% to choose between the shrinking encircling mechanism and the spiral system.

The whale optimization algorithm has been shown to be powerful in searching for an optimal solution. Indeed, despite its recent appearance, it has been widely used successfully in different fields such as image processing [Hassan 2018] and electrical engineering [Saha 2017].

3.7 Chapter summary

This chapter has been devoted to explain the basic concepts of optimization problem. This latter is, quite simply, a mathematical modeling of a real-world problem in the form of an objective function subject (or not) to constraints. The resolution of an optimization problem consists in finding the global optimum while respecting the constraints. Due to the NP-completeness nature of optimization problems, there is no exact method able to solve them in reasonable computing time. Thereby, many approximate methods have been proposed, in particular the so-called population-based metaheuristics.

Population-based metaheuristics are based on a stochastic search and manipulate a set of solutions. They have demonstrated a high ability to achieve very satisfactory results in reasonable computation costs by incorporating different diversification and intensification techniques. The aim of diversification techniques (exploration) is to encourage the algorithms to explore more extensively the search space, while the intensification techniques (exploitation) attempts to improve the searching ability to focus on promising regions.

Finally, it should be note that all concepts, definitions and approaches presented in this chapter are related only to single-objective problems. The next chapter will be devoted to one of the most difficult problems which are the multiobjective optimization problems.

CHAPTER 4

Multiobjective optimization: concepts and resolution methods

Contents

4.1 Introduction

In reality, real-world problems are rarely formulated as a single-objective optimization problems because the most problems in practice involve a multiplicity

of criteria rather than single criterion [Deb 2014]. For example, a business owner follows a policy which aimed at increase its profits by minimizing the production cost and maximizing the products quality. This policy involves two criteria, cost and quality, which are clearly contradictory. Indeed, greater product quality often requires higher production costs. Conversely, reducing the costs can results in poor product. In this case, we talk about multi-criteria problem or multiobjective optimization problem.

Multiobjective optimization find its roots in the 19^{th} century in the economic works of Edgeworth and Pareto [Talbi 2012], and since then, it has known an increasing application in several areas. As its name indicates, multiobjective optimization involves several objective functions that are often in conflict. Unlike single-objective optimization where the solution is unique, solving multiobjective optimization problems (MOPs) consists of finding a set of solutions called $optimal\ Pareto\ solutions$. This set represent the best trade-off in which the solutions are balanced based on a dominance relationship. "Balanced solutions" terms means that improving one solution cannot be done without decreasing at least one other solution, which makes the resolution of multiobjective optimization problem more difficult compared to single-objective problem.

4.2 Concepts

We provide in this section the mathematical foundations of multiobjective optimization and the basic concepts related to this research field, in particular, the so-called Pareto dominance notion.

4.2.1 Multiobjective optimization problem formulation

A multiobjective optimization problem (MOP) is defined as a problem for which we attempt to optimize several objective functions while respecting a set of constraints. In mathematical terms, MOP can be formulated as:

$$\begin{cases} Optimize\ F(x) = [f_1(x), f_2(x), ..., f_m(x)],\ m \geq 2 \\ g_i(x) \leq 0,\ i = 1, 2, ..., n \\ h_j(x) = 0,\ j = 1, 2, ..., k \\ x_L \leq x \leq x_U,\ x \in S \end{cases} \tag{4.1}$$

$F(x)$ is a vector composed of m objective functions, $h_i(x)$ are the equality constraints, $g_j(x)$ are the inequality constraints. x is the vector of decision variables. x_L and x_U are, respectively, the lower and upper bounds of the decision variables. S represents the set of feasible solutions associated with all the constraints and the decision space. On the other hand, we denote by $Y = F(S)$ the objective space which corresponds to the image of the feasible space S by the objective function F. In addition, we distinguish by fitness the value of the solution s in the objective space Y.

Unlike single-objective optimization problems where the comparison relation is of total order. If s_1 and s_2 are two candidate solutions, s_1 is better than s_2 if only if $f(s_1) \leq f(s_2)$. In MOP, there is no unique solution, but a set of solutions in which a solution may be better than another on some objectives and worst on the remaining ones. Consequently, the total order relation when comparing between solutions is not useful and another partial order relation should be defined.

4.2.2 Pareto dominance

The concept of Pareto dominance is of fundamental importance in multiobjective optimization when comparing between solutions [Emmerich 2018]. Let u and v be two candidate solutions, we say that u dominates v (denoted by $u \prec v$) if and only if:

$$\forall i \in \{1, 2, ..., m\} : f_i(u) \leq f_i(v) \ and \ \exists j \in \{1, 2, ..., m\} : f_j(u) < f_j(v) \tag{4.2}$$

with, m is the number of objective functions. Figure 4.1 illustrates this dominance relation between five solutions where f_1 and f_2 two objective functions to be minimized.

In this example, the fitness value of x_2 is lower than that of x_4 and x_5 on the two objective functions f_1 and f_2. In this case we say that x_4 and x_5 are **dominated** by x_2. In addition, the solutions x_1, x_2 and x_3 are **nondominated** by any other solution because there is no solution that dominates them on both objective functions.

Définition 4.1 ***(Pareto optimality)*** *A given solution u is called Pareto optimal solution if it is not dominated by any other solution in the feasible search space:*

$$\nexists v \in S \quad | \quad u \prec v \tag{4.3}$$

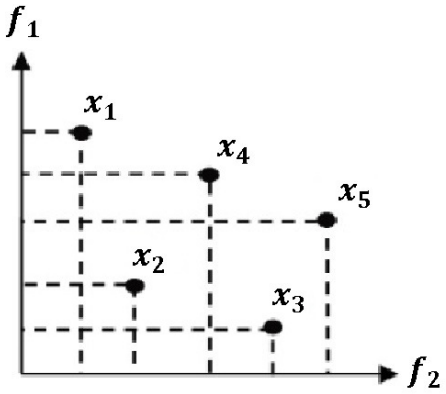

Figure 4.1: Pareto dominance relationship between five solutions

The set of Pareto optimal solutions (or even called nondominated solutions) form the Pareto set PS:

$$PS = \{u \in S | \nexists v \in X, u \prec v\} \tag{4.4}$$

Définition 4.2 ***(Pareto front)*** *The projection of Pareto optimal set (PS) by the objective function F in the objective space Y is called Pareto Front PF:*

$$PF = \{F(x) | x \in PS\} \tag{4.5}$$

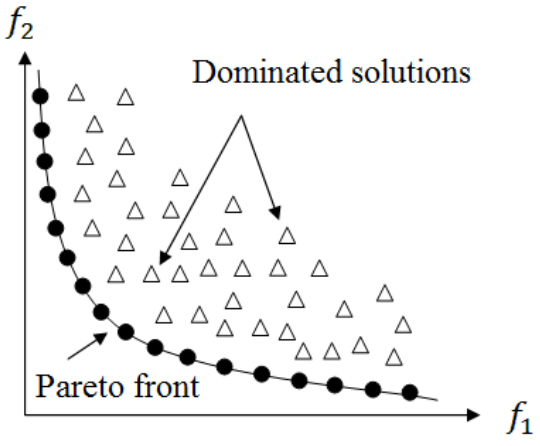

Figure 4.2: Illustration of the Pareto front

4.3 Difficulties of multiobjective problem

The main difficulty of MOP is not due only by the fact that it need the optimization of a multiplicity of criteria, but also because the conflicting design of its objectives. An improvement in one objective degrades the quality at least one of the other objectives. For such situation, there is no single solution, but a set of solutions called Pareto solutions. However, faces of the absence of an exact approach, the general goal of solving MOP is to find a solutions surface very close to the Pareto front. We talk about the convergence towards the true Pareto front.

On the other hand, having a surface very close to the Pareto front is not enough to judge the performance of the resolution method. Indeed, for a given surface, if the choice of a solution risks to considerably decreasing the quality of other objective, this solution is not useful for the decision-maker although it is a Pareto optimal solution. Therefore, it is probably that the solutions are localised in specific zone of the surface. In this case, the solutions are poorly distributed. So, we talk about the diversity of solutions along the Pareto front.

To sum-up, when solving multiobjective optimization problem, two major challenges are taken into account [Talbi 2012]; [Deb 2014]. Firstly, the convergence which consists of finding the front closest to the true Pareto front. Secondly, maintaining the diversity which consists of obtaining well distributed solutions along the obtained front. These two concepts are illustrated in figure 4.3.

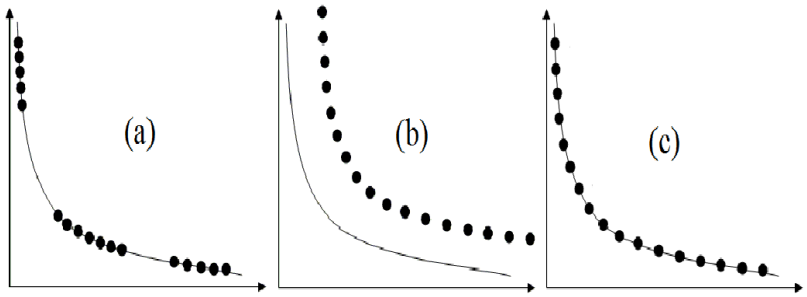

Figure 4.3: Challenges of MOP: (a) diversity (b) convergence (c) true PF

4.4 Classification of multiobjective optimization methods

Due to the growing interest of multiobjective optimization and its usefulness in the real-world, a large number of resolution methods have been proposed which

can be grouped into two types of classification. The first classification adopts a user (or decision-maker) perspective while the second classification adopts a designer perspective.

4.4.1 User's perspective classification

In this classification, the decision-maker can intervene in different steps of the optimization process and choose the method that appears better with respect to his preferences and its desired tasks. There are three possible schemes [Hwang 1980] connecting the decision-maker with the solver according to the intervention level of the decision-maker and its cooperation degree with the optimization method:

- **Priori methods**
  These methods attempt to combine the different objective functions into single utility function based on the preferences of the decision-maker. This amounts to convert the multiobjective problem into single-objective problem. The priori methods enable to the user to specify its preferences in terms of goals or relative importance of different objectives [Marler 2004]. In this case, a prior knowledge is necessary about the problem and about the weight of each objective.

- **Posterior methods**
  This type of methods try to offer a set of solutions for the decision-maker which choose the most preferred configuration among the offered alternatives.

- **Progressive or interactive methods**
  The presence of the decision-maker is essential during the optimization process in order to progressively informing the solver. They enable the decision-maker to provide its preferences during the optimisation process [Rostami 2017]. Therefor, the decision-maker preferences are well taken into consideration so that the most preferred solutions can be achieved.

4.4.2 Designer perspective classification

These methods are classified with respect to the processing manner of the objective functions. We distinguish aggregated methods, non-Pareto methods and Pareto dominance-based methods.

- **Aggregated mathods**
  These methods consist in transforming the multiobjective problem into a single-objective problem by combining the set of objective functions f_i into single-objective function F. There are many aggregation methods such as goal programming [Narasimhan 1980] and min-max method [Rao 1987]. However, The commonly used aggregation is based on the weighted sum which uses an additive model and assigns a weight coefficients which determine the relative importance of each objective function, then summing them [Jin 2002] :

$$F(x) = \sum_{i=1}^{m} w_i f_i(x) \tag{4.6}$$

  w_i : the weight assigned to the objective function f_i with, $w_i \leq 0$ and $\sum_{i=1}^{m} w_i = 1$. The aggregated methods are considered as naive methods, simple and easy to understand. In addition, by the fact that the initial problem is transformed into a single-objective problem, a classical resolution method is sufficient for solving it. However, prior knowledge on the problem is necessary in order to establish the correct weight of each objective.

- **Non-Pareto mathods**
  Unlike aggregated methods, these methods do not transform the original problem into a single-objective problem. Non-Pareto methods (also called non-aggregated methods) attempt to deal with the different objectives separately. One of the most known non-aggregated methods is the Vector Evaluated Genetic Algorithm (VEGA) proposed by [Schaffer 1985].

- **Pareto mathods**
  The Pareto methods are based on Pareto dominance criterion when comparing between solutions. Further, they refuse to separate the objective functions and try to optimize them simultaneously. These methods will be detailed in the next section.

4.5 Pareto dominance-based multiobjective methods

Let us remember that the major goal of any multiobjective optimization method is to maintains both hight convergence and wide diversity of the obtained front. Pareto-based approaches could achieve this aim through the use of the optimality Pareto concept in cooperation with different techniques. This type of approaches generate a set of balanced solutions, that is to say, they are not dominated between them. Thus, no solution is better than the others on the whole criteria of the problem. The choice of the final solution is left to the decision-maker.

In reality, the major effort of scientific community has focused on Pareto-based multiobjective algorithms. However, the most popular and appropriate algorithms for handling MOPs remain, without a doubt, those that comes from metaheuristics based on evolutionary theories such as GAs and PSO. Pareto-based algorithms are commonly grouped on non-elitist algorithms and elitist algorithms.

4.6 Non-elitist algorithms

The term ”non-elitist” means that the elite individuals (Pareto optimal solutions) found during the iterations are not kept. Thus, some relevant solutions might be lost and the convergence towards the Pareto front becomes slower.

4.6.1 Multi Objective Genetic Algorithm (MOGA)

Multi Objective Genetic Algorithm (MOGA) introduced in [Fonseca 1993] is one of the non-elitist algorithms which has a great ability to explore diverse regions of the search space. MOGA algorithm adopts a rank-based fitness assignment which consists to attribute a Pareto rank to each individual in the population based on the number of individuals which dominates it.

Given an individual x_i at the iteration t, dominated by $p_i(t)$ individuals. The rank of this individual is:

$$rank(x_i, t) = 1 + p_i(t) \tag{4.7}$$

All non-dominated individuals (that is to say $p_i(t)$ = 0) are attributed rank 1. Inversely, the dominated individuals are attributed to a high rank. Then, a fitness

value is assigned to each individual so that individuals with the lowest ranks have the best fitness. The traditional fitness assignment may be described as follows:

1. Sort the population according to the rank.

2. Assign the fitness of individuals by interpolation from the best (rank 1) to the worst (rank n) using a scaling function usually linear.

The high pressure exerted by this selection strategy risks to converge towards a single region in the front. To avoid this problem, the authors propose to integrates a sharing function in the fitness value calculation within the objective space rather than the search space in order to uniformly distribute the solutions along the obtained front. MOGA is considered to be an easy algorithm to implement and provides good results. However, MOGA is strongly sensitive to the σ_{share} parameter of the sharing function.

4.6.2 Niched Pareto Genetic Algorithm (NPGA)

This algorithm utilize the concept of Pareto dominance and binary tournament selection operator in conjunction with sharing function [Horn 1994]. In NPGA, two individuals are chosen randomly and compared against subpopulation of size t_{dom} also chosen randomly. If one of the individuals is not dominated, it will be selected to be present in the next population. If the two individuals are non-dominated or dominated, a sharing function (or niching) is applied to determine the winner of the tournament competition while ensuring well spread of solutions. To achieve this, NPGA accepts a degradation of an individual's fitness f_i according to the nich count m_i (i.e, the number of individuals contained in its niche):

$$Sharing - fitness = \frac{f_i}{m_i} \tag{4.8}$$

$m_i = \sum_{j=1}^{N} Sh(d(i,j))$, with $d(i,j)$ is the distance between two individuals and Sh is the sharing function given by:

$$Sh(d(i,j)) = \begin{cases} 1 - \frac{d(i,j)}{\sigma_{share}} & if \quad d(i,j) < \sigma_{share} \\ 0 \quad Otherwise \end{cases} \tag{4.9}$$

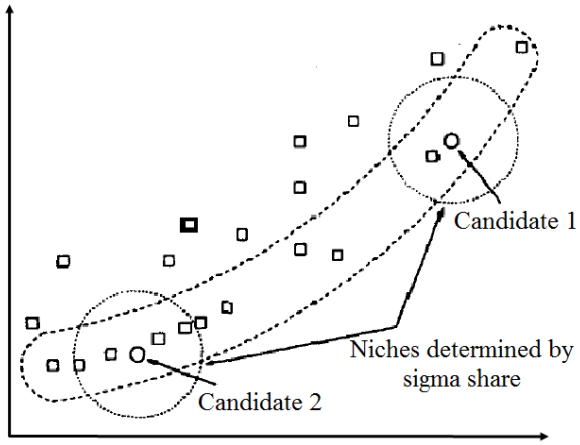

Figure 4.4: Sharing function [Horn 1994]

Unlike MOGA algorithm, the comparison and the sharing function are applied only to a portion of the population which explains the fast convergence of NPGA algorithm compared to MOGA. However, in addition to the σ_{share} parameter, NPGA requires another parameter setting which is the subpopulation size t_{dom}. The authors have shown that a value of 10% can provides good results.

4.7 Elitist algorithms

Elitist algorithms preserve the non-dominated solutions found during the course of optimization by using an external population called archive [Zitzler 2004]; [Laumanns 2001]. The elitism concept can really improve the performance of the algorithm [Fieldsend 2003]. However, at some point, the number of Pareto solutions may exceed the size of the archive. Therefore, an archive update strategies are usually employed to manage the archive and also to maintain the diversity.

4.7.1 Pareto Archived Evolutionary Algorithm (PAES)

The Pareto Archived Evolution Strategy (PAES) [Knowles 1999] is an oldest elitist optimizer. Instead the use of population of solutions, PEAS utilize one individual during the search process. To ensure a good approximation towards the Pareto front, PAES algorithm maintains an external archive of non-dominated solutions

to estimate the quality of the new candidate solutions.

In PAES, the current solution c_i generates by random mutation a new candidate solution m_i. If m_i dominates c_i, m_i is accepted. If c_i dominates m_i, the generated solution m_i is rejected. If neither solution dominates, the generated solution m_i is compared with the archive members. To generate well-distributed Pareto frontier, PAES algorithm uses a new grid location based on recursively dividing the objective space in 2^{kd} hypercube, where k is the number of bisecitons of the space carried out for each objective and d is the number of objective functions [Knowles 2000].

Compared to previous algorithms, The PAES algorithm eliminates the use of the niche size parameter because it incorporates an efficient adaptive grid method instead the sharing function. Nonetheless, the effectiveness of PAES algorithm is very sensitive to the subdivision parameter of the objective space.

4.7.2 Strength Pareto Evolutionary Algorithm 2 (SPEA2)

SPEA2 algorithm was introduced by [Zitzler 2001] at the aim to correct the shortcomings observed in SPEA algorithm proposed by the same authors [Zitzler 1998]. Three major points that make the difference between SPEA and SPEA2: fitness assignment, density estimation and the archiving strategy.

In addition of the main population P, SPEA2 utilize an external archive A of fixed-size T_{max} that stores the non-dominated solutions, and it adopts an efficient fitness assignment strategy which take into consideration, for each individual, both the individuals it dominates and individuals by which it is dominated. At each iteration and for each individual i in the set $(P \cup A)$, the algorithm calculate the strenght value $S(i)$ representing the number of solutions it dominates:

$$S(i) = |\{j/j \in \{P \cup A\},\ i\ dominate\ j\}| \tag{4.10}$$

Then, the raw fitness $R(i)$ is calculated using $S(i)$ value :

$$R(i) = \sum_{j \in \{P \cup A\}, j\ dominate\ i} S(j) \tag{4.11}$$

$R(i) = 0$ corresponds to a nondominated solution. In order to preserve the diversity of the population, the authors propose to added a density estimation measure $D(i)$ to the raw fitness $R(i)$, which yields the final fitness $F(i)$:

$$F(i) = R(i) + D(i) \tag{4.12}$$

where, $D(i)$ is estimated by the k^{th} nearest neighbor method as follows:

$$D(i) = \frac{1}{\sigma_i^k + 2} \tag{4.13}$$

σ_i^k is the distance between individual i and the k^{th} nearest neighbor, with $k = \sqrt{N + T_{max}}$.

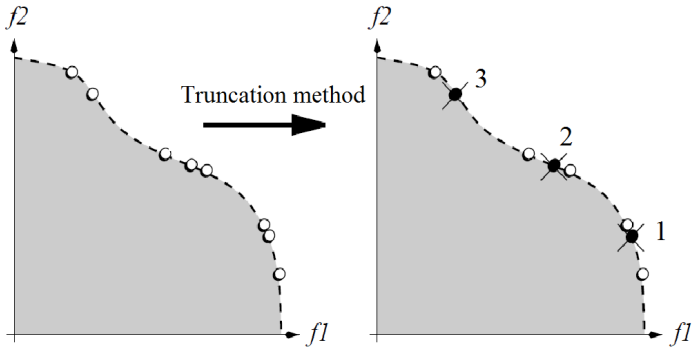

Figure 4.5: Truncation method used in SPEA2 [Zitzler 2001]

To update the archive, SPEA2 uses a truncation technique while improving the spread of non-dominated solutions. The first step is to copy all the non-dominated individuals from population P and archive A to the archive of the next generation. If the number of non-dominated individuals is less than the archive size, the best dominated individuals in $(P \cup A)$ are inserted in the archive. If the number of non-dominated individuals exceeds the archive size, the truncation procedure is invoked for iteratively removes the individual which has the minimum distance to another individual. If there are several candidate solutions with the same minimum distance, the decision is made by considering the second smallest distance. This truncation procedure is illustrated in figure 4.5 for an archive size of five elements. Further, algorithm 4.1 provides the pseudocode of SPEA2.

