Faculty of Engineering and Technology

Master of Computing (MCOM)

Effective Customer Churn Prediction (ECCP) Using Parallel Grey Wolf Optimizer

التوقع الفعال لإعراض الزبون عن الخدمة بإستخدام محسن الذئب الرمادي الموازي

Author:

Ahmad Dajani

Supervisor:

Dr. Majdi Mafarja

This Thesis was submitted in partial fulfilment of the requirements for the Master's Degree in Computing from the Faculty of Graduate Studies at Birzeit University, Palestine

February 2020

Effective Customer Churn Prediction (ECCP) Using Parallel Grey Wolf Optimizer

By

Ahmad Dajani

Approved by the thesis committee

Dr. Majdi Mafarja, Birzeit University

Dr. Ahmad Alsadeh, Birzeit University

Dr. Ahmad Afaneh, Birzet university

Date approved: 25/2/2020

Abstract

Metaheuristics algorithms gained the attention of many researchers in different optimization fields. Feature Selection (FS) is combinatorial optimization problem where different metaheuristics algorithms were used to tackle it. Grey Wolf Optimizer (GWO) is a recent population based metaheuristic algorithm that showed a good performance in tackling different optimization problems, including the FS problem. As other global optimization algorithms, GWO suffers from a set of drawbacks (e.g. stacking at the local optima and population diversity problem) that may degrade its performance.

In this thesis, a novel parallel GWO is proposed in the aim of maintaining a reasonable diversity of the population, in addition to helping the algorithm to escape the local optima. Two parallel models were proposed; the first one called homogeneous GWO, where four copies of a GWO were employed on the same population. While in the second approach, which is called heterogeneous GWO, four copies of the GWO, each one with a different updating strategy for the main parameter of the algorithm, were employed on the same population. The proposed models were bench-marked on a set of well-known UCI datasets.

To assess the efficiency of the proposed algorithms, two experimental approaches were conducted. The first experiment included comparing the proposed algorithms (independent, cooperative) with the original algorithm. In this experiment, the independent homogeneous parallel version of the GWO showed a good performance when compared with sequential one, and the independent heterogeneous GWO outperformed both the sequential and homogeneous versions of the GWO. Moreover, the cooperative heterogeneous algorithm outperformed all previous algorithms in terms of accuracy, however, it suffers from long execution time. The second experiment included comparing the proposed algorithms with selected machine learning techniques (e.g CART) in term of accuracy. Telecom company dataset from Kaggle data repository was used in this experiment to evaluate the proposed approach in order to predict the possible churner customers. The results showed the superiority of the heterogeneous cooperative algorithm.

الملخص

اكتسبت خوارزميات الأدلة العليا إهتمام العديد من الباحثين في مجالات التحسين المختلفة. إختيار الميزة هي مشكلة تحسين اندماجي حيث تستخدم خوارزميات أدلة عليا مختلفة لمعالجتها. خوارزمية الذئب الرمادي هي خوارزمية حديثة من خوارزميات الادلة العليا القائمة على اساس تعدد الحلول وتتميز باداء جيد في معالجة مشاكل التحسين المختلفة، بما في ذلك مشكلة إختيار الميزة. وكغيرها من خوارزميات التحسين الشاملة الاخرى، فإن خوارزمية الذئب الرمادي تعاني من بعض المشاكل التي قد تؤثر على ادائها، ومن الامثلة على تلك المشاكل: التكدس بجوار الحلول المؤهلة محليا، ومشكلة التنوع في الحلول.

قدمت هذه الاطروحة خوارزمية جديدة لمحسن الذئب الرمادي بشكل متوازي بهدف الحفاظ على تنوع معقول من الحلول، بالإضافة إلى مساعدة الخوارزمية على الهروب من الحلول المحلية. تم اقتراح نموذجين متوازيين: الأول متجانس، حيث تم استخدام أربع نسخ من خوارزمية الذئب الرمادي على نفس الحل. بينما في النموذج الثاني الغير المتجانس، تم استخدام أربع نسخ من خوارزمية الذئب الرمادي، حيث تعمل كل خوارزمية على استراتيجية تحديث مختلفة على نفس الحل. تم قياس اداء النماذج المقترحة على مجموعة من بيانات (UIC) المعروفة.