4.7.3 Non Dominated Sorting Genetic Algorithm (NSGA2)

On of the most popular and the most cited multiobjective algorithms in the literature is the famous Non Dominated Sorting Genetic Algorithm (NSGA2) proposed by [Deb 2002a]. The basic idea of NSGA2 is the use of a fast-non-dominated sorting ranking scheme to create several non-dominated fronts in order to de-

Algorithm 4.1: SPEA2 algorithm

```
Initialize population P
Create empty external archive A, |A| = N
for i = 1 to MaxIter do
    Evaluate each individual in P and A
    A=Nondominated(P) ∪ Nondominated(A)
    if |A| < N then
        Complete the archive A with dominated solutions in P
    else
        Remove elements from A using the truncation operator
    end
    Perform binary tournament selection
    Apply crossover and mutation
end
```

termine the rank of each individual in the population. For each solution i, the non-dominated sorting procedure starts by calculating the number of solutions n_i that dominate the solution i and determine the set of solutions S_i dominated by the solution i. The solutions with $n_i = 0$ form the first level F_1. Then, for each element i in F_1, n_j value should subtract 1 ($n_j = n_j - 1$ with $j \in S_i$). Each solution j whose $n_j = 0$ is saved in a separate list H. Therefore, The solutions contained in the list H form the second level F_2. The same reasoning is followed and applied on the members of the list H to identify the third level. This procedure is repeated until all solutions are processed and all fronts are identified.

To maintain the diversity in the population, NSGA2 uses a new crowded-comparison operator based on a novel density measure called "crowding distance". As shown in figure 4.6, the crowding distance try to estimate the density of solutions surrounding a particular solution by computing the average side-length of the cuboid formed by the two points closest to this solution. Accordingly, each individual i is characterized by two attributes:

- i_{rank} which corresponds to the front in which the individual i belongs (if the individual i belongs to the front F_1 it is assigned an $i_{rank} = 1$).

- $i_{distance}$ which represents the crowding distance of the individual i.

So, the crowded-comparison operator (denoted by $\prec_n$) is defined as :

$$i \prec_n j \quad if \quad (i_{rank} < j_{rank}) \vee ((i_{rank} = j_{rank}) \wedge (i_{distance} > j_{distance})) \tag{4.14}$$

In other words, when comparing between two solutions, the chosen solution is the one with the lower $rank$. If the two solutions have the same $rank$, the chosen solution is the one with highest crowding distance, which enable to obtain well distributed Pareto front.

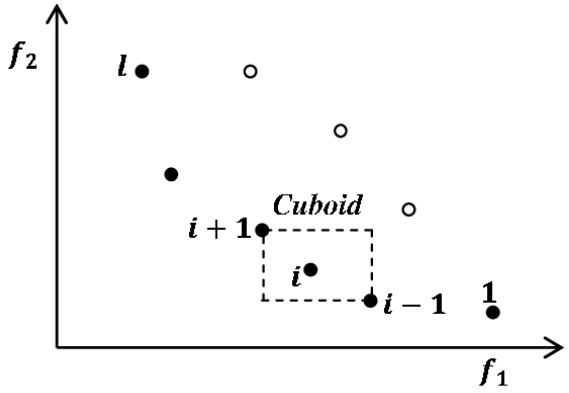

Figure 4.6: Crowding distance calculation

Algorithm 4.2 provides the pseudocode of NSGA2. At each generation t, an offspring population Q_t of size N is generated from the parent population P_t of identical size. The two populations are combined in an external population $R_t = P_t \cup Q_t$ of size $2N$. The individuals of this population are then sorted into different front levels using non-dominated sorting method to create the next population P_{t+1} by integrating successively the fronts F_i, starting from the first front F_1 to the last front. However, the population R_t is of size $2N$ and P_{t+1} is of size N. Therefore, all the fronts cannot be integrated into P_{t+1}. Indeed, $(N - |P_{t+1}|)$ solutions are selected from the last front (to be preserved) to complete the population P_{t+1}. To do this, the solutions of the front to be preserved are organized in descending order according to their crowding distance. Then, the best $(N - |P_{t+1}|)$ solutions are selected. This selection maintains the elitism concept while preserving the non-dominated solutions found during iterations. Finally, the new parent population P_{t+1} is subjected to crossover and mutation operators to reproduce the new offspring population Q_{t+1}.

NSGA2 replaces the sharing function with a non-parametric density estimator.

In addition, it uses an effective and fast elitist strategy that allow, on the one hand, to reduce the complexity of the algorithm to $O(mN^2)$, and on the other hand, to get closer to the true Pareto front and preserve diversity without using an external archive. NSGA2 algorithm is considered as a reference algorithm and it is widely used not only for solving multiobjective optimization problems, but also as benchmark to test against.

Algorithm 4.2: NSGA2 algorithm

Generate P_0 at random
$P_0 = (F_1, F_2, ...) = fast - nondominated - sort(P_0)$
Q_0=(Selection, Crossovor, mutation)(P_0)
$|P_0|$=$|Q_0|$ = N
$t = 0$
while $t < MaxIter$ **do**
    $R_t = P_t \cup Q_t$
    $F = fast - nondominated - sort(R_t)$
    $P_{t+1} = \emptyset$ and $i = 1$
    while $|P_{t+1}| + |F_i| < N$ **do**
        $crowding - distance - assignment(F_i)$
        $P_{t+1} = P_{t+1} \cup F_i$
        $i = i + 1$
    end
    Sort($P_{t+1}, \prec_n$)
    $P_{t+1} = P_{t+1} \cup F_i[1 : (N - |P_{t+1}|)]$
    Q_{t+1}=make-new-pop(P_{t+1})
    $t = t + 1$
end

4.7.4 Multiobjective Particle Swarm Optimization (MOPSO)

Although there are many multiobjective versions of Particle Swarm Optimization (see survey papers [Reyes-Sierra 2006]; [Zhou 2011]), but the Multi-Objective Particle Swarm Optimization (MOPSO) proposed in [Coello 2004] that we will present below, is considered as the most popular multiobjective PSO-based and the most cited in the literature.

MOPSO integrates the Pareto optimality into the PSO to make it able to handle several objective functions. As well, it maintains a fixed-size repository (archive) to save the best flights performed by the particles so far. The repository is then used to guide the swarm towards the Pareto front. In order to avoid stagnation in a local front, the authors propose to apply a turbulence operator (mutation) while

increasing the exploration ability of PSO.

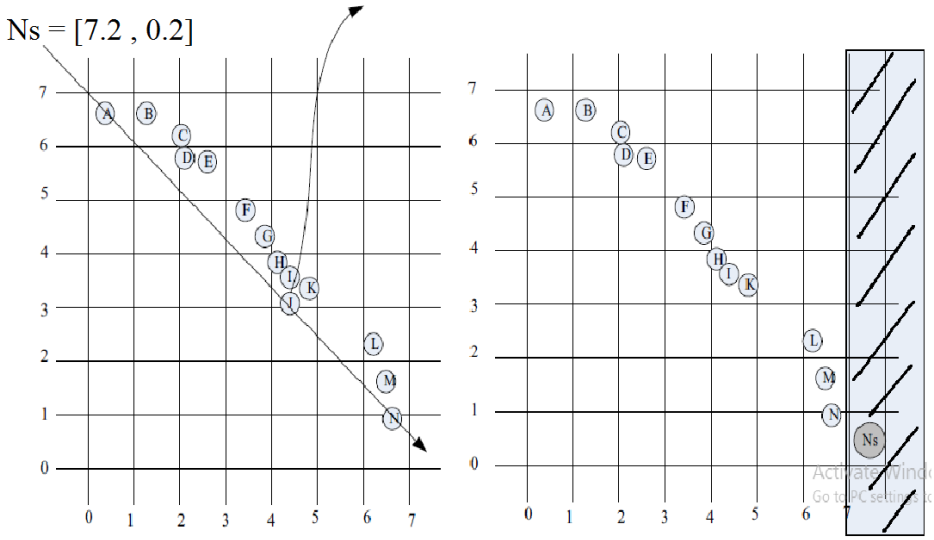

Figure 4.7: Illustration of adaptive grid procedure [Coello 2004]

To update the archive, the algorithm incorporates a controller whose role is to accept a solution to be inserted in the archive, or to eliminate some solutions from the archive. Each candidate solution is compared with the elements of the archive. If the archive is empty, the solution is accepted. If the solution is dominated by an element of the archive, it is therefore rejected. If this solution dominates some elements of the archive, these elements will be deleted from the archive. Finally, if the external archive reaches its maximum size, the most crowded solutions should be removed using an adaptive grid procedure.

To maintain the diversity of the swarm, MOPSO algorithm uses an adaptive grid (similar to PAES algorithm) represents the space formed by hypercubes that can be interpreted as a geographic regions having several solutions. The particles of the external population can be inserted outside the current limits of the grid. In this case, the grid is recalculated again with each addition of non-dominated solution (see figure 4.7).

Remember that in the standard PSO, the particles always try to follow the best particle $gBest$ in the swarm. In MOPSO, the $gBest$ (called leader) is selected by dividing the explored space into hypercubes. Each hypercube is assigned a fitness value that is inversely proportional to the number of particles it contains. Then, a selection by roulette is first applied on the fitness values to identify the hypercube

from which the leader will be choosen. Once the hypercube is identified, a random selection is performed to choose the leader.

Algorithm 4.3: Pseudocode of a general MOPSO [Reyes-Sierra 2006]

```
Initialize swarm
Initialize leaders in an external archive
Quality(leaders)
g = 0
while g < gmax do
    for each particle do
        Select leader
        Update Position (Flight)
        Mutation
        Evaluation
        Update pbest
    end
    Update leaders in the external archive
    Quality(leader)
    g++
end
Report results in the external archive
```

4.8 Performance metrics

To evaluate the quality of any multiobjective optimizer, several performances metrics are used to quantify the convergence and the diversity of the obtained Pareto front, and also to visualize the characteristics of the solutions set obtained. In our thesis, the following three performance metrics are employed:

- **Inverted Generational Distance metric (IGD)**
  The IGD-metric [Van Veldhuizen 2000] uses as a reference the true Pareto front to calculate the distance between each reference point and the nearest nondominated solution in the Pareto front, and it is formulated as follows:

$$IGD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n} \tag{4.15}$$

  where n is the number of solutions in the true Pareto front and d_i is the Euclidean distance between the i^{th} solution in the true Pareto front and the nearest member of the Pareto front obtained by an algorithm. The IGD-metric is frequently used to measure the convergence of the Pareto front obtained by an algorithm.

- **Spacing (Sp) metric**
  In order to quantitatively compare the algorithms in terms of diversity, the Spacing (Sp) metric is commonly used to estimate how far the obtained non-dominated solutions are evenly distributed [Schott 1995]. Sp-metric is defined as:

$$Sp = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (\bar{d} - d_i)^2} \tag{4.16}$$

  where d_i is defined as:

$$d_i = min_j(\sum_{k=1}^{m} |f_m^i - f_m^j|) \tag{4.17}$$

  i, j = 1...n and n is the number of solutions in the front obtained by the algorithm, $\bar{d}$ is the average of all d_i and m is the number of objective functions.
  It should be noted that the smaller values of IGD and SP indicates better performance to the compared algorithm.

- **Set coverage metric C**
  This metric is usually employed for comparing between two non-dominated Pareto sets A and B and is given as follows :

$$C(A, B) = \frac{|\{b \in B | \exists a \in A : a \succeq b\}|}{|B|} \tag{4.18}$$

This quantity corresponds to the proportion of the solutions of the set B, that are weakly dominated by at least one solution of A. It should be noted that $C(A, B)$ is not necessarily equal to $1 - C(B, A)$. To compare A and B sets, both values of $C(A, B)$ and $C(B, A)$ must be calculated.

4.9 Chapter summary

This chapter offers a brief state-of-the-art on multiobjective optimization problems (MOPs) and their resolution methods. Over the past two decades, MOP has becomes the interest's center of many researchers because the majority of real-world problems involve multiple contradictory criteria. For a long time, the conversation of multiobjective problem into a single-objective problem was the most intuitive and simplest reasoning to solve such problem. However, this method requires prior knowledge about the problem, which is not always the case. Furthermore, these methods are turns out to be useless and ineffective because they are not able to provides a set of solutions and they cannot optimize simultaneously whole the objective functions. These drawbacks gave birth to other methods which do not requires any prior knowledge on the problem and optimize simultaneously entire the objective functions while obtaining a set of optimal solutions.

Pareto-based algorithms can obtain a set of solutions closer to the Pareto optimal solutions and they are able to distribute uniformly the obtained set in the search space. Consequently, they offer to the decision-maker a set of relevant and useful trade-off configurations. In the last decades, several Pareto-based algorithms have been proposed, and which have showed a great performance, in particular, those which maintain the elitist concept. These algorithms use often an external memory to store the best configurations and apply different updating techniques in order to converge towards promising regions in terms of convergence and diversity.

In the remain, we will present the two contributions that we have achieved during this thesis. Therefore, the next chapter is devoted to the multiobjective optimization method that we have developed. The last chapter is dedicated to the application of the proposed algorithm on machine learning.

CHAPTER 5

GPAWOA for multiobjective optimization problems

Contents

5.1 Introduction

In the last few years, several population-based algorithms have been proposed for solving multiobjective optimization problems such as NSGA2 [Deb 2002a] and SPEA2 [Zitzler 2001]. The remarkable success of these algorithms has motivated researchers to propose other alternatives. For example, Multiobjective Evolutionary Algorithm Based on Decomposition (MOEA/D) proposed in [Zhang 2007] where the MOP is decomposed into a set of single objective optimization subproblems. The objective of each subproblem is a (linearly or nonlinearly) weighted aggregation of the individual objectives. Neighborhood relations among these

subproblems are defined based on the distances between their aggregation weight vectors. In MOEA/D, each subproblem is optimized by using the information of its neighboring subproblems.

Recently, many multiobjective optimization algorithms have been published. These proposals focus mainly on the extension of exisiting optimization algorithms that are initially developed to solve single-objective problems in order to make them able to deal with multiobjective optimization problems. Since the first version multiobjective PSO-based proposed by Moore and Champan [Moore 1999], several versions of MOPSO have been proposed such as the Swarm Metaphor [Ray 2002] where Pareto dominance is used and the concepts of evolutionary techniques are combined with PSO. To maintain diversity, the crowding technique is applied and multilevel sieve is employed to handle constraints. Nondominated Sorting Particle Swarm Optimizer (NSPSO) proposed in [Li 2003] incorporates the main mechanisms of NSGA2 into PSO. This algorithm combines the population of particles and the best personal position of each particle into a temporary population, and from this population, the best are selected to form the next population. To maintain diversity, NSPSO applied both the niche count and the crowding distance. Ant Colony Optimization (ACO) [Dorigo 1991] was also originally proposed to deal with single-objective optimization problems and which was afterward extended to solve MOPs [Alaya 2007]; [Angus 2007]; [Doerner 2004].

Over the past years, other multiobjective optimization algorithms have been published, the basic and common idea of these proposals is simply: extending of existing algorithms developed initially to solve single-objective problems in ordre to be able to deal with multiple objectives by incorporating different techniques and mechanisms. Among them we can cite the following algorithms: Multi-Objective Cat Swarm Optimization (MOCSO) [Pradhan 2012] by extending the Cat Swarm Optimization (CSO) [Chu 2006]. Multi-Objective Grey Wolf Optimizer (MOGWO) [Mirjalili 2016b] which is another prominent work which extend the recent Grey Wolf Optimizer (GWO) [Mirjalili 2014].

This chapter proposes a new multiobjective optimization algorithm called A Guided Population Archive Whale Optimization Algorithm (GPAWOA) by extending the recent nature inspired Whale Optimization Algorithm (WOA) [Mirjalili 2016a] to deal with multiple objectives. In this algorithm, we integrate the concepts of Pareto dominance, updating strategy and density estimation into the standard WOA. The rest of the chapter is organized as follows: The proposed GPAWOA is detailed in section 5.2. Section 5.3 summarizes extensive experimental evaluation.

5.2 The proposed algorithm

In the following, we present the proposed algorithm GPAWOA which is based on the Pareto dominance and employs an external archive as a repository to store the nondominated solutions found so far and to guide the main population towards a well accurate approximation of the true Pareto front. Over the course of iterations, the number of nondominated solutions increase rapidly, therefore, the size of the archive also increase and this affects, definitely, on the complexity of the algorithm. To avoid this problem, the external archive is limited to a maximum size T_{max}.

For a good performance in terms of convergence and diversity, two important strategies are taken into account when designing our algorithm:

i Adopting an efficient strategy of updating the external archive in order to improve the convergence of the obtained front.

ii Adopting an efficient leader selection strategy in order to guide the population agents towards promising and sparse regions within the search space.

To obtain an uniform distribution of the nondominated solutions, these two strategies employ the mechanism of crowding distance computation as a diversity criterion. Additionally, the crowding distance is used again in the deletion operation from the archive whenever it is full.

5.2.1 Updating the archive

The archiving strategy is an important step for a good performance of any multiobjective algorithm. In our algorithm, the archive is updated as follows: at each iteration, we create a repository which contains the union of the current population and the current archive, then we determine all the nondominated solutions of this repository, and store them in the new archive. In this way, we ensure that the external archive prepared to the next iteration contains only the nondominated solutions.

After updating the archive, the number of nondominated solutions might exceed the maximum size of the archive T_{max}. In this case, we use the crowding

distance technique to reduce the size of the archive to its maximum size as follows:

i The crowding distance of each nondominated solution in the archive is calculated.

ii The nondominated solutions are sorted in the archive in decreasing order according to their crowding distance.

iii The most crowded solution is removed from the archive, this step is repeated until the archive reaches its maximum size T_{max}.

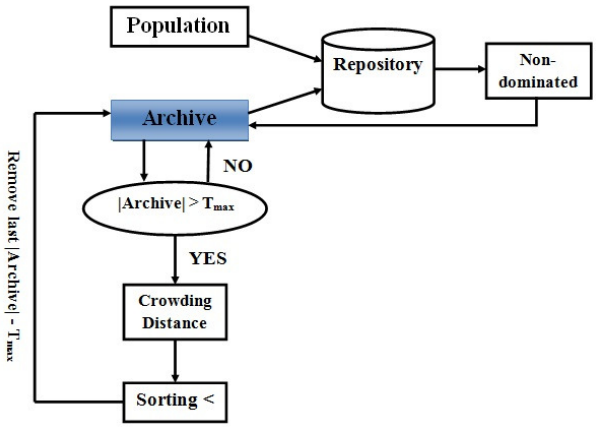

Figure 5.1: flowchart of the archiving strategy

5.2.2 Density estimator

In order to maintain uniformly distributed solutions in the obtained front, we incorporate a widely-used crowding distance computation mechanism [Deb 2002a] as a density estimator of solutions surrounding a particular solution in the archive by computing the average side-length of the cuboid formed by the two nearest neighbors of this solution (see figure 4.6 in chapter 4).

The distance of each solution in the archive is initialized to 0. For each objective m, the nondominated solutions are sorted in ascending order according to their values in the objective m. The boundary solutions which have the lowest and highest objective values are assigned an infinite crowding distance. The crowding distance of the other solutions in the archive is calculated by computing the difference between the two nearest neighbors of the solution for the corresponding objective function m, and sums it to the solution crowding distance.

Algorithm 5.1: Crowding distance computation algorithm

```
A is the external archive
I = |A|
for i ∈ A do
    A[i]_distance = 0
end
for each objective m do
    A=Sort(A,m)
    A[1]_distance = A[I]_distance = ∞
    for i = 2 to (I − 1) do
        A[i]_distance = A[i]_distance + (A[i + 1]_m − A[i − 1]_m)
    end
end
```

To produce well-distributed solutions on the obtained front, our approach uses the crowding distance computation in two steps of the algorithm:

Step 1. In the archiving strategy (as discussed previously).

Step 2. In the leader selection, we use the crowding distance once again to choose a leader from the archive in order to guide the population towards well-spread regions.

5.2.3 Leader selection strategy

The leaders are the individuals that guide the population towards a better approximation of the true Pareto front and towards well-distributed regions, which makes the selection of an appropriate leader a crucial step to further improve the performance of the algorithm. In GPAWOA, the leader is selected as follows:

- Calculate the crowding distance of each solution in the external archive.
- Sort the archive members in descending order according to their crowding distance.
- Determine the higher part of the sorted archive (highest 10 % of the sorted archive) which contains the less crowded solutions.

- From the upper part of the sorted archive determined in the previous step, select randomly a solution as a leader for each individual to guide it towards the least crowded space in order to improve the spread of the achieved Pareto front.

We note that the mathematical model used to update the individuals position is similar to that used in the standard WOA, the difference is in the Equations (3.8, 3.9, 3.15 and 3.14 in chapter 3) where the best solution X^* becomes the leader chosen from the archive, so these equations can be rewritten as follows:

$$D = |C.leader(t) - X(t)| \tag{5.1}$$

$$X(t+1) = leader(t) - A.D \tag{5.2}$$

$$D = |leader(t) - X(t)| \tag{5.3}$$

$$X(t+1) = leader(t) + B.e^{bk}.\cos(2\pi.k) \tag{5.4}$$

5.2.4 Outlines of GPAWOA algorithm

The algorithm 5.2 summarizes how the Guided Population Archive Whale Optimization Algorithm (GPAWOA) works.

The GPAWOA algorithm starts by generating randomly the set of search agents. Based on the concept of Pareto dominance, the algorithm identifies the nondominated solutions and store them in the external archive. After preparing the archive to the first iteration, the algorithm enters in the main loop: In each iteration, the crowding distance of each solution is calculated, and then all the nondominated solutions are sorted in descending order with respect to their crowding distance. For each search agent, the A, C parameters are updated using Equations 3.10 and 3.11 respectively. Then, the algorithm selects randomly a leader for the current search agent from the top part of the sorted archive. Depending on the values of p and A, the algorithm switches between three cases to update the position of the current search agent using Equations 5.2, 3.13 and 5.4 (as shown in Algorithm 5.2). After updating the position of all the search agents of the population, the algorithm updates the archive and checks whether the archive is full or not to prune it to its allowable size T_{max}. This main loop is repeated until the maximum number of iterations $iter\_max$ is reached. Finally, the algorithm returns the external archive which contains the final Pareto front.

Algorithm 5.2: Pseudocode of GPAWOA

```
Initialize the whales population X_i = (1, 2, ..., n)
Evaluate each search agent in X_i by calculating its fitness value
Initialize Archive Ar (with maximum size equal to T_max)
while t < iter_max do
    Calculate the crowding distance of each solution in Ar
    Sort Ar in descending order according to crowding distance values
    for each search agent do
        update a, A, C, k and p
        randomly select a leader from the top part of the sorted archive.
        if (p < 0.5) then
            if (|A| < 1) then
                calculate D using Equation 5.1
                update the position of the current search agent by using the
                  Equation 5.2
            else
                select a random search agent X_rand
                calculate D using Equation 3.12
                update the position of the current search agent by using the
                  Equation 3.13
            end
        else
            calculate D using Equation 5.3
            update the position of the current search agent by using the
              Equation 5.4
        end
    end
    update the Archive Ar.
    if (the archive Ar is full) then
        calculate the crowding distance of each solution in Ar.
        sort Ar in descending order according to crowding distance values.
        remove the last solution in the sorted archive Ar.
    end
    t = t+1.
end
Return Ar.
```

5.2.5 Complexity of GPAWOA

In terms of computational complexity, GPAWOA performs one loop and two basic operations when considering the complexity of one iteration. Let N the number of individuals in the main population and in the external archive, and M the number of objectives. The loop has an $O(N)$ complexity (the same complexity of the single-objective Whale optimization Algorithm). The first operation (updating archive) need $M(N+N) = 2MN$ comparisons, so the complexity of this operation in its worst case is $O(MN^2)$. The second operation (crowding distance computing) has an $O(2Nlog(2N))$ complexity [Deb 2002a]. Therefore, the complexity of GPAWOA is $max(O(N), O(MN^2), O(2Nlog(2N))) = O(MN^2)$. This complexity is equal to other well-known multi-objective meta-heuristics such as NSGA-II [Deb 2002a], SPEA2 [Zitzler 2001], MOPSO [Coello 2004] and MOGWO [Mirjalili 2016b]. However, the complexity of GPAWOA is higher than that of MOEA/D which is equal to $O(mNT)$ (T: the number of weight vectors in the neighborhood) [Zhang 2007].

5.3 Experimental results and comparisons

To evaluate the performance of the proposed GPAWOA algorithm, we use two sets of benchmark functions taken from the specialized literature : five bi-objective problems (ZDT-series [Zitzler 2000]) and seven tri-objective problems (DTLZ-series [Deb 2002b]). The characteristics of muli-objective test functions are listed in Table 5.1. Moreover, the GPAWOA is applied to four engineering design problems. In order to know the competitivity of our approach, we compare qualitatively and quantitatively its results with respect to three well-regarded algorithms: MOEA/D [Zhang 2007], MOPSO [Coello 2004] and MOGWO [Mirjalili 2016b]. For each test function, we performed 30 independent runs per algorithm with the parameters settings summarized in Table 5.2. All algorithms were run using a population size of 100, an archive size of 100, the number of generations is set to 5000 for ZDT test functions and 30.000 for DTLZ test functions.

For qualitative comparison, we plot the best Pareto front obtained by all algorithms over the 30 runs. In order to perform the quantitative comparison, three widely used performance metrics (detailed in chapter 4 section 4.8) are employed to quantify the convergence and the diversity of the Pareto front obtained by an

Table 5.1: Characteristics of multi-objective test functions

Bi-objective test function	
Test function	**Characteristics**
ZDT1	with a convex front
ZDT2	with a non-convex front
ZDT3	with a discontinuous front
ZDT4	with 2^{21} local Pareto-optimal fronts and therefore is highly multi-modal
ZDT6	with a non-uniform search space
Three-objective test functions	
Test function	**Characteristics**
DTLZ1	with a linearl Pareto-optimal front
DTLZ2	with a spherical Pareto-optimal front
DTLZ3	with a many Pareto-optimal fronts
DTLZ4	with Pareto-optimal front has dense set of solutions to exist near the $f_M - f_1$
DTLZ5	This problem will verify the ability of MOEA to converge to a degenerated curve.
DTLZ6	This problem has 2^{M-1} disconnected Pareto-optimal front.
DTLZ7	This problem has Pareto-optimal front which is a combination of a straight line and a hyper-plane.

Table 5.2: Parameters of algorithms

Algorithm	**Parameters**
MOPSO	$\phi_1 = \phi_2 = 2.05$, $\phi = \phi_1 + \phi_2$, Inertia weight: $w = \frac{2}{\phi - 2 + \sqrt{\phi^2 - 4\phi}}$ Personal coefficient: $c_1 = w * \phi_1$, Social coefficient: $c_2 = w * \phi_2$, Number of grids per each dimension: $nGrid = 10$
MOGWO	Grid inflation : $\alpha = 0.1$, Archive member selection pressure: $\gamma = 2$ Grids number per each dimension: $nGrid = 10$, Leader selection pressure parameter : $\beta = 4$
MOEA/D	Subproblems: $N = 100$, Mutation rates: $CR = F = 0.5$ Number of neighborsr: $T = 10$, Distribution index : $\eta = 30$ Probability that parent solutions are selected from the neighborhood : $\delta = 0.9$ Maximal number of solutions replaced by each child solution : $n_r = 1$
GPAWOA	$b = 1$, Parcentage which determine the top part of the archive : $a = 0.1$

algorithm. The algorithms MOEA/D, MOPSO, MOGWO and GPAWOA were run on Matlab2016b programming environment under 32-bits Windows 8.1, the experimental tests were performed on the same computer: Intel Core i3, 2.39 GHz and 4 GB RAM.

5.3.1 Results on ZDT test functions

The statistical results of IGD and Spacing(Sp) metrics obtained by the different algorithms after 30 runs are summarized in Tables 5.3 and 5.4 where the best results (minimum average of each metric) are shown in **boldface**. In addition, we illustrate in figures 5.2-5.6, the graphical Pareto fronts corresponding to the lowest IGD-value obtained by each algorithm.

Table 5.3: Statistical results for IGD on ZDT test functions

Algorithm	best	worst	median	average	Std.
ZDT1					
MOEA/D	7.068.E-4	7.154.E-3	9.614.E-4	1.454.E-3	1.581.E-3
MOPSO	6.271.E-4	1.104.E-3	5.791.E-4	5.385.E-4	2.481.E-4
MOGWO	4.866.E-4	7.410.E-4	5.404.E-4	5.641.E-4	2.501.E-4
GPAWOA	4.236.E-4	5.907.E-4	4.223.E-4	**4.180.E-4**	5.120.E-5
ZDT2					
MOEA/D	2.110.E-3	9.195.E-2	2.521.E-2	3.127.E-2	2.371.E-2
MOPSO	6.852.E-4	1.154.E-3	6.114.E-4	7.068.E-4	1.179.E-4
MOGWO	4.112.E-4	7.233.E-4	6.853.E-4	5.341.E-4	1.667.E-5
GPAWOA	4.445.E-4	6.567.E-4	5.074.E-4	**4.048.E-4**	4.431.E-5
ZDT3					
MOEA/D	1.091.E-2	1.935.E-2	1.430.E-2	1.466.E-2	2.958.E-3
MOPSO	4.417.E-3	2.051.E-2	1.120.E-2	1.157.E-2	4.221.E-3
MOGWO	4.823.E-4	7.104.E-4	5.528.E-4	**5.600.E-4**	1.791.E-4
GPAWOA	1.113.E-3	2.159.E-3	7.113.E-3	1.544.E-3	4.105.E-4
ZDT4					
MOEA/D	1.745.E-2	6.297.E-1	8.991.E-2	1.264.E-1	1.336.E-1
MOPSO	4.031.E-2	7.321.E-1	2.638.E-1	2.578.E-1	1.465.E-1
MOGWO	5.977.E-4	7.963.E-4	5.219.E-4	5.353.E-4	1.311.E-4
GPAWOA	4.511.E-4	6.497.E-4	5.408.E-4	**4.843.E-4**	2.281.E-5
ZDT6					
MOEA/D	5.135.E-4	1.598.E-1	8.348.E-4	2.491.E-2	5.191.E-2
MOPSO	3.290.E-4	6.864.E-4	4.833.E-4	**4.025.E-4**	1.440.E-4
MOGWO	3.067.E-4	7.122.E-4	4.882.E-4	5.042.E-4	1.491.E-5
GPAWOA	3.119.E-4	9.702.E-4	5.975.E-4	6.257.E-4	2.345.E-4

From Table 5.3 which shows the comparison among the four algorithms considering IGD-metric, it can be seen that our approach accomplished the best average IGD in ZDT1, ZDT2 and ZDT4 test problems, which means that our approach outperforms the other algorithms in three among the five test problems. Figures 5.2-5.3 show the graphical results of ZDT1 and ZDT2 respectively. It can be seen that GPAWOA produce a good convergence toward the true Pareto optimal front. For ZDT4 test functions, as it can be easily seen in figure 5.5, GPAWOA algorithm is significantly better than MOEA/D and MOPSO in terms of convergence.

Also, the results of Table 5.3 indicate that GPAWOA is better than MOEA/D and MOPSO on ZDT3 and ranks second after the MOGWO algorithm. In addition, figure 5.4 which illustrates the Pareto fronts produced by all algorithms on ZDT3,

Table 5.4: Statistical results for Sp on ZDT test functions

Algorithm	best	worst	median	average	Std.
ZDT1					
MOEA/D	6.407.E-3	2.071.E-2	1.474.E-2	1.466.E-2	4.368.E-3
MOPSO	6.511.E-3	1.120.E-2	8.065.E-3	7.815.E-3	1.703.E-2
MOGWO	5.702.E-3	1.062.E-2	7.612.E-3	7.901.E-3	1.245.E-3
GPAWOA	5.411.E-3	1.025.E-2	7.715.E-3	**7.595.E-3**	6.022.E-3
ZDT2					
MOEA/D	4.497.E-3	2.091.E-2	6.351.E-3	9.802.E-3	5.127.E-3
MOPSO	6.503.E-3	1.410.E-2	7.913.E-3	8.515.E-3	2.171.E-3
MOGWO	6.344.E-3	1.109.E-2	8.046.E-3	8.255.E-3	9.206.E-4
GPAWOA	5.701.E-3	1.042.E-2	7.802.E-3	**7.911.E-3**	7.015.E-4
ZDT3					
MOEA/D	4.034.E-3	3.019.E-2	1.260.E-3	1.381.E-2	8.875.E-3
MOPSO	4.833.E-3	1.230.E-2	6.395.E-3	**6.613.E-3**	1.402.E-3
MOGWO	9.014.E-3	1.502.E-2	1.107.E-2	1.138.E-2	1.614.E-3
GPAWOA	5.377.E-3	1.608.E-2	1.077.E-2	9.995.E-3	3.366.E-3
ZDT4					
MOEA/D	2.405.E-3	3.748.E-2	1.029.E-2	9.234.E-3	9.602.E-3
MOPSO	4.202.E-3	1.005.E-2	6.352.E-3	**6.445.E-3**	6.321.E-3
MOGWO	6.524.E-3	9.012.E-3	8.445.E-3	8.211.E-3	8.131.E-4
GPAWOA	7.533.E-3	1.039.E-2	8.397.E-3	9.055.E-3	7.867.E-3
ZDT6					
MOEA/D	5.914.E-3	9.791.E-3	7.614.E-3	8.112.E-3	1.426.E-3
MOPSO	5.813.E-3	1.327.E-2	7.144.E-3	8.235.E-3	2.805.E-3
MOGWO	6.401.E-3	2.761.E-2	8.160.E-3	1.089.E-2	6.375.E-3
GPAWOA	3.227.E-3	7.146.E-3	7.518.E-3	**5.288.E-3**	2.605.E-3

show the ability of our approach to approximate the true Pareto front and cover different disconnected regions.

The behavior of algorithms on ZDT6 test function indicates that GPAWOA is preferred rather than MOEA/D and is slightly worse compared to MOPSO and MOGWO. However, since the best result (minimum of IGD) on ZDT6 is obtained by MOGWO, the best IGD obtained by GPAWOA is getting closer to the best IGD of MOGWO. Therefore, these results indicate that GPAWOA has a better convergence ability with respect to the other algorithms.

This conclusion can be confirmed by seeing figures 5.2-5.6. From these figures, we can see clearly that our approach attains a good convergence on all the five test functions.

The statistical results of the Sp metric presented in Table 5.4 show that GPAWOA wins on three among the five test functions, and ranks second and third on ZDT3 and ZDT4 respectively, but with highly competitive results with respect to MOEA/D and MOGWO. Additionally, the visual comparisons in figures 5.1-5.2 show that the nondominated solutions produced by the proposed GPAWOA algorithm on the test functions ZDT1 and ZDT2, respectively, are well-distributed along the obtained Pareto front.

Also, we observed that the proposed GPAWOA algorithm can easily obtains a well-extended Pareto front in ZDT3 and ZDT4 (Figures 5.3 and 5.4). By observing figure 5.5 which illustrates the Pareto fronts of ZDT6 obtained by GPAWOA, MOEA/D and MOPSO, we can seen that GPAWOA produce well-distributed nondominated solutions within the search space.

Overall, we can say that the performance of GPAWOA is very competitive even superior on the most bi-objectives test functions when compared with the selected state-of-the-art algorithms. In the next section, we evaluate the performance of the proposed algorithm on the tri-objectives test functions, these functions are considered more difficult than the first test functions.

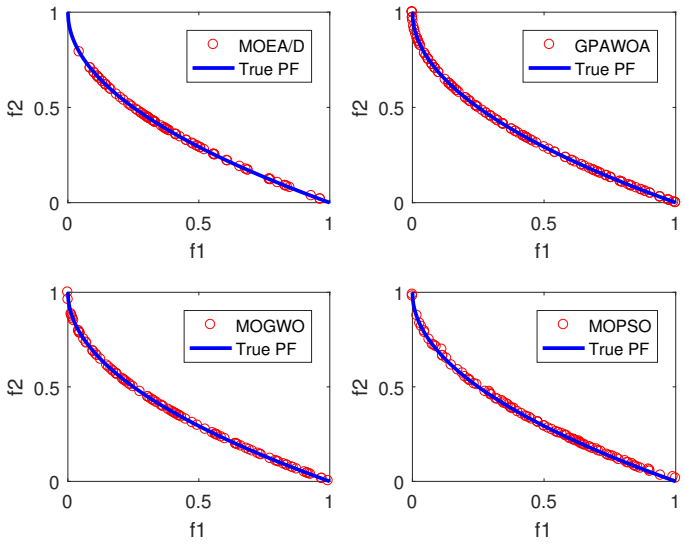

Figure 5.2: Best Pareto fronts produced by GPAWOA, MOEA/D, MOGWO and MOPSO of ZDT1 test function

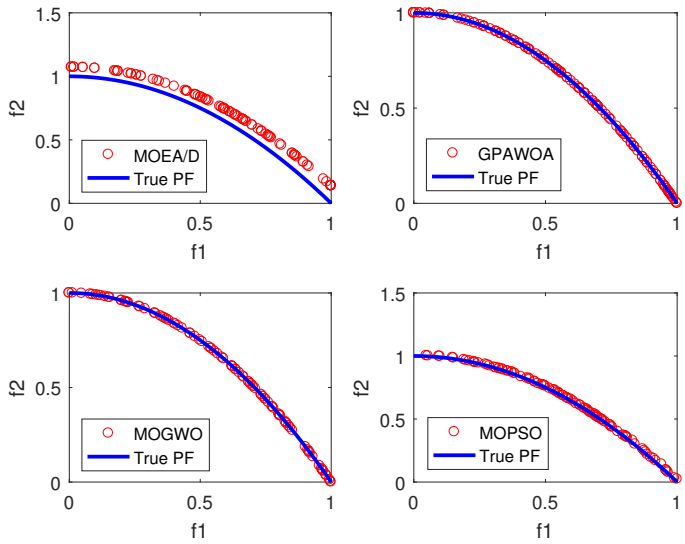

Figure 5.3: Best Pareto fronts produced by GPAWOA, MOEA/D, MOGWO and MOPSO of ZDT2 test function

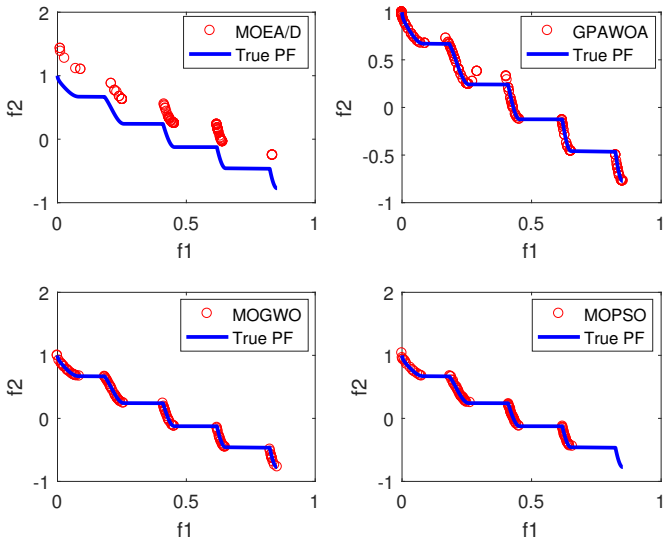

Figure 5.4: Best Pareto fronts produced by GPAWOA, MOEA/D, MOGWO and MOPSO of ZDT3 test function

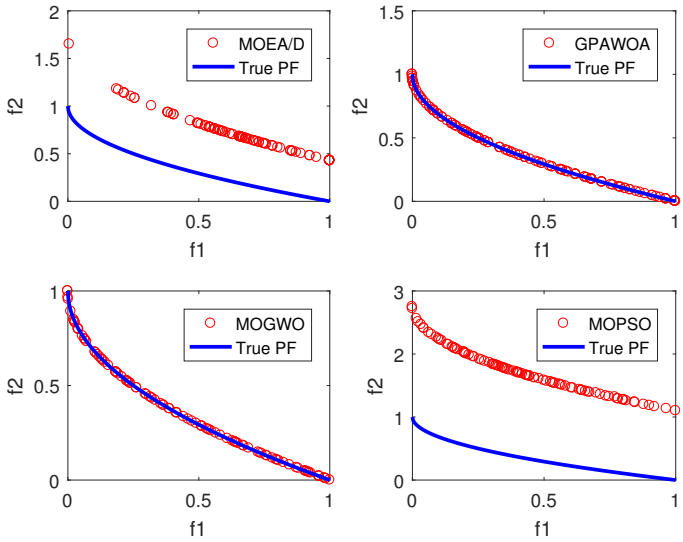

Figure 5.5: Best Pareto fronts produced by GPAWOA, MOEA/D, MOGWO and MOPSO of ZDT4 test function

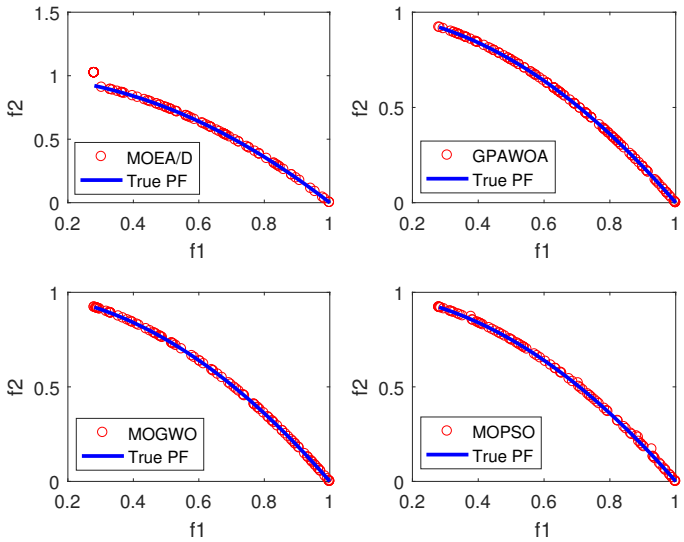

Figure 5.6: Best Pareto fronts produced by GPAWOA, MOEA/D, MOGWO and MOPSO of ZDT6 test function

5.3.2 Results on DTLZ test functions

The results of IGD and Spacing(Sp) are reported in Tables 5.5 and 5.6 respectively, whereas Figures 5.7-5.13 illustrate the graphical results produced by each algorithm on DTLZ test functions.