لتقييم كفاءة الخوارزميات المقترحة ، تم إجراء نهجين تجريبيين. تضمنت التجربة الأولى مقارنة الخوارزميات المقترحة المستقلة والتعاونية مع الخوارزمية المتسلسلة. في هذه التجربة، أظهرت الخوارزمية المتوازية المتجانسة المستقلة من محسن الذئب الرمادي أداءً جيدًا عند مقارنتها بالنسخة المتسلسلة الاصلية، وتفوقتت الخوارزمية الغير المتجانسة المستقلة من محسن الذئب الرمادي على كل من الخوارزمية المتسلسلة والخوارزمية المتجانسة. علاوة على ذلك ، تفوقت الخوارزمية التعاونية الغير المتجانسة على جميع الخوارزميات السابقة من حيث الدقة، ومع ذلك ، فإنها تعاني من وقت تنفيذ طويل. تضمنت التجربة الثانية مقارنة الخوارزميات المقترحة مع تقنيات تعلم آلي مختارة مثل (CART) بهدف تقييم النهج المقترحة في دقة التنبؤ بإعراض العملاء المحتملين. وفي هذه التجربة تم استخدام بيانات لشركة اتصالات من مستودع بيانات (Kaggle). وأظهرت نتائج هذه التجربة تفوق الخوارزمية التعاونية غير المتجانسة.

Acknowledgements

I wish to dedicate this thesis to my beloved late father, who always had confidence in me and offered me encouragement and support in all my endeavors.

And, I would like to pay special thankfulness and appreciation to Dr. Majdi Mafarja for his vital support and assistance. His encouragement made it possible to achieve the goal of this research.

Finally, I must express my very profound gratitude to my mother, my wife Ayah, my children Tayseer & Mohammad, and my aunt Fatima for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them.

Contents

List of Abbreviations

ABC	Artificial Bee Colony
ALO	Ant Lion Optimizer
ANN	Artificial Neural Networks
BA	Bat Algorithm
BC	Bee Colony
BGWO	Binary Grey Wolf Optimizer
CA	Clustering Algorithm
CART	Classification And Regression Tree
CHAID	Chi-squared Automatic Interaction Detector
CPU	Central processing Unit
CRISP-DM	Cross-Industry Process for Data Mining Model
CRM	Customer-Relationship Management
CSF	Core Service Failure
CSV	Comma-Separated Values file
DE	Differential Evolution
DE	Dolphin Echolocation
DF	Dragon-Fly Algorithm
DiPS	Distributed Problem Solver platform
DM	Data Mining
DMA	Direct Memory Access
DR	Dimensionality Reduction

EA	Evolutionary Algorithms
EPD	Evolutionary Population Dynamics
FPGA	Field Programmable Gate Array
FPU	Floating Point Unit
FS	Feature Selection
F-test	Friedman Test
GA	Genetic Algorithm
GCA	Genetic Clustering Algorithm
GD-RCGA	Gradual Distributed Real-Coded GA
GOA	Grasshopper Optimization Algorithm
GPU	Graphical Processing Unit
GSA	Gravitational Search Algorithm
GWO	Grey Wolf Optimizer
HGA-NN	Hybrid Genetic Algorithm with a Neural Network
I/O	Input/Output
JSS	Job Shop Scheduling problem
KDD	Knowledge Discovery in Database
KNN	K-Nearest Neighbors algorithm
KNN-LR	K-Nearest Neighbor algorithm and Logistic Regression
MaxItr	Maximum Iterations
MIMD	Multiple Instruction streams - Multiple Data streams
MISD	Multiple Instruction streams - Single Data stream
MOS	Multiple Offspring Sampling
MPI	Message Passing Interface
NB	Naive Bayes classifier
NFL	No Free Lunch theorem
NUMA	Non-uniform Memory Access
OpenMP	Open Multi Processing Framework
pBGWO	Parallel Binary Grey Wolf Optimizer