By observing the IGD values obtained by each algorithm in Table 5.5, the

Table 5.5: Statistical results for IGD on DTLZ test functions

Algorithm	best	worst	median	average	Std.
DTLZ1					
MOEA/D	2.045.E-2	3.112.E-1	5.167.E-1	6.140.E-1	5.711.E-2
MOPSO	5.261.E-3	7.580.E-2	1.964.E-2	**2.472.E-2**	1.912.E-2
MOGWO	4.518.E-2	1.432.E-1	9.962.E-2	9.821.E-2	2.643.E-2
GPAWOA	1.185.E-1	4.277.E-1	3.703.E-1	3.520.E-1	8.868.E-2
DTLZ2					
MOEA/D	1.352.E-3	2.014.E-3	1.698.E-3	**1.623.E-3**	2.018.E-4
MOPSO	3.541.E-3	4.711.E-3	4.197.E-3	4.035.E-3	4.678.E-4
MOGWO	1.934.E-3	7.566.E-3	5.278.E-3	5.023.E-3	6.002.E-4
GPAWOA	1.310.E-3	4.322.E-3	2.549.E-3	2.733.E-3	1.145.E-3
DTLZ3					
MOEA/D	8.023.E-1	7.114.E+0	5.166.E+0	6.615.E+0	3.409.E-1
MOPSO	5.155.E-1	4.163.E+0	1.625.E+0	1.423.E+0	5.098.E-1
MOGWO	1.737.E+0	3.518.E+0	3.387.E+0	2.454.E+0	4.580.E-1
GPAWOA	3.102.E-1	2.483.E+0	1.110.E+0	**1.379.E+0**	6.035.E-1
DTLZ4					
MOEA/D	1.455.E-3	8.725.E-3	1.633.E-3	**2.341.E-3**	2.228.E-3
MOPSO	1.509.E-3	6.277.E-3	5.044.E-3	4.418.E-3	1.795.E-3
MOGWO	2.444.E-3	3.388.E-3	2.563.E-3	2.625.E-3	1.109.E-3
GPAWOA	2.078.E-3	5.118.E-3	2.741.E-3	3.156.E-3	9.845.E-4
DTLZ5					
MOEA/D	2.854.E-4	5.995.E-4	3.477.E-4	4.236.E-4	1.137.E-4
MOPSO	2.560.E-4	9.011.E-4	4.225.E-4	5.487.E-4	2.113.E-4
MOGWO	5.416.E-4	7.166.E-4	6.020.E-4	6.309.E-4	1.166.E-4
GPAWOA	1.035.E-4	5.012.E-4	4.136.E-4	**3.995.E-4**	1.457.E-4
DTLZ6					
MOEA/D	3.925.E-4	9.005.E-4	5.097.E-4	4.921.E-4	1.187.E-4
MOPSO	2.915.E-4	5.231.E-2	3.019.E-3	1.259.E-2	1.970.E-2
MOGWO	2.447.E-4	2.514.E-3	1.709.E-3	1.522.E-3	4.097.E-4
GPAWOA	2.038.E-4	5.007.E-4	2.991.E-4	**3.257.E-4**	1.077.E-4
DTLZ7					
MOEA/D	9.322.E-3	4.271.E-2	1.366.E-2	2.227.E-2	1.362.E-2
MOPSO	1.108.E-2	1.157.E-2	1.138.E-2	1.132.E-2	1.075.E-4
MOGWO	8.044.E-4	7.814.E-3	9.106.E-4	1.611.E-3	2.235.E-3
GPAWOA	1.577.E-3	2.861.E-3	1.430.E-3	**1.419.E-3**	5.087.E-4

Table 5.6: Statistical results for Sp on DTLZ test functions

Algorithm	best	worst	median	average	Std.
DTLZ1					
MOEA/D	4.321.E-2	5.126.E-2	4.767.E-2	4.530.E-2	3.437.E-2
MOPSO	2.582.E-2	2.042.E-1	4.639.E-2	7.444.E-2	5.650.E-2
MOGWO	5.036.E-2	8.838.E-2	7.389.E-2	7.264.E-2	1.226.E-2
GPAWOA	9.030.E-3	8.160.E-2	2.117.E-2	**3.303.E-2**	2.628.E-2
DTLZ2					
MOEA/D	3.542.E-2	5.081.E-2	4.189.E-2	4.342.E-2	5.221.E-3
MOPSO	2.271.E-2	4.071.E-2	3.411.E-2	**3.425.E-2**	6.599.E-3
MOGWO	3.306.E-2	4.540.E-2	3.621.E-2	3.782.E-2	4.822.E-3
GPAWOA	2.551.E-2	6.885.E-2	4.281.E-2	4.307.E-2	5.891.E-3
DTLZ3					
MOEA/D	1.032.E-2	2.433.E-1	3.189.E-2	1.132.E-1	5.302.E-2
MOPSO	2.876.E-2	6.536.E-2	4.142.E-2	4.510.E-2	1.352.E-2
MOGWO	3.134.E-2	9.140.E-2	7.455.E-2	6.759.E-2	1.871.E-2
GPAWOA	1.945.E-4	2.736.E-2	3.646.E-2	**3.117.E-2**	8.558.E-2
DTLZ4					
MOEA/D	3.691.E-2	5.170.E-2	4.321.E-2	4.435.E-2	4.209.E-3
MOPSO	7.955.E-2	1.285.E-1	4.250.E-2	9.402.E-2	3.111.E-2
MOGWO	4.605.E-2	6.193.E-2	5.120.E-2	5.271.E-2	5.733.E-3
GPAWOA	2.143.E-2	6.301.E-2	3.715.E-2	**3.740.E-2**	1.975.E-2
DTLZ5					
MOEA/D	2.702.E-2	7.453.E-2	4.128.E-2	4.243.E-2	4.822.E-3
MOPSO	6.312.E-2	7.699.E-2	4.174.E-2	7.009.E-2	2.665.E-3
MOGWO	1.451.E-2	2.163.E-2	1.799.E-2	1.754.E-2	2.956.E-3
GPAWOA	4.402.E-3	1.711.E-2	5.143.E-3	**9.067.E-3**	4.035.E-3
DTLZ6					
MOEA/D	9.798.E-3	1.311.E-2	1.212.E-2	1.170.E-2	1.250.E-3
MOPSO	1.022.E-2	4.541.E-2	1.790.E-2	2.045.E-2	1.128.E-2
MOGWO	9.223.E-3	1.270.E-2	1.075.E-2	1.079.E-2	1.445.E-3
GPAWOA	2.667.E-3	5.115.E-3	4.885.E-3	**4.052.E-3**	1.054.E-3
DTLZ7					
MOEA/D	1.781.E-3	3.171.E-3	3.005.E-3	**3.027.E-3**	9.633.E-3
MOPSO	2.887.E-3	3.809.E-2	3.404.E-2	3.365.E-2	3.387.E-3
MOGWO	3.993.E-3	3.289.E-2	2.641.E-2	2.340.E-2	8.477.E-3
GPAWOA	5.025.E-3	3.499.E-2	1.831.E-2	4.857.E-2	1.130.E-2

proposed algorithm is considered superior to the other algorithms in terms of convergence by obtaining four best average values of IGD, on DTLZ3 and DTLZ5-7 test functions, while MOEA/D and MOPSO achieve two and one, respectively, best average of IGD on the other test functions.

On DTLZ1 test function, GPAWOA performed worst comparing to MOPSO

and MOGWO (MOPSO was the best performer) and is better than MOEA/D. The results of IGD on DTLZ2 indicate that the GPAWOA produces better approximation towards the true Pareto front comparing to MOPSO and MOGWO. As compared with MOEA/D, our algorithm ranks second with highly competitive results. On DTLZ4, GPAWOA is preferred than MOPSO and it places slightly blow MOEA/D and MOGWO with a lower standard deviation.

Regarding Spacing(Sp) results highlighted in Table 5.6, it can be seen that our approach wins on five test functions (DTLZ1 and DTLZ3-6). On DTLZ2 and DTLZ7, MOPSO and MOEA/D gets, respectively, the best average Sp. However, it can be noticed that the proposed algorithm produces a results reasonably good on these two functions. Moreover, we can clearly observe in figure 5.13 that GPAWOA is able to attain a raisonable distribution of nondominated solutions on DTLZ7. Figure 5.8 shows the fronts produced by the algorithms on DTLZ2. It can be seen that GPAWOA is better than MOPSO in term of diversity. On DTLZ5 and DTLZ6, all the algorithms have a similar behavior except MOGWO on DTLZ5 which failed the Pareto optimal front (see figures 5.11 and 5.12).

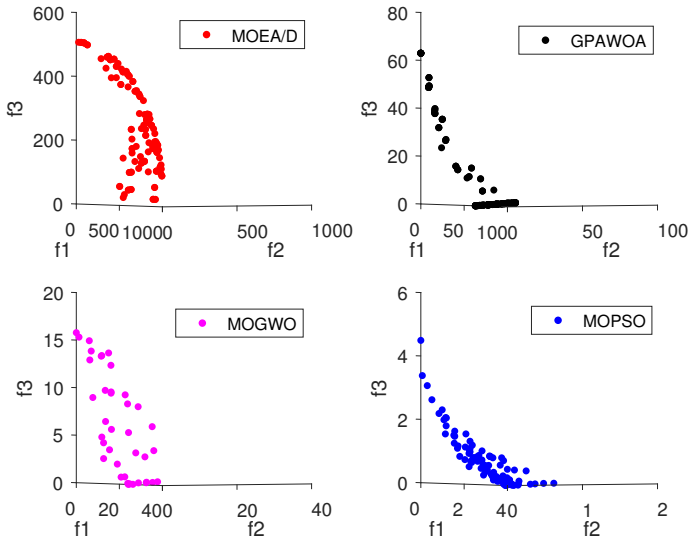

Figure 5.7: Best Pareto fronts produced by GPAWOA, MOEA/D, MOGWO and MOPSO of DTLZ1 test function

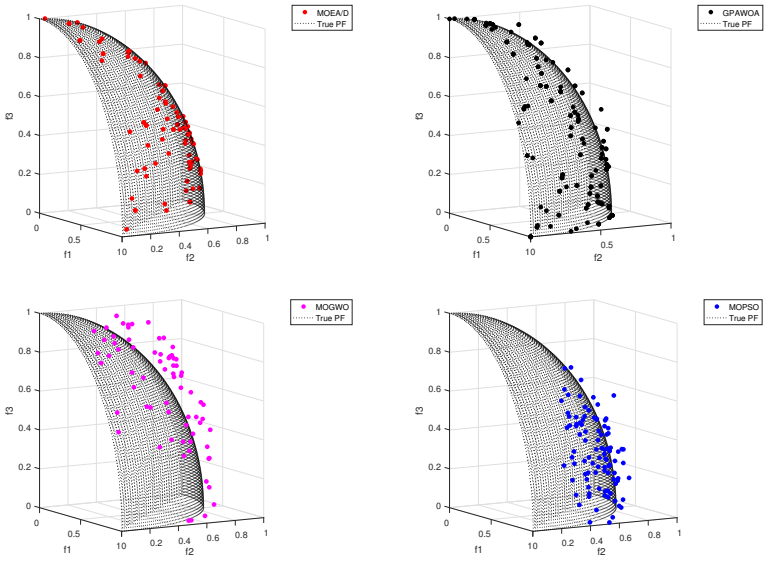

Figure 5.8: Pareto front of DTLZ2: a comparison of the front found by MOEA/D, MOGWO, MOPSO, GPAWOA and the true Pareto front

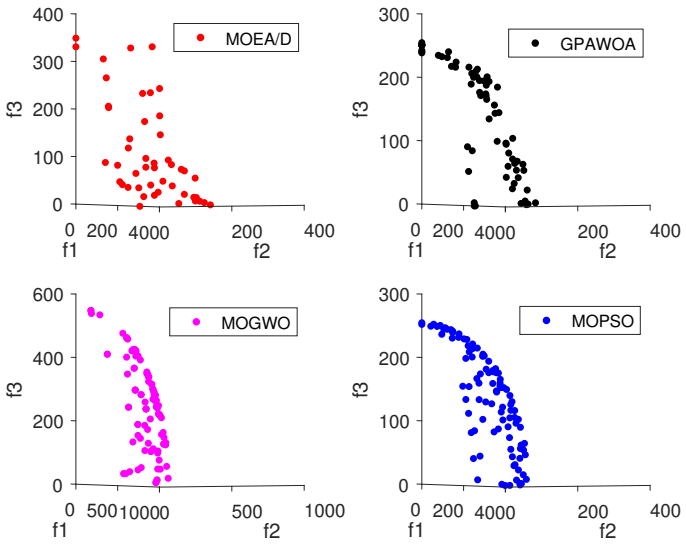

Figure 5.9: Best Pareto fronts produced by GPAWOA, MOEA/D, MOGWO and MOPSO of DTLZ3 test function

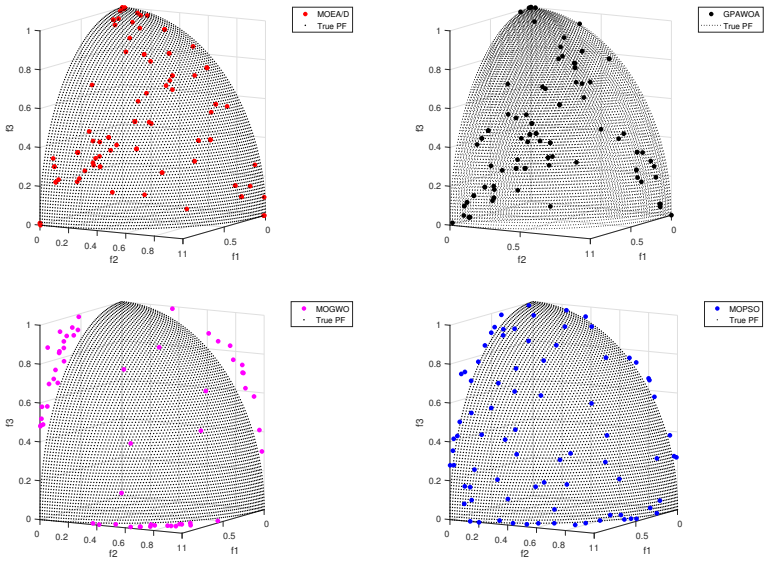

Figure 5.10: Pareto front of DTLZ4: a comparison of the front found by MOEA/D, MOGWO, MOPSO, GPAWOA and the true Pareto front

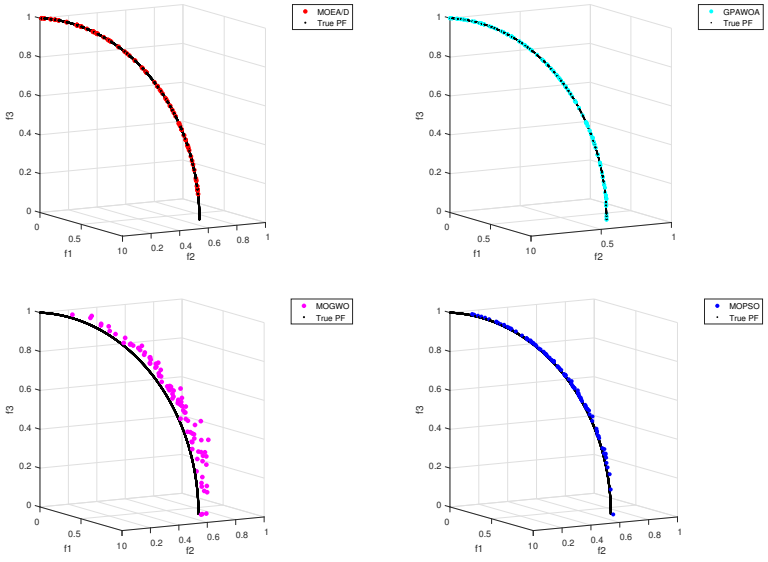

Figure 5.11: Pareto front of DTLZ5: a comparison of the front found by MOEA/D, MOGWO, MOPSO, GPAWOA and the true Pareto front

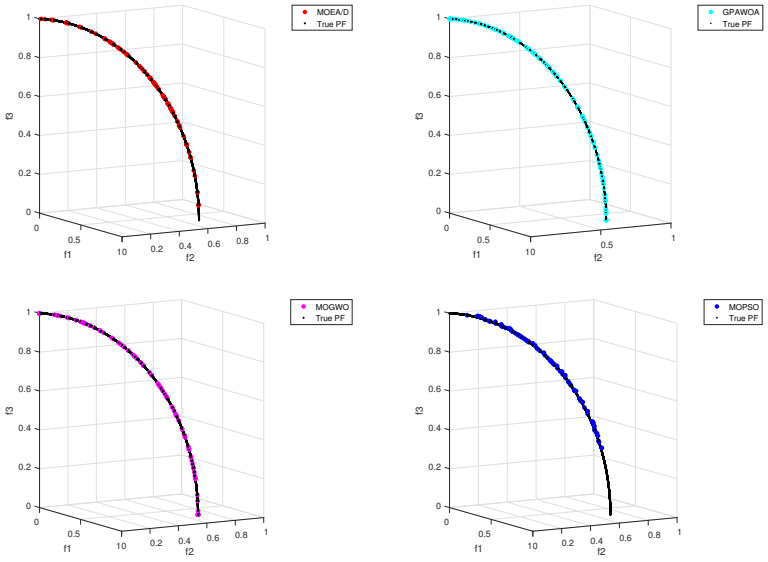

Figure 5.12: Pareto front of DTLZ6: a comparison of the front found by MOEA/D, MOGWO, MOPSO, GPAWOA and the true Pareto front

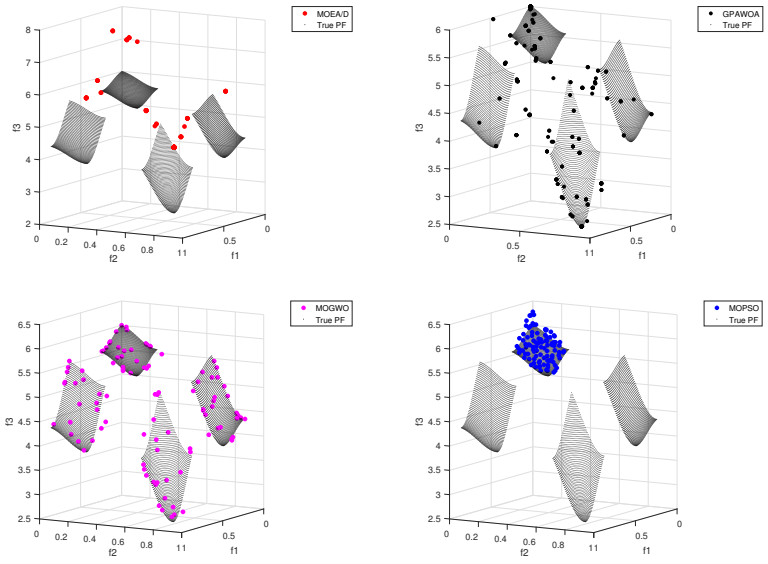

Figure 5.13: Pareto front of DTLZ7: a comparison of the front found by MOEA/D, MOGWO, MOPSO, GPAWOA and the true Pareto front

5.3.3 Statistical comparison of Wilcoxon test

To confirm the above conclusions, we applied the statistical comparison called "Wilcoxon test" [Derrac 2011] to judge, according to p-value and a given significance level α, the significant differences between the performances of two compared algorithms by confirming the null hypothesis H_0 (no significant difference between the results of a pair algorithms) if $p\_value > \alpha$, or the alternative hypothesis H_1 (there's a significant difference between the results of a pair algorithms) if $p\_value \leq \alpha$

We performed the wilcoxon test between GPAWOA and each peer algorithm (MOEA/D, MOGWO and MOPSO) on the results of the IGD and Sp metrics obtained for 30 independently runs for each test function, with a significance level which equals 5%. We present the results of GPAWOA as μ_1 and the results of each peer algorithm as μ_2 and we define the following hypotheses for IGD and Sp metrics:

- $H_0 : \mu_1 = \mu_2$, which indicates that the IGD and Sp of GPAWOA are similar to the IGD and Sp of the other algorithms.

- $H_1 : \mu_1 < \mu_2$, which indicates that the IGD and Sp of GPAWOA are smaller than to the IGD and Sp of the other algorithms.

The results of p-value obtained by Wilcoxon test for the IGD metric, as shown in Table 5.7, indicate that GPAWOA is equivalent to MOGWO and significantly outperforms MOEA/D and MOPSO for the most test functions. From Table 5.8, which shows the results of p-value on the basis of Sp metric, it can be seen that GPAWOA is comparable to MOGWO and it is performing better than MOEA/D and MOPSO for large test functions.

Since IGD and Spacing(Sp) are good metrics to measure the convergence and diversity, respectively, of an algorithm, so from the numerical results of these metrics and the graphical results and the significant comparison performed using the wilcoxon test, we can conclude that our approach is highly competitive when compared with the selected state-of-the-art approaches.

As a summary, the performance of the GPAWOA algorithm and its ability to get closer to the true Pareto front can be explained, well-obviously, by the high performance of the original WOA on the one hand, and on the other hand, by the effectiveness of the archiving and leader selection strategies adopted. Moreover,

Table 5.7: The p values of Wilcoxon test for IGD metric

GPAWOA vs			
	MOEA/D	**MOGWO**	**MOPSO**
ZDT1	**5.412E-6**	**2.621E-2**	**5.412E-6**
ZDT2	**5.412E-6**	0.3696	**5.412E-6**
ZDT3	**5.412E-6**	0.9919	**5.412E-6**
ZDT4	**5.412E-6**	**5.250E-4**	**5.412E-6**
ZDT6	**1.044E-3**	0.6303	0.6847
DTLZ1	0.9698	0.9862	1
DTLZ2	1	**3.150E-2**	**3.934E-2**
DTLZ3	**2.621E-2**	**7.523E-4**	**2.621E-2**
DTLZ4	0.2406	0.9783	0.6847
DTLZ5	**6.917E-3**	**5.412E-6**	**4.465E-3**
DTLZ6	**1.044E-3**	**7.523E-4**	**1.623E-4**
DTLZ7	**5.412E-6**	0.5441	**5.412E-6**

Table 5.8: The p values of Wilcoxon test for spacing(Sp) metric

GPAWOA vs			
	MOEA/D	**MOGWO**	**MOPSO**
ZDT1	**3.150E-2**	0.4347	0.2893
ZDT2	0.3696	0.3421	**1.773E-2**
ZDT3	0.1736	**4.460E-2**	0.5225
ZDT4	**2.162E-2**	0.8762	0.8034
ZDT6	**1.044E-3**	**2.435E-4**	**2.435E-4**
DTLZ1	**1.161E-2**	**1.044E-3**	**1.161E-2**
DTLZ2	0.8089	0.6782	0.9496
DTLZ3	**1.439E-3**	**1.623E-4**	**7.523E-4**
DTLZ4	0.6578	0.2179	**5.412E-6**
DTLZ5	**5.412E-6**	**3.788E-5**	**5.412E-6**
DTLZ6	**5.412E-6**	**5.412E-6**	**5.412E-6**
DTLZ7	0.9895	0.3979	0.2893

its ability to produce well-distributed solutions along the obtained front reflects the impact of the crowding distance mechanism employed to improve the diversity of the final Pareto front.

5.3.4 Application of GPAWOA on engineering design problems

As mentioned above, the proposed algorithm is applied to 4 real engineering design problems [Sadollah 2015], [Askarzadeh 2016] whose two problems without constraints (Four-bar truss design problem and gear train problem), and two problems with constraints (Disk brake design problem and welded beam design problem):

- **Four bar truss problem**
  This problem attempt to minimize the volume and displacement of a 4-bar truss. It is mathematically formulated as follows:

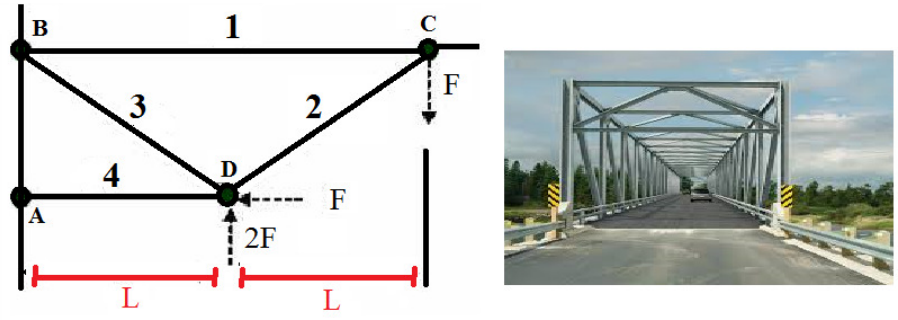

Figure 5.14: Four bar truss problem

$$minimize \begin{cases} f_1(x) = L(2x_1 + \sqrt{2x_2} + \sqrt{x_3} + x_4) \\ f_2(x) = \frac{FL}{E}(\frac{2}{x_2} + \frac{2\sqrt{2}}{x_2} - \frac{2\sqrt{2}}{x_3} + \frac{2}{x_4}) \end{cases} \tag{5.5}$$

where
$F = 10, E = 2e^5, L = 200$
$1 \leq x_1, x_4 \leq 3, \sqrt{2} \leq x_2, x_3 \leq 3$

- **Gear train problem**
  The aim of this problem is to minimize the maximum size of any one of the gear and minimize the error between the obtained gear ratio and a required gear ratio of $\frac{1}{1.931}$. This problem can be stated as::

$$minimize \begin{cases} f_1(x) = (\frac{1}{6.931} - \frac{x_1 x_2}{x_3 x_4})^2 \\ f_2(x) = max(x_1, x_2, x_3, x_4) \end{cases} \tag{5.6}$$

where
$12 \leq x_1, x_2, x_3, x_4 \leq 60$

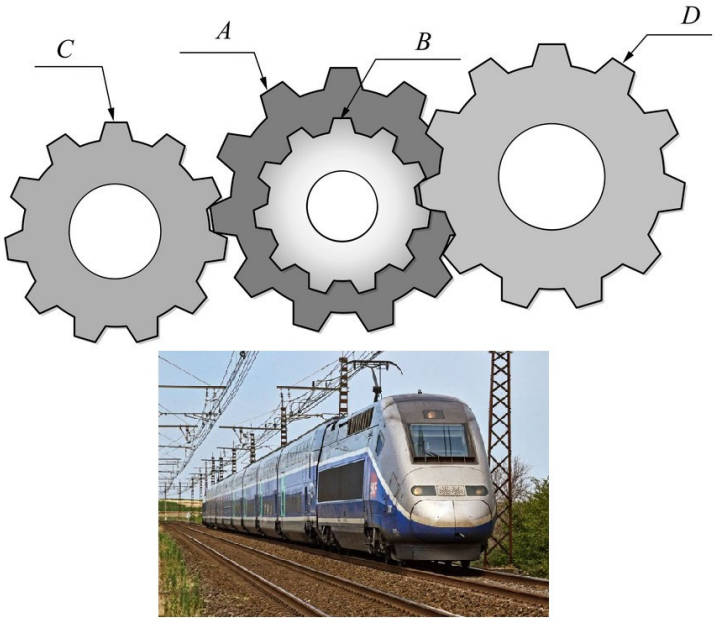

Figure 5.15: Gear train problem

- **Disk brake design problem**
  The disk brake design problem has two objectives to be minimized which are the mass of the brake and the stopping time.

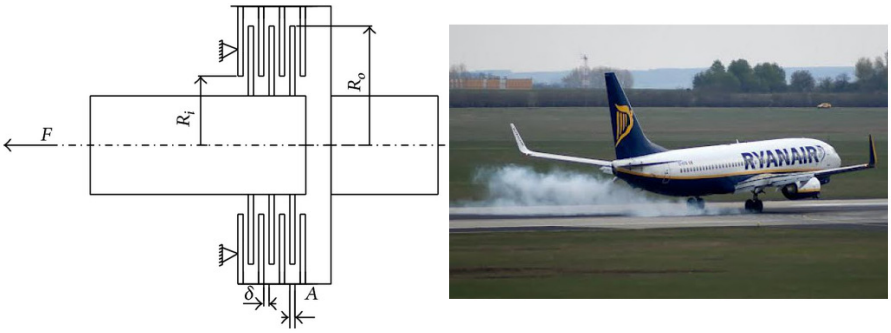

Figure 5.16: Disk brake design problem

$$minimize \begin{cases} f_1(x) = 4.9e - 5(x_2^2 - x_1^2)(x_4 - 1) \\ f_2(x) = (9.82e^6)\frac{x_2^2 - x_1^2}{x_3 x_4 (x_2^3 - x_1^3)} \end{cases} \tag{5.7}$$

Subject to :

$g_1 = 20 + x_1 - x_2$

$g_2 = 2.5(x_4 + 1) - 30$

$g_3 = \frac{x_3}{3.14(x_2^2-x_1^2)^2} - 0.4$
$g_4 = 2.22e - 3x_3\frac{x_2^3-x_1^3}{(x_2^2-x_1^2)^2} - 1$
$g_5 = 900 - \frac{2.66e-2x_3x_4(x_3^3-x_1^3)}{x_2^2-x_1^2}$
where
$\forall g_i \leq 0$
$55 \leq x_1 \leq 80, 75 \leq x_2 \leq 110, 1000 \leq x_3 \leq 3000, 2 \leq x_4 \leq 20$

- **Welded beam desing problem**
  In this problem, the fabrication cost and deflection of the beam of a welded beam should be minimized.

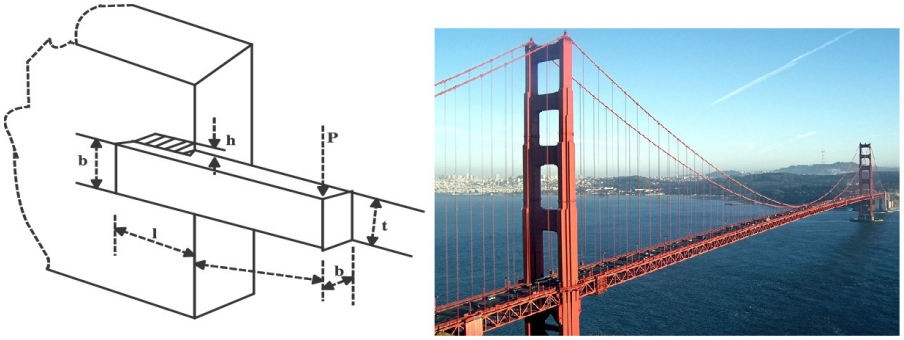

Figure 5.17: Welded beam design problem

$$minimize \begin{cases} f_1(x) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14 + x_2) \\ f_2(x) = del \end{cases} \quad (5.8)$$

Subject to :
$g_1 = -(tau - taumax)$
$g_2 = -(sig - sigmax)$
$g_3 = -(x_1 - x_4)$
$g_4 = -(P - pc))$
where,
$pc = (\frac{4.013*E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2})(1 - ((\frac{x_3}{2L})\sqrt{\frac{E}{4G}}))$
$del = \frac{4PL^3}{Ex_3^3x_4}$
$sig = \frac{6PL}{x_4x_3^2}$
$J = 2 * (\sqrt{2}x_1x_2((\frac{x_2^2}{12}) + (\frac{x_1+x_3}{2})^2))$
$R = \sqrt{(\frac{x_2^2}{4}) + (\frac{x_1+x_3}{2})^2}$

$M1 = P * (L + \frac{x_2}{2})$
$tau2 = \frac{M1R}{J}$
$tau1 = \frac{P}{\sqrt{2}x_1x_2}$
$tau = \sqrt{tau1^2 + 2.tau1.tau2.\frac{x_2}{2R} + tau2^2}$
$\forall g_i \geq 0$
$P = 6000, L = 14, E = 30e^6, G = 12e^6, taumax = 13600, sigmax = 30000$
$0.125 \leq x_1, x_4 \leq 5, 0.1 \leq x_2, x_3 \leq 10$

To handle the constraints in the algorithm in order to keep the search within feasible regions, we use the static penalty method [Rao 2019] by adding a penalized value into the objective function as follows:

$$f_m(x) = f_m(x) + \sum_{i=1}^{p} P_i.max(g_i(x), 0) + \sum_{i=1}^{K} P_i.max(|h_i(x)| - \delta, 0) \tag{5.9}$$

where
$f_m(x)$,$m = 1, 2, ...M$ are the objective function to be optimized.
$g_i(x) \leq 0$, $i = 1, 2, ...P$ are inquality constraints.
$h_i(x) = 0$, $i = 1, 2, ...K$ are equality constraints.
P_i and δ are, respectively, the penalty facteur and the tolerance on the equality constraints.

The results (covergae metric) are reported in Table 5.9. In addition, figures 5.18-5.21 depicts the graphical comparisons among the proposed approach and the others reported algorithms.

From results in Table 5.9, it can be evidently found that for the metric C, C(MOEA/D, GPAWOA); C(MOGWO, GPAWOA); C(MOPSO, GPAWOA) are almost all near of 0 value for all considered engineering design problems except C(MOPSO, GPAWOA) for welded beam design problem. Conversely, in the case where the coverage metric computes the dominance proportion of nondominated solutions found by the proposed algorithm compared with the set of nondominated solution found by the other algorithms, C(GPAWOA, MOEA/D); C(GPAWOA, MOGWO); C(GPAWOA, MOPSO) are almost all near 1 except C(GPAWOA, MOPSO) for welded beam problem. This means that most of the nondominated solutions obtained by MOEA/D, MOGWO and MOPSO are dominated by solutions generated by the proposed algorithm. Moreover, from the Pareto fronts shown in figures 5.18-5.21, we can seen that the diversity of GPAWOA is better compared to its competitors for all the selected engineering design problems.

Table 5.9: Results of set coverage metric of the four engineering design problems

Algorithms	GPAWOA	MOEA/D	MOGWO	MOPSO
4-bar truss				
GPAWOA	/	**1**	**1**	**0.991**
MOEA/D	**0**	/	0	0
MOGWO	**0**	1	/	0.028
MOPSO	**0.011**	1	0.942	/
Gear train				
GPAWOA	/	**1**	**1**	**0.926**
MOEA/D	**0**	/	0.215	0.076
MOGWO	**0**	0.769	/	0.510
MOPSO	**0.010**	0.986	0.433	/
Disk brake				
GPAWOA	/	**0.997**	**965**	**0.982**
MOEA/D	**0.007**	/	0.019	0.232
MOGWO	**0.021**	0.810	/	0.912
MOPSO	**0.013**	0.734	0.009	/
Welded beam				
GPAWOA	/	**1**	**1**	**0.746**
MOEA/D	**0**	/	0.217	0.556
MOGWO	**0**	0.705	/	0.681
MOPSO	**0.326**	0.433	0.176	/

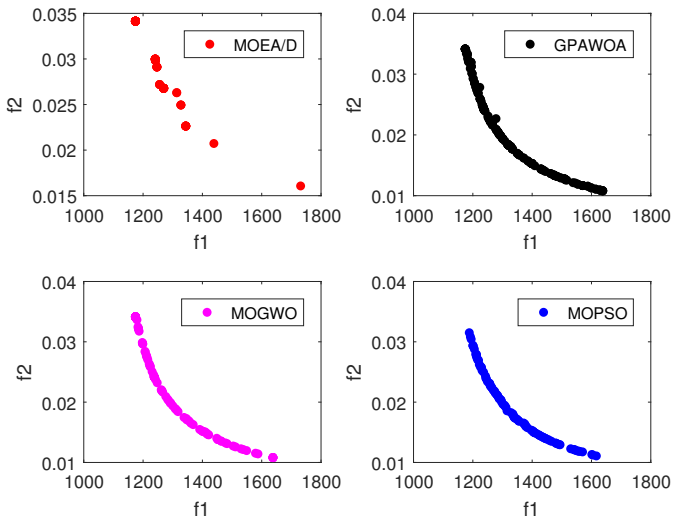

Figure 5.18: Comparison of nondominated solutions obtained by MOEA/D, GPAWOA, MOPSO and MOGWO for the 4-bar truss

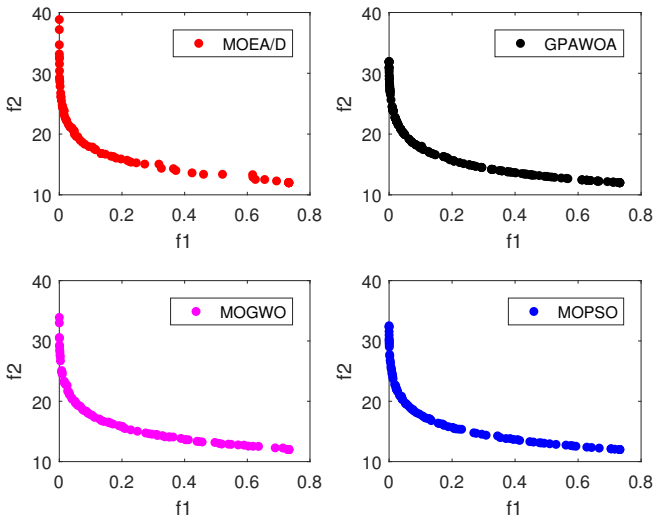

Figure 5.19: Comparison of nondominated solutions obtained by MOEA/D, GPAWOA, MOPSO and MOGWO for the Gear train

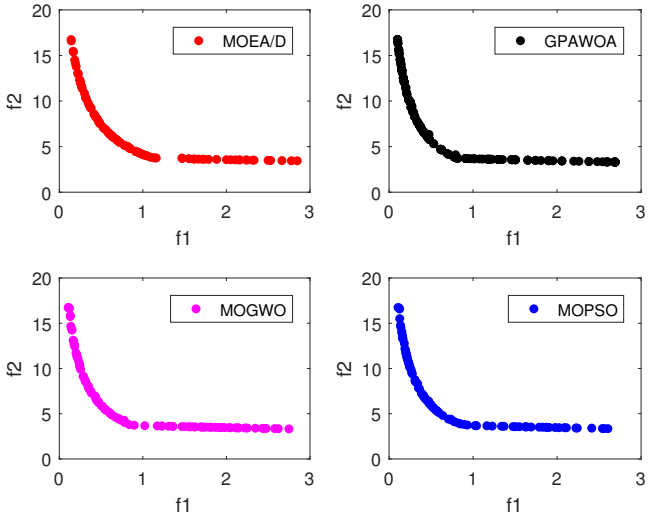

Figure 5.20: Comparison of nondominated solutions obtained by MOEA/D, GPAWOA, MOPSO and MOGWO for the Disk brake

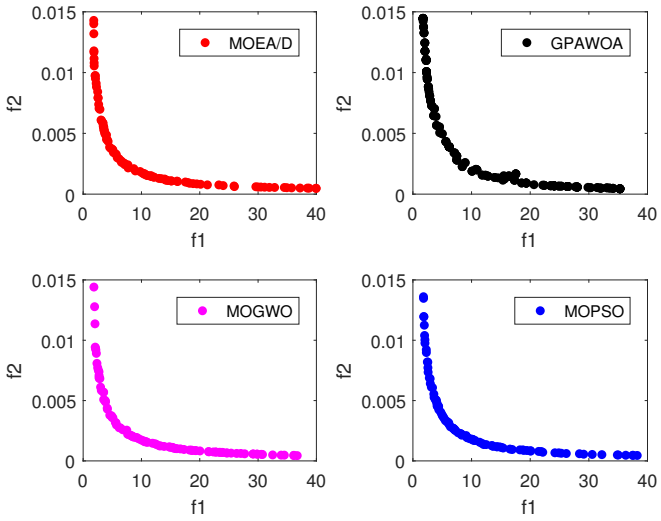

Figure 5.21: Comparison of nondominated solutions obtained by MOEA/D, GPAWOA, MOPSO and MOGWO for the Welded beam

5.4 Chapter summary

In this chapter, we have developed a new multiobjective optimizer named A Guided Population Archive Whale Optimization Algorithm (GPAWOA) by extending the recently Whale Optimization Algorithm. The proposed GPAWOA incorporates an external archive as a guide of the main population towards the optimal front; an effective leader selection strategy is integrated to improve both convergence and diversity. Finally, the mechanism of crowding distance is employed as a density estimator in the update of the archive and leader selection strategy to maintain a good coverage of the objective space.

A comparative study of the proposed approach with three well-regarded algorithms has been performed on twelve challenging benchmark test functions widely used in this field. The numerical and graphical results showed clearly that the proposed GPAWOA is highly competitive even outperforms the selected algorithms on most test functions. The proposed algorithm was also applied to four multiobjective engineering design problems to demonstrate its applicability in practical problems and the results obtained by GPAWOA were very satisfactory. Hence, the proposed algorithm can be effectively applied to real-world optimization problems to arrive at desired solutions. According to this comprehensive study, we can conclude that the proposed GPAWOA is a viable alternative for solving multiobjective optimization problems.

CHAPTER 6

Hybride filter-wrapper GPAWOA for feature selection

Contents

6.1 Introduction

Feature selection aims at finding the minimum number of features that result in high classification accuracy. It plays a crucial role in machine learning as it is the best weapon against the so-called "curse of dimensionality" problem. The feature space usually contains a large number of features which can be relevant, irrevelant or redundant [Xue 2015]. Irrelevant and redundant features are not useful as they affect the performance of the learning algorithm. Therefore, feature selection is invoked to select the optimal subset of relevant features in order to: decrease the dimensionality of the feature space, simplify the learned classifier, improve the classification performance and reduce the running time [Dash 1997], [Unler 2010].

In this chapter, we investigate our GPAWOA algorithm to propose a novel approach for handling feature selection problem. The proposed algorithm takes on one hand; the searching ability of GPAWOA to get promising regions in the feature space. On the other hand, it merges the merits of filter and wrapper models into a single system in order to achieve better performance. Two objective

functions are taken into account during the optimization process. The first objective estimate the relevance and redundancy between features by using mutual information (MI) as filter fitness function to identify the non-dominated subsets that result in minimum redundancy and maximum relevance of features to the target class, while the second objective estimate the classification accuracy by using a learning classifier as wrapper fitness function.