pcMBGWO	Parallel cooperative Multi Offspring Samplings Binary Grey Wolf Optimizer
piMBGWO	Parallel Independent Multi Offspring Samplings Binary Grey Wolf Optimizer
P-metaheuristics	Population-based metaheuristics
PSO	Particle Swarm Optimization
P-value	Probability - value
RBF	Radial Basis Function
RF	Random Forest
RPC	Remote Procedure Call
SA	Simulated Annealing algorithm
SEF	Service Encounter Failure
SIMD	Single Instruction stream - Multiple Data streams
SISD	Single Instruction stream - Single Data stream
S-metaheuristics	Single-Based metaheuristic
SVM	Support Vector Machines
TSP	Traveling Salesman Problem
t-test	Student's Test
UMA	Uniform Memory Access
WOA	Whale Optimization Algorithm

List of Figures

List of Tables

Chapter 1

Introduction

The world has recently witnessed rapid development and growth in the telecommunications sector. Due to the existence of many service providers, the focus should be on maintaining existing customers rather than acquiring new ones [1]. From this perspective, predicting the customers that are welling to move to a competing operator (which is known as customer churn), becomes important for the survival of companies and reduce expenses. In this thesis, the focus will be on predicting the churner customers in the prepaid programs, who are characterized by non-compliance with the contract with the telecommunications companies.

1.1 Research Motivation

The customers database volume is increasing due to technology advancement in data storage and decreasing its cost as well. In this sense, the problem of data reduction arises. Feature selection (FS) is one solution to this problem, which is used to remove noisy, redundant and irrelevant data from database (dataset). Moreover, (FS) is important to reduce data size stored in database, as well as decrease over-fitting problem. Its worth mentioning that, Feature Selection (FS) is an optimization problem itself, means that we need optimization techniques such meta-heuristic algorithms to conduct (FS) as a solution, since the complete search is impossible due to enormous data size, and computational power needed.

There are varieties of metaheuristic algorithms, but there is no one algorithm capable to solve all problems (NFL) theorem. This motivated us to use Grey Wolf Optimizer (GWO)

with different update functions using MOS technique (section 5.3.1). Moreover, the sequential GWO algorithm is a time wasting in the huge datasets, which was also motivated us to write a parallel version of GWO algorithm using OpenMP framework to run on multi-processors (section 3.4.1) to speed up the algorithm.

1.2 Research Objectives

The main goal of this thesis is to propose two parallel versions of the GWO; namely homogeneous GWO and heterogeneous GWO as searching strategies in a wrapper (FS) method to enhance the prediction accuracy of a churn prediction model. To achieve this goal; the following research objectives can be formulated:

- To assess the influence of employing multiple copies of the GWO on the same population, in a parallel form, on the quality of the produced feature subset, rather than using the original GWO.
- To evaluate two different versions independent and cooperative algorithms, (i.e., homogeneous and heterogeneous) of the parallel GWO.

1.3 Research Questions

In the context of improving the accuracy of the churn prediction model by eliminating the irrelevant and/or redundant features from the original feature set using a parallel GWO algorithm, this thesis seeks to answer the following research questions:

- RQ 1: What is the influence of employing a (FS) method on the churn prediction model in telecom industry?
- RQ 2: How the parallel version of the (GWO) enhances its performance in selecting the most informative features that enhances the performance of the prediction model?
- RQ 3: How the performance of the (GWO) is affected by using different parallelizim approaches (i.e., heterogeneous or homogeneous)?

- RQ 4: What is the benefit of using advanced computational intelligence methods instead of the basic machine learning algorithms for predicting consumer churn?