6.2 The propsed algorithm

6.2.1 Entropy and mutual information

Information theory, including entropy and mutual information, offers a means for measuring the information of random variables [Shannon 1948]. The entropy is a measure of uncertainty of random variable X. It is defined as:

$$H(X) = -\sum_{x \in X} p(x) \log_2 p(x) \tag{6.1}$$

where, x a discrete value and $p(x) = Pr(X = x)$ stands for the probability density function of X. For two discrete random variables X and Y with their probability density function $p(x, y)$, the joint entropy $H(X, Y)$ is defined as:

$$H(X, Y) = -\sum_{x \in X, y \in Y} p(x, y) \log_2 p(x, y) \tag{6.2}$$

In the case when a certain variable Y is identified and others are unidentified, the remaining uncertainty is measured by the conditional entropy $H(X|Y)$:

$$H(X|Y) = -\sum_{x \in X, y \in Y} p(x, y) \log_2 p(x|y) \tag{6.3}$$

The common information quantity of two random variables X and Y is defined as the mutual information between them:

$$I(X; Y) = -\sum_{x \in X, y \in Y} p(x, y) \log_2 \frac{p(x, y)}{p(x)p(y)} \tag{6.4}$$

If the two variables X and Y are independant, that is $p(x, y) = p(x)p(y)$, then $I(X; Y)$ will be zero. Otherwise, $I(X; Y)$ will be large.

6.2.2 GPAWOA for discrete problems

Feature selection occur in a discrete search space - i.e, the solutions are restricted to 0 or 1 values, for this reason, it is preferable to develop firstly a binary variant of the GPAWOA.

A simple way to convert the continuous version of GPAWOA to the binary version is to use a transfer function. In our case, the hyperbolic tangent function is used as follows:

$$T(x^{i+1}) = \frac{e^{-\lambda(x^{i+1})} - 1}{e^{-\lambda(x^{i+1})} + 1} \tag{6.5}$$

where, $\lambda = 1$ is examined in this work, x^{i+1} is the real-value of the current search agent, thus, the binary update is carried out as:

$$x_{bin} = \begin{cases} 1 & if\ \mu\ <\ T(x^{i+1}) \\ 0 & Otherwise \end{cases} \tag{6.6}$$

with, μ is a random number with uniform distribution in (0, 1).

Each search agent in the population represents a subset of features. For a n-dimensional feature search space, each whale is encoded by a n-bit binary string. The i^{th} whale's position is a n-dimensional vector $X_i = [x_1, x_2, ..., x_n]$ with $x_j = \{0, 1\}$. $x_j = 1$ indicates that the corresponding j^{th} feature is selected, whereas $x_j = 0$ means that the corresponding feature is not selected.

6.2.3 Outlines of FW-GPAWOA algorithm

The proposed algorithm, namely FW-GPAWOA (for Filter-Wrapper Guided Population Archive WOA), combines filter and wrapper measures into a single model. Here, it should be noted that FW-GPAWOA considers both filter and wrapper measures simultaneously during the optimization process. In other words, all iterations are attributed to evaluate and optimize the two objectives in the same time, which make the difference between our algorithm and other existing hybrid filter-wrapper feature selection algorithms which attribute a certain number of iterations to filter evaluation and attribute the rest of iterations to wrapper evaluation (two-stage), which may lead to eliminate some relevant features in the filter stage without considering to their contribution in the wrapper stage.

Mutual information (filter function) and classification accuracy (wrapper function) are used to formulated our objective function. Mutual information (MI)

is a filter metric that can be used as a measure of the relevance of any feature to the target class, and also as a measure of the redundancy among a set of features. Since that our main objective is to select the smaller subset of features that increase the classification accuracy, we propose to use a mixed filter fitness function MI-based which combines the relevance and the redundancy into a single function so that the average relevance of the selected feature subset is to be maximized, while the average redundancy among the selected features is to be minimized.

$$Fit_1 = Avg(I(x;c)) - Avg(I(x_i;x_j)) \tag{6.7}$$

where, $x, x_i, x_j \in X$, c is the target class and X is the selected feature subset. Fit_1 is a maximization function that attempt to increase the classification accuracy by maximizing $Avg(I(x;c))$ which stands for the average relevance between the selected subset and the target class, and reduce the number of features by minimizing $Avg(I(x_i;x_j))$ which indicates the average redundancy among the selected feature subset.

In Fit_1, the relevance and the redundancy are equally important. However, as known in classification problems, the classification accuracy is more important than the number of features. Therefore, Fit_1 is weighted as follows:

$$Fit_1 = \alpha.Avg(I(x;c)) - (1-\alpha).Avg(I(x_i;x_j)) \tag{6.8}$$

where, α is a constant number in the interval]0.5, 1[.

The second objective function Fit_2 maintains the wrapper evaluation which aimed at maximize the classification accuracy (Acc) given by the following equation:

$$Fit_2 = Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{6.9}$$

where, TP, TN, FP, and FN stand for true positives, true negatives, false positives, and false negatives, respectively. Fit_2 is evaluated by using a preselected classifier; K-nearest neighbors (KNN) in the current case.

Accordingly, our feature selection problem is formulated by the following multi-objective optimization problem:

$$\begin{cases} Maximize\ Fit_1 = \alpha.Avg(I(x;c)) - (1-\alpha).Avg(I(x_i;x_j)) \\ Maximize\ Fit_2 = \frac{TP+TN}{TP+TN+FP+FN} \\ Subject\ to\ x_i \in \{0,1\} \end{cases} \tag{6.10}$$

6.3 Experimental results

In order to assess the performance of the proposed algorithm, 10 benchmark datasets chosen from UCI machine learning repository [Frank 2010] are used in the experiments. Each dataset is randomly divided into two sets: 70% as the training set and 30% as the test set. The training set is used to select a feature subset(s), whereas the test set is used to evaluate the classification performance of the selected features by using a learning algorithm. In our case, the K-Nearest Neighbour (KNN) with K=5 was chosen as a learning algorithm to calculate the classification accuracy (Acc) of the selected features on the test set.

Table 6.1: Summary of utilized datasets

Datasets	No. of samples	No. of features	No. of classes
Breast cancer	699	9	2
Lymphography (Lymph)	148	18	4
Musk	476	166	2
Semeion	1593	265	2
Cervical cancer (Sobar)	72	19	2
Spect	267	22	2
Spectf	160	44	2
Sports articles	1000	59	2
Vehicle	846	18	4
Wholesale customers	440	7	2

Five algorithms including three algorithms are single-objective, BDE [Too 2019a], BPSO [Too 2019b] and BGWO [Too 2018], and two algorithms are multi-objective, MOPSO [Coello 2004] and MOGWO [Mirjalili 2016b], are used as benchmark techniques in the comparative study. For all algorithms and during the training evolutionary process, KNN (K=5) with 10-fold cross-validation is employed to assess the prediction accuracy of the selected features on the training set. To ensure the fairness of the comparison, the population size and the number of iterations in all algorithms are set to be 30 and 50, respectively.

All the algorithms have been conducted for 10 independent runs for each dataset, and they are implemented using Matlab2016b under 32-bit Windows 8.1. All experiments have been performed on the same computer: Intel Core i3, 2.39 GHz and 4 GB RAM.

Table 6.2: Parameter settings

Algorithm	Parameters
BDE	Crossover rate: $CR = 0.9$
BPSO	$c_1 = 2$, $c_2 = 2$, $V_{max} = 6$, $w_{max} = 0.9$, $w_{min} = 0.4$
BGWO	-
MOPSO	$\phi_1 = \phi_2 = 2.05$, $\phi = 4.1$, Inertia weight: $w = 0.73$ Personal coefficient: $c_1 = 1.4962$, Social coefficient: $c_2 = 1.4962$, $nGrid = 10$
MOGWO	Grid inflation : $\alpha = 0.1$, Archive member selection pressure: $\gamma = 2$, $nGrid = 10$ Leader selection pressure parameter : $\beta = 4$
GPAWOA	$b = 1$, Parcentage which determine the top part of the archive : $a = 0.1$

Table 6.3 shows the results obtained by FW-GPAWOA, BDE, BPSO and BGWO where, "Size" means the average number of the selected features and "Acc" means the average of the classification accuracies achieved by the algorithms. The values in brackets show the classification accuracies obtained by all the available features, while the best results on each dataset are marked in **boldface**.

Table 6.3: Experimental results of FW-GPAWOA, BDE, BPSO and BGWO

Datasets	Breast cancer (9, 98.35 %)				Lymph (18, 76.92 %)			
Method	FW-GPAWOA	BDE	BPSO	BGWO	FW-GPAWOA	BDE	BPSO	BGWO
Size	**2.06**	6.1	5.7	7.3	**3.3**	11.6	7.3	10.8
Acc %	**97.14**	96.70	96.85	95.05	**85.89**	80.76	60.25	82.56
Datasets	Musk (166, 84.67 %)				Semeion (265, 97.35 %)			
Method	FW-GPAWOA	BDE	BPSO	BGWO	FW-GPAWOA	BDE	BPSO	BGWO
Size	**36**	139.9	82.8	110.7	**120**	218.6	133.6	197.8
Acc %	**85.16**	79.67	84.03	84.75	97.37	**97.40**	96.78	96.58
Datasets	Sobar (19, 89.47 %)				Spect (22, 82.85 %)			
Method	FW-GPAWOA	BDE	BPSO	BGWO	FW-GPAWOA	BDE	BPSO	BGWO
Size	**3.8**	12.4	9.4	11.2	**3.4**	15.9	8.7	15.6
Acc %	**93.68**	86.31	90	88.42	**85.71**	83.57	70	82.14
Datasets	Spectf (44, 81.42 %)				Sports (59, 81.60 %)			
Method	FW-GPAWOA	BDE	BPSO	BGWO	FW-GPAWOA	BDE	BPSO	BGWO
Size	**4.82**	34.9	21.5	29.9	**9.1**	45.2	30.3	42.7
Acc %	**84.28**	76.28	73.85	74.57	**83.18**	82.26	78.35	78.65
Datasets	Vehicle (18, 66.06 %)				Wholesale (7, 86.08 %)			
Method	FW-GPAWOA	BDE	BPSO	BGWO	FW-GPAWOA	BDE	BPSO	BGWO
Size	**9.05**	13.1	9.3	12.1	**2.49**	2.9	5,1	4
Acc %	71.49	70.22	69.55	**73.48**	**92.17**	86.95	90.78	90.60

According to Table 6.3, it can be seen that the proposed approach could effectively select fewer number of features with higher classification accuracy in almost all datasets. For example, in Breast cancer dataset, FW-GPAWOA selected on average of 2.06 from 9 features, which means around 23 % of the initial feature

set, selected 2.49 from 7 features in Wholesale dataset, i.e, 36 % of the original feature set, and it was able to select approximately 11 % of the available features (on average of 4.82 from 44 features) in Spectf dataset.

Regarding the classification performance, it can be seen clearly that FW-GPAWOA could achieve higher classification accuracy than using all features in most datasets. Indeed, except on Breast cancer dataset where the classification accuracy was slightly decreased from 98.35 % to 97.14 %, FW-GPAWOA was able to increase the prediction accuracy in all remaining cases. Generally speaking, the results suggest that the proposed algorithm can obtain feature subsets that included a smaller number of features with superior classification performance than using all features. Hence, FW-GPAWOA can be successfully used in feature selection to reduce the dimensionality while improving the classification performance.

Compared with the other algorithms, the results of Table 6.3 indicate that FW-GPAWOA performs better in terms of classification accuracy and the number of selected features in almost all datasets. For example, compared against BPSO, in all datasets, FW-GPAWOA accomplished higher classification accuracy and selected feature subsets with smaller size. Further, on eight from the ten used datasets (Breast cancer, Lymph, Musk, Sobar, Spect, Spectf, Sports and Wholesale), FW-GPAWOA achieved better results than BDE and BGWO in terms of both the number of features and the classification performance. However, it obtains a lower classification accuracy than BDE and BGWO in Semeion and Vehicle datasets, respectively, but with smaller feature subset size. Accordingly, the proposed algorithm showed a higher ability to better explore the features space than the selected single-objective approaches.

From the above comparison and discussion, we can say that FW-GPAWOA effectively reduces the dimensionality of the feature space by removing the unnecessary (irrelevant/redundant) features, while improving the classification performance in almost all cases.

In the following, we will compare the performance of the proposed method with the selected multi-objective algorithms. For each dataset, FW-GPAWOA, MOGWO and MOPSO generate different subsets of non-dominated features in each of the 10 independent runs. In order to compare these results, the features subsets obtained by each algorithm are combined into one union set. From this union, the non-dominated solutions are identified as the $best$ Pareto front and they are used in the comparison. Figure 6.1 illustrates the non-dominated results of FW-GPAWOA, MOGWO and MOPSO. In each chart, the horizontal axis shows the number of selected features, and the vertical axis shows the classification accuracy.

According to figure 6.1, in eight datasets, we can observe that the best Pareto front produced by FW-GPAWOA, contains at least two solutions that selected less than half of the available features and increased the classification accuracy regarding to that provided using all the features. For example, in Lymphography dataset, FW-GPAWOA selected 3 from 18 features and increased the classification accuracy from 76.92 % to approximately 90 %, and selected around 3 features from the 19 features of Sobar dataset, while it was able to increase the classification accuracy until 100 %. However, in Breast cancer and Musk datasets, our approach selected only one subset which includes fewer features and obtained higher classification performance than using all features.

Comparing FW-GPAWOA against MOGWO and MOPSO, in six datasets (Breast, Lymph, Sobar, Spect, Spectf and Wholesale), it can be seen that the Pareto fronts resulted by FW-GPAWOA significantly dominate those of MOGWO and MOPSO. In other words, the proposed algorithm outperforms its competitors in terms of both the number of features and classification performance. However, in Vehicle dataset, the Pareto front of FW-GPAWOA is dominated by those of MOGWO and MOPSO, especially for the number of feature criterion. In the other datasets, our method can always get better results than MOGWO and MOPSO in terms of feature subsets size or classification accuracy. For instance, compared with MOGWO on Semeion dataset, FW-GPAWOA obtained some solutions that include smaller features size, but with lower classification performance, and inversely, compared with MOPSO on Sports dataset, the proposed approach can achieve better classification performance, but with a larger feature subset.

In order to further evaluate the performance of the multi-objective algorithms, the hyper-Volume (HV) metric [Auger 2009] is applied to compute the volume of the objective space bounded by the obtained Pareto front and a reference point r. This metric has been frequently used for comparing the multi-objective optimization algorithms as it can evaluate both the convergence and the diversity of solutions.

$$HV = volume\Big(\bigcup_{s \in P} v(s, r)\Big) \tag{6.11}$$

For each algorithm, 10 Hyper-volumes over the 10 independent runs are calculated based on the obtained testing Pareto fronts, and then Wilcoxon test (with significance level α equal to 5 %) is applied to check, according to p_value, whether there exists a significant difference between the algorithms by confirming the null hypothesis H_0 (no significant difference between the results of a pair

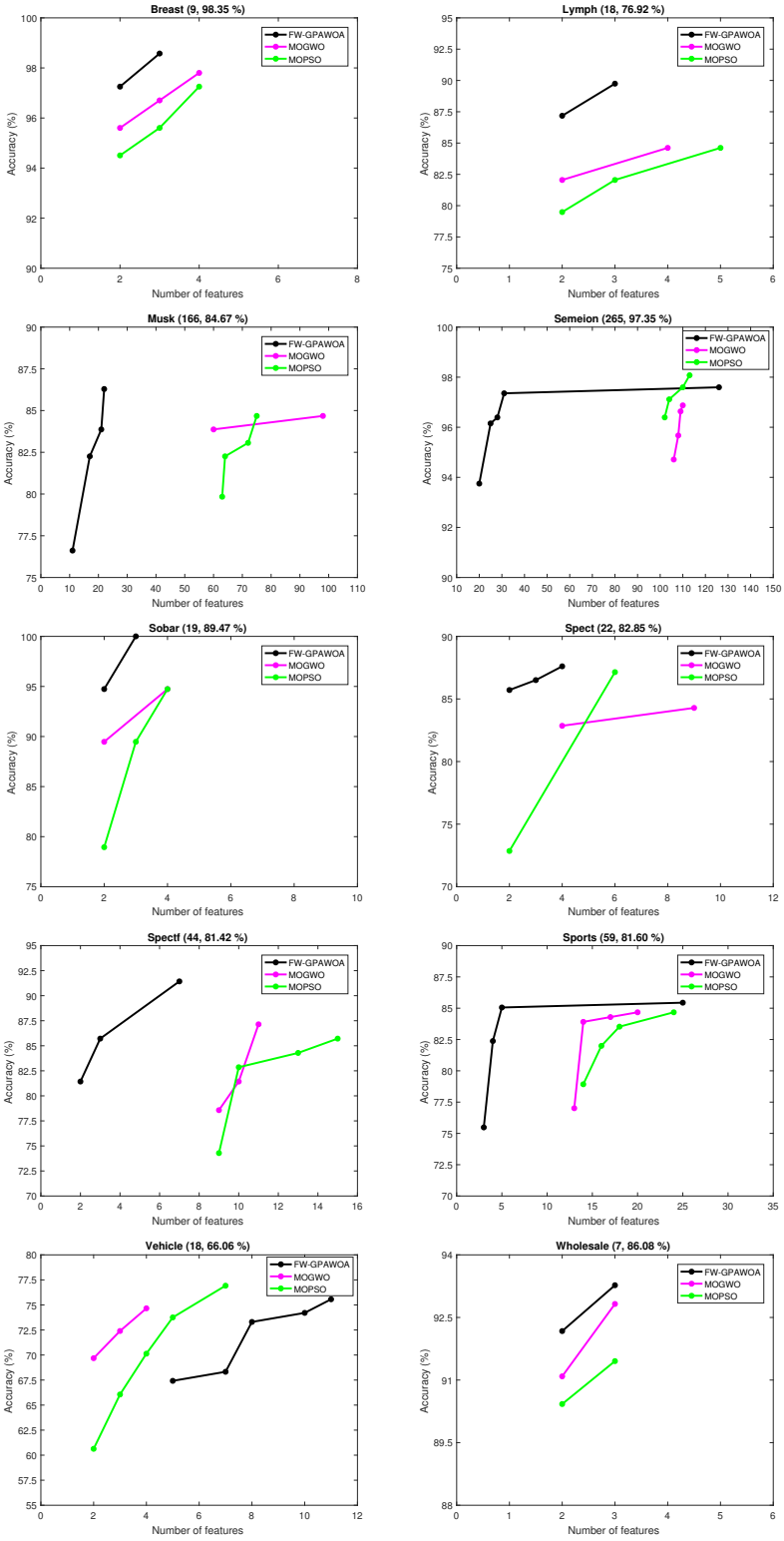

Figure 6.1: Experimental results of FW-GPAWOA, MOGWO and MOPSO

algorithms) if $p\_value > \alpha$, or confirming the alternative hypothesis H_1 (there's a significant difference between a pair algorithms) if $p\_value < \alpha$. We performed the Wilcoxon test between FW-GPAWOA and each peer algorithm (MOGWO, MOPSO) on the results of HV-metric for each dataset. We present the results of FW-GPAWOA as μ_1 and the results of each peer algorithm as μ_2 and we define the following hypothesis:

- $H_0 : \mu_1 = \mu_2$, which indicates that the results of FW-GPAWOA are similar "=" to the results of each peer approach.

- $H_1 : \mu_1 \neq \mu_2$, which indicates that the results of FW-GPAWOA are significantly better "+" (or worse "-") than those of each peer algorithm.

Table 6.4: The results of HV metric of FW-GPAWOA, MOGWO and MOPSO

Algorithms	FW-GPAWOA			MOGWO			MOPSO		
	Average	Best	Worst	Average	Best	Worst	Average	Best	Worst
Breast cancer	**0.8237**	0.8498	0.8061	0.7988	0.8165	0.7017	0.8108	0.8144	0.8084
Lymph	0.6082	0.7453	0.4169	**0.7564**	0.8894	0.6931	0.7228	0.8339	0.6949
Musk	**1.0240**	1.0540	0.9777	0.7344	0.7964	0.7119	0.7094	0.7537	0.6832
Semeion	**0.5094**	0.5258	0.5026	0.3327	0.3450	0.3239	0.3505	0.3599	0.3366
Sobar	**0.8598**	0.9302	0.7558	0.8139	0.8550	0.7182	0.6720	0.8276	0.3693
Spect	**0.9328**	0.9990	0.8192	0.7253	0.7967	0.6802	0.8233	0.8833	0.7484
Spectf	**1.2404**	1.3120	1.1358	1.0126	1.0633	0.9410	0.8233	0.8833	0.7484
Sports	**1.7042**	1.7648	1.6217	1.3923	1.5150	1.2931	1.3997	1.4851	1.3461
Vehicle	0.4819	0.6168	0.4264	**0.6739**	0.7649	0.6055	0.6365	0.6781	0.5797
Wholesale	**0.7035**	0.7079	0.6879	0.5565	0.6492	0.4715	0.6823	0.6946	0.6679

The results of HV-metric, as shown in Table 6.5, indicate that FW-GPAWOA achieves larger HV values for eight out of the ten datasets. Comparing by MOGWO, the proposed method outperforms its competitor on eight cases, and compared against MOPSO, the results show that FW-GPAWOA has better HV-metric on all datasets. Therefore, we can say that our algorithm presents solutions near to the true Pareto front as well as preserving the diversity within the whale's population. Additionally, the results of p-value of Wilcoxon test, as listed in Table 6.5, indicate that the HV-metrics obtained by FW-GPAWOA were significantly better than those obtained by MOGWO for seven datasets, similar for one dataset, and significantly worse for two datasets, while the superior performance of the proposed algorithm is more obvious with respect to MOPSO algorithm. Broadly speaking and for the 20 p-values (2 algorithms $\times$ 10 datasets),

Table 6.5: The p values of Wilcoxon test for HV metric

FW-GPAWOA vs				
	MOGWO	**Sig. diff**	**MOPSO**	**Sig. diff**
Breast	**4.860E-3**	+	**1.697E-2**	+
Lymph	**1.213E-2**	-	0.060	=
Musk	**7.936E-3**	+	**7.936E-3**	+
Semeion	**1.082E-5**	+	**1.082E-5**	+
Sobar	0.089	=	**8.324E-3**	+
Spect	**1.082E-5**	+	**5.033E-3**	+
Spectf	**1.082E-5**	+	**1.082E-5**	+
Sports	**1.082E-5**	+	**1.082E-5**	+
Vehicle	**7.577E-5**	-	**4.330E-3**	-
Wholesale	**1.418E-2**	+	**1.082E-5**	+

our algorithm obtains similar or significantly better results for 17 cases, and it gets significantly worse results than its counterparts only for 3 cases.

The computational cost of FW-GPAWOA, MOGWO and MOPSO is also compared in our simulations. Table 6.6 shows the average running time (in minutes) of over the 10 independent runs. Here, it shoulde be remember that all algorithms have the same population size, the same number of iterations, and they have been executed on the same machine.

Table 6.6: The average running time consumed by the algorithms (unit: minutes).

	FW-GPAWOA	MOGWO	MOPSO
Breast	**2.62**	**3.73**	4.25
Lymph	4.42	**4.15**	4.35
Musk	**3.54**	3.66	4.02
Semeion	6.97	**6.55**	7.63
Sobar	3.87	**3.49**	3.57
Spect	4.18	**3.81**	4.04
Spectf	4.23	**3.88**	3.98
Sports	4.30	**3.76**	4.12
Vehicle	4.85	4.99	**4.33**
Wholesale	**2.29**	3.64	4.21

From Table 6.6, it can be seen that our approach consumes a longer running time than the other approaches for most datasets. Although the proposed algorithm selected smaller feature subsets, which be supposed to lead to less wrapper evaluation, hence, low computational time should be required. So, the reason why

the proposed approach spent more time can be explained, mainly, because it uses the Crowding distance (that requires high computational cost) in the archiving strategy and in the deletion process whenever the external archive is full. The other reason may be that MOGWO and MOPSO use the ratio between the selected features and the original features set as a filter fitness function, which requires obviously less computational effort than using mutual information. However, the growth in the running time consumed by FW-GPAWOA was acceptable for effectively reducing the dimensionality and keeping a highest prediction accuracy. Indeed, the FW-GPAWOA tends to increase the computational time in the hope of improving its performance.

As a summary, all the above results showed that our algorithm gets a superior performance and outperforms the benchmarking algorithms in almost all cases. The experiments demonstrated that the proposed algorithm could effectively explore the Pareto front to accomplish the main challenges of the feature selection problem which are the curse of dimensionality of the features space and the improvement of the classification performance.

6.4 Chapter summary

This chapter presented a new multi-objective optimizer for solving feature selection problem. The proposed algorithm investigates our GPAWOA algorithm and combines filter and wrapper models into a single scheme in the hope of benefits from each model's merits. Therefore, mutual information and KNN classifier are used as filter and wrapper evaluation criteria during the training process. In addition, the hyperbolic tangent function is employed to make GPAWOA able to deal with discrete problems.

A comparative study with five well-regarded algorithms has been performed on ten benchmark datasets. Experimental results show that the proposed algorithm generally outperforms the selected approaches in terms of both number of features and classification performance. However, we have observed that the proposed algorithm usually requires more running time due to the crowding distance computation and the used filter function. Therefore, as part of future work, we plan to investigate other objective functions with the aim of maintaining better performance without increasing the running time. The ration between the selected features and the original set may be a future direction.

CHAPTER 7

Conclusions and future works

Contents

7.1 Conclusions

This thesis focuses, mainly, on the impactful application of the concept of multi-objective optimization in machine learning. The overall goal was to explore the ability of the evolutionary techniques faced on the challenges of machine learning by developing a new multi-objective optimization algorithm for addressing feature selection problem. This goal was successfully achieved by investigating two main resources: The first is the collective intelligence of population-based bio-inspired metaheuristics, while the second resource is the concepts of filter and wrapper evaluations. Hence, two methods, namely GPAWOA and FW-GPAWOA, were developed for solving multi-objective optimization problems and tackling the feature selection problem, respectively.

The first proposed approach, i.e., GPAWOA algorithm, extended the standard Whale Optimization Algorithm in order to be able to solve multi-objective problems. GPAWOA uses the concept of the Pareto dominance and equipped with an external-archive to keep the best configurations at the aim of maintaining the elitism. Moreover, the crowding distance computation was used in the archiving strategy and the deletion process for maintaining as well as possible the diversity of solutions in the search space.

In this thesis, we have also developed a new algorithm called FW-GPAWOA for solving feature selection problem. The proposed algorithm combines filter and wrapper models into the GPAWOA algorithm during the training process to better explore the feature space. Since that the feature selection is a discrete problem, we have adapted the proposed algorithm to be able to deal with discrete problems. In addition, we have adopted the mutual information and a classifier learning as

filter and wrapper evaluations in order to handle the "curse of dimensionality" of the feature space while enhancing the classification performance. The results have shown a clear pattern that the proposed approach can be effectively used to treat feature selection problem for reducing the number of features and achieving similar or even better classification performance than using all the original features.

Finally, we can say that this thesis has achieved the following research objectives:

- Discovering the domain of multi-objective optimization: its definition, modeling, challenges, and its hight relationship with the real-world applications.

- Discovering the domain of feature selection: its problems, its methods, and its importance in machine learning.

- Developing effective algorithms based on the domains evoked below to solve optimization problems and adapting them to machine learning applications.

7.2 Future works

Understanding in depth the mentioned above research lines and propose substantial contributions in this field remains my long term objective. This thesis is a first step in this investigation process which will need further efforts and years of training and work. My short-term objectives consist in completing the works initiated in the present thesis, namely:

- Concretely, develop a new population-based metaheuristic for solving single and multi-objective optimization problems

- Applying the proposed GPAWOA for other machine learning problems such as the optimization of Support vector machine (SVM) parameters.

- Using and even developing other techniques and mechanisms for solving optimization problems.

- Addressing the application of machine learning and feature selection in the real life such as medical field.

Bibliography

[Aggarwal 2014] Charu C Aggarwal. Data classification: algorithms and applications. CRC press, 2014. (Cited on page 9.)

[Akbari 2012] Reza Akbari, Ramin Hedayatzadeh, Koorush Ziarati and Bahareh Hassanizadeh. *A multi-objective artificial bee colony algorithm*. Swarm and Evolutionary Computation, vol. 2, pages 39–52, 2012. (Cited on page 3.)

[Alaya 2007] Ines Alaya, Christine Solnon and Khaled Ghedira. *Ant colony optimization for multi-objective optimization problems*. In 19th IEEE international conference on tools with artificial intelligence (ICTAI 2007), volume 1, pages 450–457. IEEE, 2007. (Cited on page 58.)

[Angus 2007] Daniel Angus. *Crowding population-based ant colony optimisation for the multi-objective travelling salesman problem*. In 2007 IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making, pages 333–340. IEEE, 2007. (Cited on page 58.)

[Arora 2009] Sanjeev Arora and Boaz Barak. Computational complexity: a modern approach. Cambridge University Press, 2009. (Cited on page 23.)

[Askarzadeh 2016] Alireza Askarzadeh. *A novel metaheuristic method for solving constrained engineering optimization problems: crow search algorithm*. Computers & Structures, vol. 169, pages 1–12, 2016. (Cited on page 79.)

[Auger 2009] Anne Auger, Johannes Bader, Dimo Brockhoff and Eckart Zitzler. *Theory of the hypervolume indicator: optimal distributions and the choice of the reference point*. In Proceedings of the tenth ACM SIGEVO workshop on Foundations of genetic algorithms, pages 87–102, 2009. (Cited on page 93.)

[Barnhart 1998] Cynthia Barnhart, Ellis L Johnson, George L Nemhauser, Martin WP Savelsbergh and Pamela H Vance. *Branch-and-price: Column generation for solving huge integer programs*. Operations research, vol. 46, no. 3, pages 316–329, 1998. (Cited on page 27.)

[Bartels 1969] Richard H Bartels and Gene H Golub. *The simplex method of linear programming using LU decomposition*. Communications of the ACM, vol. 12, no. 5, pages 266–268, 1969. (Cited on page 26.)

[Boser 1992] Bernhard E Boser, Isabelle M Guyon and Vladimir N Vapnik. *A training algorithm for optimal margin classifiers*. In Proceedings of the fifth annual workshop on Computational learning theory, pages 144–152, 1992. (Cited on page 11.)

[BoussaïD 2013] Ilhem BoussaïD, Julien Lepagnot and Patrick Siarry. *A survey on optimization metaheuristics*. Information sciences, vol. 237, pages 82–117, 2013. (Cited on pages 27 and 29.)

[Byun 2002] Hyeran Byun and Seong-Whan Lee. *Applications of support vector machines for pattern recognition: A survey*. In International Workshop on Support Vector Machines, pages 213–236. Springer, 2002. (Cited on page 12.)

[Černỳ 1985] Vladimír Černỳ. *Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm*. Journal of optimization theory and applications, vol. 45, no. 1, pages 41–51, 1985. (Cited on page 29.)

[Chaiyaratana 1997] N Chaiyaratana and AMS Zalzala. *Recent developments in evolutionary and genetic algorithms: theory and applications*. 1997. (Cited on page 32.)

[Chandrashekar 2014] Girish Chandrashekar and Ferat Sahin. *A survey on feature selection methods*. Computers & Electrical Engineering, vol. 40, no. 1, pages 16–28, 2014. (Cited on page 18.)

[Chen 2009] Jie Chen, Bin Xin, Zhihong Peng, Lihua Dou and Juan Zhang. *Optimal contraction theorem for exploration–exploitation tradeoff in search and optimization*. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, vol. 39, no. 3, pages 680–691, 2009. (Cited on page 25.)

[Chomboon 2015] Kittipong Chomboon, Pasapitch Chujai, Pongsakorn Teerarassamee, Kittisak Kerdprasop and Nittaya Kerdprasop. *An empirical study of distance metrics for k-nearest neighbor algorithm*. In Proceedings of the 3rd international conference on industrial application engineering, pages 1–6, 2015. (Cited on page 11.)

[Chu 2006] Shu-Chuan Chu, Pei-Wei Tsai and Jeng-Shyang Pan. *Cat swarm optimization*. In Pacific Rim international conference on artificial intelligence, pages 854–858. Springer, 2006. (Cited on page 58.)

[Coello 2004] Carlos A Coello Coello, Gregorio Toscano Pulido and M Salazar Lechuga. *Handling multiple objectives with particle swarm optimization*. IEEE

Transactions on evolutionary computation, vol. 8, no. 3, pages 256–279, 2004. (Cited on pages vi, 52, 53, 64 and 90.)

[Cook 1971] Stephen A Cook. *The complexity of theorem-proving procedures*. In Proceedings of the third annual ACM symposium on Theory of computing, pages 151–158, 1971. (Cited on page 23.)

[Cortes 1995] Corinna Cortes and Vladimir Vapnik. *Support-vector networks*. Machine learning, vol. 20, no. 3, pages 273–297, 1995. (Cited on page 11.)

[Dash 1997] Manoranjan Dash and Huan Liu. *Feature selection for classification*. Intelligent data analysis, vol. 1, no. 3, pages 131–156, 1997. (Cited on pages vi, 2, 15, 16, 17 and 86.)

[Dash 2003] Manoranjan Dash and Huan Liu. *Consistency-based search in feature selection*. Artificial intelligence, vol. 151, no. 1-2, pages 155–176, 2003. (Cited on page 18.)

[Deb 2002a] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal and TAMT Meyarivan. *A fast and elitist multiobjective genetic algorithm: NSGA-II*. IEEE transactions on evolutionary computation, vol. 6, no. 2, pages 182–197, 2002. (Cited on pages 3, 49, 57, 60 and 64.)

[Deb 2002b] Kalyanmoy Deb, Lothar Thiele, Marco Laumanns and Eckart Zitzler. *Scalable multi-objective optimization test problems*. In Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600), volume 1, pages 825–830. IEEE, 2002. (Cited on page 64.)

[Deb 2014] Kalyanmoy Deb. *Multi-objective optimization*. In Search methodologies, pages 403–449. Springer, 2014. (Cited on pages 3, 39 and 42.)

[Derrac 2011] Joaquín Derrac, Salvador García, Daniel Molina and Francisco Herrera. *A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms*. Swarm and Evolutionary Computation, vol. 1, no. 1, pages 3–18, 2011. (Cited on page 77.)

[Doerner 2004] Karl Doerner, Walter J Gutjahr, Richard F Hartl, Christine Strauss and Christian Stummer. *Pareto ant colony optimization: A metaheuristic approach to multiobjective portfolio selection*. Annals of operations research, vol. 131, no. 1-4, pages 79–99, 2004. (Cited on page 58.)

[Dorigo 1991] Marco Dorigo, Alberto Colorni and Vittorio Maniezzo. *Distributed optimization by ant colonies*, 1991. (Cited on pages 34 and 58.)

[Dorigo 2006] Marco Dorigo, Mauro Birattari and Thomas Stutzle. *Ant colony optimization*. IEEE computational intelligence magazine, vol. 1, no. 4, pages 28–39, 2006. (Cited on page 34.)

[Dougherty 1995] James Dougherty, Ron Kohavi and Mehran Sahami. *Supervised and unsupervised discretization of continuous features*. In Machine Learning Proceedings 1995, pages 194–202. Elsevier, 1995. (Cited on page 9.)

[Drexl 1988] Andreas Drexl. *A simulated annealing approach to the multiconstraint zero-one knapsack problem*. Computing, vol. 40, no. 1, pages 1–8, 1988. (Cited on page 30.)

[Elmohamed 1997] MA Saleh Elmohamed, Paul Coddington and Geoffrey Fox. *A comparison of annealing techniques for academic course scheduling*. In International Conference on the Practice and Theory of Automated Timetabling, pages 92–112. Springer, 1997. (Cited on page 30.)

[Emmerich 2018] Michael TM Emmerich and André H Deutz. *A tutorial on multiobjective optimization: fundamentals and evolutionary methods*. Natural computing, vol. 17, no. 3, pages 585–609, 2018. (Cited on page 40.)

[Fieldsend 2003] Jonathan E Fieldsend, Richard M Everson and Sameer Singh. *Using unconstrained elite archives for multiobjective optimization*. IEEE Transactions on Evolutionary Computation, vol. 7, no. 3, pages 305–323, 2003. (Cited on page 47.)

[Fonseca 1993] Carlos M Fonseca, Peter J Fleming*et al. Genetic Algorithms for Multiobjective Optimization: FormulationDiscussion and Generalization.* In Icga, volume 93, pages 416–423. Citeseer, 1993. (Cited on page 45.)

[Frank 2010] Andrew Frank. *UCI machine learning repository*. http://archive. ics. uci. edu/ml, 2010. (Cited on page 90.)

[Ghahramani 2003] Zoubin Ghahramani. *Unsupervised learning*. In Summer School on Machine Learning, pages 72–112. Springer, 2003. (Cited on page 2.)

[Glover 1986] Fred Glover. *Future paths for integer programming and links to ar tifi cial intelli g en ce*. Computers operations research, vol. 13, no. 5, pages 533–549, 1986. (Cited on page 28.)

[Gogna 2013] Anupriya Gogna and Akash Tayal. *Metaheuristics: review and application*. Journal of Experimental & Theoretical Artificial Intelligence, vol. 25, no. 4, pages 503–526, 2013. (Cited on page 27.)

[Goldberg 1989] David E Goldberg. *Genetic algorithms in search*. Optimization, and MachineLearning, 1989. (Cited on page 31.)

[Goldberg 1991] David E Goldberg and Kalyanmoy Deb. *A comparative analysis of selection schemes used in genetic algorithms*. In Foundations of genetic algorithms, volume 1, pages 69–93. Elsevier, 1991. (Cited on page 31.)

[Hamdani 2007] Tarek M Hamdani, Jin-Myung Won, Adel M Alimi and Fakhri Karray. *Multi-objective feature selection with NSGA II*. In International conference on adaptive and natural computing algorithms, pages 240–247. Springer, 2007. (Cited on page 3.)

[Hancer 2015] Emrah Hancer, Bing Xue, Mengjie Zhang, Dervis Karaboga and Bahriye Akay. *A multi-objective artificial bee colony approach to feature selection using fuzzy mutual information*. In 2015 IEEE Congress on Evolutionary Computation (CEC), pages 2420–2427. IEEE, 2015. (Cited on page 4.)

[Hassan 2018] Gehad Hassan and Aboul Ella Hassanien. *Retinal fundus vasculature multilevel segmentation using whale optimization algorithm*. Signal, Image and Video Processing, vol. 12, no. 2, pages 263–270, 2018. (Cited on page 37.)

[Holland 1975] John Holland. *Adaptation in natural and artificial systems: an introductory analysis with application to biology*. Control and artificial intelligence, 1975. (Cited on page 30.)

[Horn 1994] Jeffrey Horn, Nicholas Nafpliotis and David E Goldberg. *A niched Pareto genetic algorithm for multiobjective optimization*. In Proceedings of the first IEEE conference on evolutionary computation. IEEE world congress on computational intelligence, pages 82–87. Ieee, 1994. (Cited on pages vi, 46 and 47.)

[Hwang 1980] Ching-Lai Hwang, Sudhakar R Paidy, Kwangsun Yoon and Abu Syed Md Masud. *Mathematical programming with multiple objectives: A tutorial*. Computers & Operations Research, vol. 7, no. 1-2, pages 5–31, 1980. (Cited on page 43.)

[Indyk 1998] Piotr Indyk and Rajeev Motwani. *Approximate nearest neighbors: towards removing the curse of dimensionality*. In Proceedings of the thirtieth

annual ACM symposium on Theory of computing, pages 604–613, 1998. (Cited on page 10.)

[Jain 2000] Anil K Jain, Robert P. W. Duin and Jianchang Mao. *Statistical pattern recognition: A review*. IEEE Transactions on pattern analysis and machine intelligence, vol. 22, no. 1, pages 4–37, 2000. (Cited on page 13.)

[Jin 2002] Yaochu Jin. *Effectiveness of weighted aggregation of objectives for evolutionary multiobjective optimization: methods, analysis and applications*, 2002. (Cited on page 44.)

[Jolliffe 2016] Ian T Jolliffe and Jorge Cadima. *Principal component analysis: a review and recent developments*. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, vol. 374, no. 2065, page 20150202, 2016. (Cited on page 16.)

[Juan 2000] Alfons Juan and Enrique Vidal. *Comparison of four initialization techniques for the k-medians clustering algorithm*. In Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR), pages 842–852. Springer, 2000. (Cited on page 14.)

[Karp 1972] Richard M Karp. *Reducibility among combinatorial problems*. In Complexity of computer computations, pages 85–103. Springer, 1972. (Cited on page 23.)

[Kennedy 1995] James Kennedy and Russell Eberhart. *Particle swarm optimization*. In Proceedings of ICNN'95-International Conference on Neural Networks, volume 4, pages 1942–1948. IEEE, 1995. (Cited on page 32.)

[Kirkpatrick 1983] Scott Kirkpatrick, C Daniel Gelatt and Mario P Vecchi. *Optimization by simulated annealing*. science, vol. 220, no. 4598, pages 671–680, 1983. (Cited on page 29.)

[Knowles 1999] Joshua Knowles and David Corne. *The pareto archived evolution strategy: A new baseline algorithm for pareto multiobjective optimisation*. In Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), volume 1, pages 98–105. IEEE, 1999. (Cited on page 47.)

[Knowles 2000] Joshua D Knowles and David W Corne. *Approximating the nondominated front using the Pareto archived evolution strategy*. Evolutionary computation, vol. 8, no. 2, pages 149–172, 2000. (Cited on page 48.)

[Kohavi 1997] Ron Kohavi, George H John*et al.* *Wrappers for feature subset selection*. Artificial intelligence, vol. 97, no. 1-2, pages 273–324, 1997. (Cited on page 3.)

[Land 1960] AH Land and AG Doig. *An Automatic Method of Solving Discrete Programming Problems*. Econometrica: Journal of the Econometric Society, pages 497–520, 1960. (Cited on page 27.)

[Langley 1994] Pat Langley*et al.* *Selection of relevant features in machine learning*. In Proceedings of the AAAI Fall symposium on relevance, volume 184, pages 245–271, 1994. (Cited on page 16.)

[Laumanns 2001] Marco Laumanns, Lothar Thiele, Kalyanmoy Deb and Eckart Zitzler. *On the convergence and diversity-preservation properties of multi-objective evolutionary algorithms*. TIK-Report, vol. 108, 2001. (Cited on page 47.)