1.4 Customer Churn In Telecommunication Industry

The volume of data has recently evolved due to advances in information technology. The characteristics of this data are raw and contain many valuable and hidden information. Customers churn prediction is one of the biggest challenges for telecom companies. As mentioned above, companies prefer to keep the current customers more than attracting new ones, because new customers mean an increase in expenses and new recruits of manpower [2]. Telecommunication companies use databases to store customer data, where these databases are used by churn models to detect customers who intend to leave the service provider. The model should disclose the reason why the customer left the company in addition it should be accurate.

Customer-relationship management (CRM) is one of the ways the company interacts with current or future customers, which is based on historical customer data analysis. Customer retention is one of the tasks of this management. Data mining techniques are widely used for customer churn prediction, which provide better prediction performance outweigh their traditional statistical methods [3]. In this context, companies are collecting and monitor customer behavior and storing data in databases, where knowledge discovery in database (KDD) techniques are applied to extract hidden information [4, 5, 6, 7]. The KDD process consists of: dataset selection, data preprocessing, data analysis, and result interpretation and evaluation.

Customers datasets are very large, and contains unrepresentative (noisy) data. Such datasets negatively affect the prediction quality, and require large memory, in addition to the slow execution speed. Here comes the importance of data preprocessing in KDD, which used to remove a specific number of noise from the data. On the other hand, dimensionality reduction (or Feature Selection) aims to select the most representative features over a given dataset.

In most of telecom companies, there are two types of customers: postpaid customers who are bound by a contract, and prepaid customers who are difficult to churn predict. It is worth mentioning that both prepaid and postpaid are opposed to each others. Prepaid customers, who are not obliged to pay monthly bills, and do not receive offers from the service provider (e.g free calling minutes, call discounts, etc). These customers must charge their call credit to make outgoing calls, once the call credit is used up, they have to recharge their credit again, but they can receive calls regardless the call credit. Most prepaid customers are classified as irregular users, who may not recharge their call credit for long periods, such calling behaviour that is very hard to predict.

Postpaid customers are obligated by contract with the service provider. After accepting the terms of the contract, postpaid customers will provide their personal information to the service provider (e.g name, address, gender, etc). In the opposite, prepaid customers may not give their personal information when purchasing Sim-cards from the service provider, also they may share this Sim-card between individuals without notifying the service provider.

1.4.1 Churning

Formally speaking, customer churn is defined as "the process of subscribers (either prepaid or post paid) switching from one service provider to the next competitor." [9]. The benefit that can be obtained by predicting consumer churn is reducing the probability of customers leaving and thus maximizing the profits of the company.

Customer churn behavior can adversely affect the company in general by: revenue losses, reflected negatively on long-term customers, reduce ratio of new customers due to uncertainty and put the reputation of the company at stake against the competitors in light of the loss of customers. Furthermore, the costs of attracting new customers are much higher than the costs of retaining existing customers. And could cause a low sales due to the small number of short-term customers who buy a service from the company. Finally, it helps to attract customers who are not satisfied with the service by competitors [63].

Customer churns can be categorized into three types. The first type is active churner (deliberate, volunteer); in this type the customers quit the contract and move to another

service provider. The second type is passive churner (non-Volunteer); where the customers disconnected by the service provider, usually due to non-payment for a long period of time. In many cases, the consumer is out of service (churned) before being stopped by the service provider on the basis of non-obligation to charge calls. In this case, deactivation date is not suitable for churn prediction. The third type is rotational churner (incidental, silent); where customers discontinue the contract without the prior knowledge of the service provider, which make this type hard to predict [10].

Different approaches has been widely used to build predictive model to discover customer churn patterns such: data mining techniques and statistical methods (e.g linear regression, logistic regression, etc). We used some of these methods for comparison reason with our proposed approach (section 6.3). In section 1.5, we present the most common data mining (DM) techniques.

1.5 Data Mining Techniques

The tremendous evolution of information technology in the past two decades has led to the existence of large storage media from databases and other repositories. Hence the need to use robust methods to analyze stored data to help companies extract hidden patterns and knowledge, which finally support companies to make the right decision [12].

Data mining is one of the most effective methods for recognizing reasonable models, rules and examples of information. Data mining consists of six stages: classification, estimation, prediction, group dependence, clustering and description [11]. Many data mining methods have been used to predict the consumer churn, and most common is decision tree, where the decision is made according to this method based on the tree branch.

CRISP-DM stands for a cross-industry process for data mining model. The CRISP-DM methodology provides a structured approach to planning a data mining project [13]. Figure 1.1 illustrates the six phases of CRISP-DM life cycle.

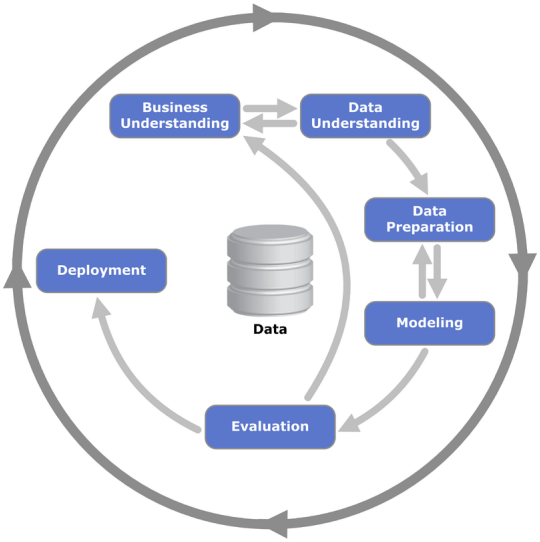


FIGURE 1.1: The phases of the CRISP data mining model[1].

CRISP-DM is an iterative process, that begins with understanding the nature of the data, this entails preprocessing data that may be incomplete (e.g missing values) or need conversion (e.g. categorical into numerical). The next step is building a model capable of extracting information that benefit companies in decision-making, it is worth mentioning that this model must achieve a degree of accuracy and performance. The final step is report generation, and deployment [13].

The following is a review of the most widely used data mining methods of predicting customer churn:

1. Naive Bayes Classifier:

   Naive Bayes is a supervised learning classifier based on Bayesian theorem. It calculates the probability that a given input sample belongs to a certain class [18], equation 1.1 presents the Bayesian formula for calculating probability.

$$P(c|x) = \frac{P(x|c).P(c)}{P(x)} \tag{1.1}$$

[1]https://www.datasciencecentral.com/profiles/blogs/crisp-dm-a-standard-methodology-to-ensure-a-good-outcome

2. K-nearest neighbors algorithm:

   K-nearest neighbors algorithm is a supervised machine learning algorithm that is used for both classification and regression problems. It calculates the distances between the testing sample and the samples in the training dataset based on specific metrics like: euclidean, chebyshev, cosine, etc. Then it sorts the calculated distances in ascending order and pick the first k neighbors. The final step is to predict the response based on neighbors voting, where each neighbor votes for its class feature, then take the majority vote as the prediction [22]. Although kNN algorithm is easy to implement, it takes considerable execution time based on dataset size. Algorithm 1 shows the pseudo code for brute force kNN algorithm.

Algorithm 1 Brute force kNN algorithm

Inputs:
    Training_dataset,
    Testing_instance,
    k
Outputs:
    predicted class
Initialize:
    List of empty neighbors with k size
neighbors = CalcDistance(Training_dataset, Testing_instance, k) *//First **k** neighbors (sorted ascendly).*
result = ClassVote(neighbors)
Return result

3. Support Vector Machines (SVM):

   The SVM is a discriminative classifier that be used for both classification and regression problems. It creates a hyper-plane with a maximum margin possible between two classes in the training dataset. SVM uses different types of kernels: linear, nonlinear, polynomial, Sigmoid, and radial basis function (RBF). SVM have been previously used for customer churn prediction [32].