[Lawler 1966] Eugene L Lawler and David E Wood. *Branch-and-bound methods: A survey*. Operations research, vol. 14, no. 4, pages 699–719, 1966. (Cited on page 27.)

[Li 2003] Xiaodong Li. *A non-dominated sorting particle swarm optimizer for multiobjective optimization*. In Genetic and Evolutionary Computation Conference, pages 37–48. Springer, 2003. (Cited on pages 3 and 58.)

[Liu 1998] Huan Liu and Hiroshi Motoda. Feature extraction, construction and selection: A data mining perspective, volume 453. Springer Science & Business Media, 1998. (Cited on page 2.)

[Liu 2005] Huan Liu and Lei Yu. *Toward integrating feature selection algorithms for classification and clustering*. IEEE Transactions on knowledge and data engineering, vol. 17, no. 4, pages 491–502, 2005. (Cited on pages 17 and 20.)

[MacQueen 1967] James MacQueen*et al.* *Some methods for classification and analysis of multivariate observations*. In Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, volume 1, pages 281–297. Oakland, CA, USA, 1967. (Cited on page 13.)

[Malek 1989] Miroslaw Malek, Mohan Guruswamy, Mihir Pandya and Howard Owens. *Serial and parallel simulated annealing and tabu search algorithms for the traveling salesman problem*. Annals of Operations Research, vol. 21, no. 1, pages 59–84, 1989. (Cited on page 30.)

[Marler 2004] R Timothy Marler and Jasbir S Arora. *Survey of multi-objective optimization methods for engineering*. Structural and multidisciplinary optimization, vol. 26, no. 6, pages 369–395, 2004. (Cited on page 43.)

[Mirjalili 2014] Seyedali Mirjalili, Seyed Mohammad Mirjalili and Andrew Lewis. *Grey wolf optimizer*. Advances in engineering software, vol. 69, pages 46–61, 2014. (Cited on pages 4 and 58.)

[Mirjalili 2016a] Seyedali Mirjalili and Andrew Lewis. *The whale optimization algorithm*. Advances in engineering software, vol. 95, pages 51–67, 2016. (Cited on pages vi, 35 and 58.)

[Mirjalili 2016b] Seyedali Mirjalili, Shahrzad Saremi, Seyed Mohammad Mirjalili and Leandro dos S Coelho. *Multi-objective grey wolf optimizer: a novel algorithm for multi-criterion optimization*. Expert Systems with Applications, vol. 47, pages 106–119, 2016. (Cited on pages 58, 64 and 90.)

[Moore 1999] Jacqueline Moore and Richard Chapman. *Application of particle swarm to multiobjective optimization*. Department of Computer Science and Software Engineering, Auburn University, vol. 32, 1999. (Cited on page 58.)

[Muñoz 2018] Mario A Muñoz, Laura Villanova, Davaatseren Baatar and Kate Smith-Miles. *Instance spaces for machine learning classification*. Machine Learning, vol. 107, no. 1, pages 109–147, 2018. (Cited on page 9.)

[Narasimhan 1980] Ram Narasimhan. *Goal programming in a fuzzy environment*. Decision sciences, vol. 11, no. 2, pages 325–336, 1980. (Cited on page 44.)

[Nguyen 2008] Thuy TT Nguyen and Grenville Armitage. *A survey of techniques for internet traffic classification using machine learning*. IEEE communications surveys & tutorials, vol. 10, no. 4, pages 56–76, 2008. (Cited on pages 1 and 2.)

[Oliveira 2002] Luiz S Oliveira, Robert Sabourin, Flávio Bortolozzi and Ching Y Suen. *Feature selection using multi-objective genetic algorithms for handwritten digit recognition*. In Object recognition supported by user interaction for service robots, volume 1, pages 568–571. IEEE, 2002. (Cited on page 16.)

[Padberg 1991] Manfred Padberg and Giovanni Rinaldi. *A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems*. SIAM review, vol. 33, no. 1, pages 60–100, 1991. (Cited on page 27.)

[Papadimitriou 2003] Christos H Papadimitriou. Computational complexity. John Wiley and Sons Ltd., 2003. (Cited on page 23.)

[Park 2009] Hae-Sang Park and Chi-Hyuck Jun. *A simple and fast algorithm for K-medoids clustering*. Expert systems with applications, vol. 36, no. 2, pages 3336–3341, 2009. (Cited on page 14.)

[Pawlak 1995] Zdzislaw Pawlak, Jerzy Grzymala-Busse, Roman Slowinski and Wojciech Ziarko. *Rough sets*. Communications of the ACM, vol. 38, no. 11, pages 88–95, 1995. (Cited on page 4.)

[Pimple 2016] Suruchi Pimple. *Application of Support Vector Machine for diagnosis of diabetes: A systematic review*. IOSR Journal of Computer Engineering (IOSR-JCE), 2016. (Cited on page 12.)

[Pradhan 2012] Pyari Mohan Pradhan and Ganapati Panda. *Solving multiobjective problems using cat swarm optimization*. Expert Systems with Applications, vol. 39, no. 3, pages 2956–2964, 2012. (Cited on page 58.)

[Puchinger 2005] Jakob Puchinger and Günther R Raidl. *Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification*. In International work-conference on the interplay between natural and artificial computation, pages 41–53. Springer, 2005. (Cited on page 26.)

[Rao 1987] Singiresu S Rao. *Game theory approach for multiobjective structural optimization*. Computers & Structures, vol. 25, no. 1, pages 119–127, 1987. (Cited on page 44.)

[Rao 2019] Singiresu S Rao. Engineering optimization: theory and practice. John Wiley & Sons, 2019. (Cited on page 82.)

[Ray 2002] Tapabrata Ray and KM Liew. *A swarm metaphor for multiobjective design optimization*. Engineering optimization, vol. 34, no. 2, pages 141–153, 2002. (Cited on page 58.)

[Reyes-Sierra 2006] Margarita Reyes-Sierra, CA Coello Coello*et al. Multi-objective particle swarm optimizers: A survey of the state-of-the-art*. International journal of computational intelligence research, vol. 2, no. 3, pages 287–308, 2006. (Cited on pages v, 52 and 54.)

[Rostami 2017] Shahin Rostami, Ferrante Neri and Michael Epitropakis. *Progressive preference articulation for decision making in multi-objective optimisation*

problems. Integrated Computer-Aided Engineering, vol. 24, no. 4, pages 315–335, 2017. (Cited on page 43.)

[Russell 2003] Stuart J Russell and J Stuart. *Norvig*. Artificial Intelligence: A Modern Approach, pages 111–114, 2003. (Cited on page 2.)

[Sadollah 2015] Ali Sadollah, Hadi Eskandar and Joong Hoon Kim. *Water cycle algorithm for solving constrained multi-objective optimization problems*. Applied Soft Computing, vol. 27, pages 279–298, 2015. (Cited on page 79.)

[Saha 2017] Arindita Saha and Lalit Chandra Saikia. *Utilisation of ultra-capacitor in load frequency control under restructured STPP-thermal power systems using WOA optimised PIDN-FOPD controller*. IET Generation, Transmission & Distribution, vol. 11, no. 13, pages 3318–3331, 2017. (Cited on page 37.)

[Sahoo 2017] Anita Sahoo and Satish Chandra. *Multi-objective grey wolf optimizer for improved cervix lesion classification*. Applied Soft Computing, vol. 52, pages 64–80, 2017. (Cited on page 4.)

[Schaffer 1985] J David Schaffer. *Multiple objective optimization with vector evaluated genetic algorithms*. In Proceedings of the first international conference on genetic algorithms and their applications, 1985. Lawrence Erlbaum Associates. Inc., Publishers, 1985. (Cited on page 44.)

[Schoen 1991] Fabio Schoen. *Stochastic techniques for global optimization: A survey of recent advances*. Journal of Global Optimization, vol. 1, no. 3, pages 207–228, 1991. (Cited on page 25.)

[Schölkopf 2002] Bernhard Schölkopf, Alexander J Smola, Francis Bach*et al.* Learning with kernels: support vector machines, regularization, optimization, and beyond. MIT press, 2002. (Cited on page 12.)

[Schott 1995] Jason R Schott. *Fault tolerant design using single and multicriteria genetic algorithm optimization.* Rapport technique, Air force inst of tech Wright-Patterson afb OH, 1995. (Cited on page 55.)

[Selim 1984] Shokri Z Selim and Mohamed A Ismail. *K-means-type algorithms: A generalized convergence theorem and characterization of local optimality*. IEEE Transactions on pattern analysis and machine intelligence, no. 1, pages 81–87, 1984. (Cited on page 13.)

[Shannon 1948] Claude E Shannon. *A mathematical theory of communication*. The Bell system technical journal, vol. 27, no. 3, pages 379–423, 1948. (Cited on page 87.)

[Srinivas 1994] Nidamarthi Srinivas and Kalyanmoy Deb. *Muiltiobjective optimization using nondominated sorting in genetic algorithms*. Evolutionary computation, vol. 2, no. 3, pages 221–248, 1994. (Cited on page 3.)

[Suman 2006] Balram Suman and Prabhat Kumar. *A survey of simulated annealing as a tool for single and multiobjective optimization*. Journal of the operational research society, vol. 57, no. 10, pages 1143–1160, 2006. (Cited on page 30.)

[Sweilam 2010] Nasser H Sweilam, AA Tharwat and NK Abdel Moniem. *Support vector machine for diagnosis cancer disease: A comparative study*. Egyptian Informatics Journal, vol. 11, no. 2, pages 81–92, 2010. (Cited on page 12.)

[Talbi 2009] El-Ghazali Talbi. Metaheuristics: from design to implementation, volume 74. John Wiley & Sons, 2009. (Cited on page 26.)

[Talbi 2012] El-Ghazali Talbi, Matthieu Basseur, Antonio J Nebro and Enrique Alba. *Multi-objective optimization using metaheuristics: non-standard algorithms*. International Transactions in Operational Research, vol. 19, no. 1-2, pages 283–305, 2012. (Cited on pages 39 and 42.)

[Too 2018] Jingwei Too, Abdul Rahim Abdullah, Norhashimah Mohd Saad, Nursabillilah Mohd Ali and Weihown Tee. *A new competitive binary Grey Wolf Optimizer to solve the feature selection problem in EMG signals classification*. Computers, vol. 7, no. 4, page 58, 2018. (Cited on page 90.)

[Too 2019a] Jingwei Too, Abdul Rahim Abdullah and Norhashimah Mohd Saad. *Hybrid Binary Particle Swarm Optimization Differential Evolution-Based Feature Selection for EMG Signals Classification*. Axioms, vol. 8, no. 3, page 79, 2019. (Cited on page 90.)

[Too 2019b] Jingwei Too, Abdul Rahim Abdullah, Norhashimah Mohd Saad and Weihown Tee. *EMG feature selection and classification using a Pbest-guide binary particle swarm optimization*. Computation, vol. 7, no. 1, page 12, 2019. (Cited on page 90.)

[Unler 2010] Alper Unler and Alper Murat. *A discrete particle swarm optimization method for feature selection in binary classification problems*. European Journal of Operational Research, vol. 206, no. 3, pages 528–539, 2010. (Cited on page 86.)

[Van Veldhuizen 2000] David A Van Veldhuizen and Gary B Lamont. *Multiobjective evolutionary algorithms: Analyzing the state-of-the-art*. Evolutionary computation, vol. 8, no. 2, pages 125–147, 2000. (Cited on page 55.)

[Waqas 2009] Kashif Waqas, Rauf Baig and Shahid Ali. *Feature subset selection using multi-objective genetic algorithms*. In 2009 IEEE 13th International Multitopic Conference, pages 1–6. IEEE, 2009. (Cited on page 4.)

[Ward Jr 1963] Joe H Ward Jr. *Hierarchical grouping to optimize an objective function*. Journal of the American statistical association, vol. 58, no. 301, pages 236–244, 1963. (Cited on page 14.)

[Xu 2014] Junqin Xu and Jihui Zhang. *Exploration-exploitation tradeoffs in metaheuristics: Survey and analysis*. In Proceedings of the 33rd Chinese Control Conference, pages 8633–8638. IEEE, 2014. (Cited on page 25.)

[Xue 2012a] Bing Xue, Liam Cervante, Lin Shang, Will N Browne and Mengjie Zhang. *A multi-objective particle swarm optimisation for filter-based feature selection in classification problems*. Connection Science, vol. 24, no. 2-3, pages 91–116, 2012. (Cited on page 3.)

[Xue 2012b] Bing Xue, Mengjie Zhang and Will N Browne. *Particle swarm optimization for feature selection in classification: A multi-objective approach*. IEEE transactions on cybernetics, vol. 43, no. 6, pages 1656–1671, 2012. (Cited on pages 4 and 16.)

[Xue 2014a] Bing Xue, Liam Cervante, Lin Shang, Will N Browne and Mengjie Zhang. *Binary PSO and rough set theory for feature selection: A multi-objective filter based approach*. International Journal of Computational Intelligence and Applications, vol. 13, no. 02, page 1450009, 2014. (Cited on pages 2 and 4.)

[Xue 2014b] Bing Xue, Mengjie Zhang and Will N Browne. *Particle swarm optimisation for feature selection in classification: Novel initialisation and updating mechanisms*. Applied soft computing, vol. 18, pages 261–276, 2014. (Cited on pages 2 and 18.)

[Xue 2015] Bing Xue, Mengjie Zhang and Will N Browne. *A comprehensive comparison on evolutionary feature selection approaches to classification*. International Journal of Computational Intelligence and Applications, vol. 14, no. 02, page 1550008, 2015. (Cited on page 86.)

[Zhang 2007] Qingfu Zhang and Hui Li. *MOEA/D: A multiobjective evolutionary algorithm based on decomposition*. IEEE Transactions on evolutionary computation, vol. 11, no. 6, pages 712–731, 2007. (Cited on pages 57 and 64.)

[Zhou 2011] Aimin Zhou, Bo-Yang Qu, Hui Li, Shi-Zheng Zhao, Ponnuthurai Nagaratnam Suganthan and Qingfu Zhang. *Multiobjective evolutionary algorithms: A survey of the state of the art*. Swarm and Evolutionary Computation, vol. 1, no. 1, pages 32–49, 2011. (Cited on page 52.)

[Zitzler 1998] Eckart Zitzler and Lothar Thiele. *An evolutionary algorithm for multiobjective optimization: The strength pareto approach*. TIK-report, vol. 43, 1998. (Cited on page 48.)

[Zitzler 2000] Eckart Zitzler, Kalyanmoy Deb and Lothar Thiele. *Comparison of multiobjective evolutionary algorithms: Empirical results*. Evolutionary computation, vol. 8, no. 2, pages 173–195, 2000. (Cited on page 64.)

[Zitzler 2001] Eckart Zitzler, Marco Laumanns and Lothar Thiele. *SPEA2: Improving the strength Pareto evolutionary algorithm*. TIK-report, vol. 103, 2001. (Cited on pages vi, 48, 49, 57 and 64.)

[Zitzler 2004] Eckart Zitzler, Marco Laumanns and Stefan Bleuler. *A tutorial on evolutionary multiobjective optimization*. In Metaheuristics for multiobjective optimisation, pages 3–37. Springer, 2004. (Cited on page 47.)

Machine Learning using Multi-Objective Evolutionary Algorithms
Abstract

Broadly speaking, machine learning consists of handling a large amount of data. The the quality of these data affect so much the accuracy of the learning model whatever the performance of the employed learning algorithm. Therefore, a technique should be invoked to improve the representation of the dataset.

Feature selection try to offer to the learning algorithm well-represented dataset by removing irrelevant and redundant features and selecting the most informative features. This act results, mainly, in decreasing the number of features and improving the prediction accuracy of the learning algorithmn. However, the conflicting design between number/accuracy makes feature selection a multi-objective problem. Therefore, it is more suitable to treat such as situation by using a multi-objective optimization algorithm rather than single-objective approach. Consequently, we propose in this thesis, two evolutionary computation algorithms for solving multi-objective optimization problems in general manner, and for tackling feature selection problem.

The first algorithm called Guided Population Archive Whale Optimization Algorithm "GPAWOA". The proposed algorithm represents a viable alternative for solving multi-objective optimization problems. It uses the notion of Pareto dominance to compare between the candidate solutions, adopts an external archive to maintain the elitism concept and guide the population towards promising regions within the search space, and employed the computation of the crowding distance to improve the distribution of solutions.

The second algorithm investigates GPAWOA for addressing feature selection in classification problem. The proposed algorithm, namely FW-GPAWOA, employs a transfert function to make it able to deal with discrete problems, and combines filter and wrapper models into a single system to benefits from each model's merits in order to reduce the feature set cardinality and improve the prediction accuracy of the learning algorithm.

Keywords: Machine Learning; Dimensionality Reduction; Feature Selection; Optimization Problems; Multi-objective Optimization; Evolutionary Computation

Apprentissage Automatique par Algorithmes Evolutionnaires Multiobjectifs

Résumé

En général, l'apprentissage automatique consiste à gérer une grande quantité de données. La qualité de ces données influence tellement sur la précision du modéle d'apprentissage quelque soit la performance de l'algorithme utilisé.

La sélection d'attributs tente à offrir à l'algorithm d'apprentissage un ensemble de données bien représentées en supprimant les attributs non pertinents et redondants, et en sélectionnant les attributs les plus informatrifs. Cet acte mène, principalement, à diminuer le nombre d'attributs et à améliorer la précision de la prédiction de l'algorithme d'apprentissage. Cependant, la conception contradictoire entre nombre / précision fait la sélection d'attributs un problème multiobjectif. Par conséquent, nous proposons dans cette thèse, deux algorithmes évolutionnaires pour résoudre les problèmes d'optimisation multiobjectifs d'une manière générale, et pour aborder le problème de sélection d'attributs.

Dans la premiŕe contribution, nous avons proposé un nouvel algorithme d'optimisation multiobjectifs appelé GPAWOA. L'algorithme proposé utilise la notion de dominance de Paréto, et il maintient un répertoire externe pour sauvegarder les solutions dites "élites". De plus, il utilise le mécanisme de la distance d'encombrement pour avoir des solutions bien réparties dans l'espace de recherche.

Dans la deuxiéme contribution, nous avons appliqué l'algorithme GPAWOA pour résoudre le problème de la sélection d'attributs. Pour cela, nous avons adapté, en premier lieu, l'algorithme GPAWOA avec les problèmes d'optimisation combinatoire, puis, nous avons combiné les approches "filtrantes et enveloppantes" dans un seul système pour bénéficier des avantages de chaque modèle afin de minimiser le nombre d'attributs et maximiser la précision de l'algorithme d'apprentissage.

Keywords: Apprentissage Automatique; Réduction de Dimensionnalité; Sélection d'Attributs; Problèmes d'Optimisation; Optimisation Multiobjectif; Calcul Evolutionnaire

التعلم الآلي باستخدام الخوارزميات التطورية متعددة الأهداف

ملخـــص

اختيار المميزات هي تقنية من التقنيات التي تسمح بتحسين جودة البيانات المراد معالجتها و ذلك من خلال إزالة المميزات الغير هامة، و في المقابل تسمح هاته التقنية بتحديد المميزات الأكثر الأهمية داخل قاعدة البيانات. تطبيق هاته التقنية يساهم، خصوصا، في تخفيض عدد المميزات و بالتالي تخفيض أبعاد البيانات، كما تساهم كذلك في الرفع من دقة التنبؤ لدى خوارزمية التعلم. غير أن التناقض بين الهدفين عدد / دقة يجعل من تقنية اختيار المميزات مشكلة متعددة الأغراض. و بالتالي، نقترح في هاته الأطروحة، خوارزميتين تطوريتان من أجل حل مشاكل التحسين متعددة الأهداف بصورة عامة، و كذا من أجل معالجة مشكلة تحديد المميزات.

الخوارزمية الأولى المقترحة مستوحاة من طريقة الصيد الجماعية لدى الحوت الأزرق و تعتمد على ما يسمى بهيمنة باريتو للمقارنة بين الحلول، كما تستعمل ذاكرة اضافية تسمى "أرشيف" لتخزين أفضل الحلول "النخبة". كما تستخدم تقنية حساب مسافة الإكتظاظ و ذلك من أجل الحصول على حلول جد قريبة من جبهة باريتو و ذات انتشار جيد داخل مجموعة البحث.

أما في المساهمة الثانية فقد قمنا باستغلال الخوارزمية الأولى في حل مشاكل اختيار المميزات. و على هذا الأساس، فإن الخوارزمية الثانية المقترحة تستعمل دالة تحويل و ذلك من أجل التأقلم مع مشاكل التحسين التوافقية، إضافة إلى هذا، فالخوارزمية الثانية المقترحة هي خوارزمية هجينة بين طريقة التصفية و طريقة التغليف و ذلك قصد الاستفادة من مزايا كل طريقة قصد تخفيض عدد المميزات و الرفع من دقة نموذج التعلم.

و خلصنا في الأخير إلى نتائج مشجعة سواء بالنسبة لحل مشاكل التحسين متعددة الأهداف أو مشاكل اختيار المميزات، و ذلك مقارنة بالحلول المتوفرة على الساحة العلمية.

كلمات مفتاحية : التعلم الآلي، تقليص الأبعاد، اختيار المميزات، مشاكل التحسين، التحسين متعدد الأهداف، الخوارزمية التطورية