4. Neural Networks:

   Neural networks are one of data mining techniques inspired by the nervous system in

the human brain, where the neural network learns from errors [14], this feature enabled it to be suitable for Artificial Neural Networks (ANN) [15].

Neural networks overcame the decision trees in a study conducted by Au et al [16], but it appears that neural networks are unable to detect hidden patterns in many obvious cases.

5. Decision Trees:

   The decision tree is the most common method for predicting customer churn by building a predictive model that able to predict class (label) from multiple input variables in the dataset. The decision tree is based on the idea of "divide and conquer" for binary tree construction. The tree consists of nodes that represent features (attributes) as inputs from the dataset, leaves which represent class(label), and branches that present routes between features and labels [19].

   The process of creating a decision tree starts with searching the attribute that contains the best information gain at the root of the tree, and then the tree is divided into partial trees (sub-trees) recursively. The partitioning process stops when if no information gain available, or reach the end of the tree node (leaf node). Figure 1.2 presents a simplified decision tree for customer churn prediction in telecom industry [18]. Finally the imbalanced class in the dataset is a critical issue that must be solved during a preprocessing phase. In term of customers churn, class imbalance happened when the ratio of passive churner is less than active churner, or vice versa [20].

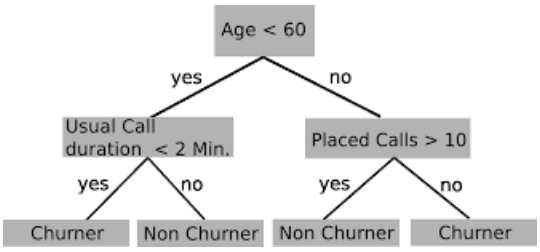


FIGURE 1.2: A simplified churn prediction decision tree.

There are several decision tree techniques such:

- CHAID: It is an acronym for (Chi-squared Automatic Interaction Detector). In this techniques, chi-square test has been used to figure out the best split at each step.
- CHAID variant: Is an improvement on traditional CHAID regarding accuracy, which applying intensive testing and merging of predictors. On the other side, Because of the large number of calculations, this method is considered slow.
- CART: It is an acronym for (Classification Regression Trees). It splits the features into non-overlapping regions. This technique is suitable for datasets that include continuous dependent variables and categorical predictor variables.
- QUEST: Is a statistical tree that uses ANNOVA F-Statistical tests to select feature, that used to split the tree. This technique is characterized by quick performance, efficiency and unbiased.

6. Random Forest (RF):

   A Random Forest is an ensemble learning method for regression and classification. It builds multitude trees randomly at training phase, and the output is the most popular class (mode) in case of classification, or roughly estimated responses (average) of the dependent variable in case of regression [25].

In this chapter, we presented a customer churn prediction in telecommunication industry and its definition; how customer churn can adversely affect the telcom companies. Moreover, we presented the most common data mining techniques used to solve customer churn problem, in addition we highlighted research questions, objectives and motivation.

Machine learning is considered one of the well known approaches to solve the customer churn prediction in telecom companies, which attracted several researchers to conduct studies in this field. In chapter 2, we will investigate the metaheuristic algorithms 2.1, metaheuristc for Feature Selection 2.2, parallel metaheuristic 2.3 and customer churn studies 2.4.

Chapter 2

Literature Review

In this section, we will provide a literature review on Metaheuristic 2.1, Metaheuristc for Feature Selection (2.2), Parallel Metaheuristic (2.3) and Customer Churn (2.4).

2.1 Metaheuristic

Metaheuristics are generic strategies to find approximate solutions. In practice, many optimisation problems (searching, machine learning, etc.) are NP-hard, where a lot of computing effort is required to solve them. There are many algorithms within this category such as grey wolf, Artificial Bee Colony (ABC), Bat Algorithm (BA), etc. Metaheuristic algorithms are characterized by many pros but suffer from some negatives as well. For instance, Grey Wolf Optimizer (GWO) counters bias towards the exploitation, on the other hand, Whale Optimization algorithm employs more exploration over the course of iterations.

Mirjalili et al. [45] proposed nature inspired metaheuristic optimization algorithm, that mimics the grey wolves social dominant hierarchy. The algorithm contains three phase: hunting, searching and encircling and attacking the prey. The proposed algorithm has been compared to several algorithms such GSA, PSO, DE, etc, and the results pointed that GWO performs was better. Emary et al. [64] proposed a binary version of grey wolf algorithm, to solve classification problems, which used to select optimal subset of features. They used special crossover (bGWO1) and Sigmoid function (bGWO2) to transform continuous values into binary.

Mirjalili et al. [49] presented another nature inspired metaheuristic algorithm, that mimics the whale hunting behavior. The proposed algorithm is called (WOA), and consists of three phases: searching, encircling and attacking the prey. The proposed algorithm shows remarkable results compared to other algorithms. Mafarja et al. [42] proposed a binary version of whale algorithm. The proposed algorithm consists of two approaches: (tournament roulette) and (evolutionary operators e.g. crossover). Comparing to other FS algorithms such GA, ALO, etc. The proposed algorithm shows better results.

Kirkpatrick et al. [70] proposed the simulated annealing algorithm (SA), which is a probabilistic technique to approximate the global minimum for a specific function with large number of variables. The algorithm is based on metallurgy annealing. Where the material is heated and cooled to reduce its crystals defects. The algorithm has the ability to avoid local minima. But there is a trade-off between the quality of the solution and the computation time.

Nakamura et. al [77] proposed Bat Algorithm (BA) which simulates the bats behavior.A binary version of this approach has been used in the classification. Where classification accuracy has been maximized by taking advantage of bat exploration with the optimum path forest classifier using wrapper model. Features are manipulated using bits string (0: not selected, 1: selected). Finally, the authors showed that the algorithm outperformed the algorithms in three out of five datasets.

Karaboga [76], proposed Artificial Bee Colony (ABC) algorithm, which is nature-inspired optimizer that simulates the swarm behavior of honey bees. The algorithm has an exploitation mechanism that allows it to be efficient in search. In addition, the algorithm is easy to apply in many problems. Nevertheless, the algorithm takes too much time in onlooker bee processing phase.

Kaveh and Farhoudi [52], presented Dolphin Echolocation (DE) approach, which simulates the hunting and navigation activities of dolphins in nature. The convergence factor has been readdressed by authors to reach better optimization than used in [78]. The authors claim that, (DE) have high convergence rates with very good results.

The Genetic algorithm (GA) was first introduced by John Holland [71]. The algorithm is

inspired by Darwin's theory of gene evolution. GA starts with random initial solution, which evolve over the course of iteration and guided by objective function to reach near optimal solution. GA encodes the search space parameters by strings (chromosomes), and a group of chromosomes is called a population. In addition, GA apply's crossover and mutation operations over the chromosomes to generate new chromosomes that have better features [72, 73, 74, 75].

To achieve a balance between exploration and exploitation, a hybrid combination between different optimization algorithms is used. For instance, Whale optimization algorithm is hybridized with simulated annealing. Mafarja and Mirjalili [41] conducted a hybrid whale optimization algorithm with simulated annealing for Feature Selection (FS). In this case, WOA runs first, and SA comes next to improve the exploitation, which aims to get better solution. Furthermore, a hybrid between (WOA) and (simulated annealing) has been proposed in a wrapper model. Both low level teamwork hybrid and high level rely hybrid were used In the low level.

Another hybrid model proposed by [79], which combine differential evolution (DE) and artificial bee colony (ABC) for Feature Selection (FS). This hybridization includes: new binary mutation phase, modifying the onlooker bee process to avoid local optima, and taking advantage of (DF) convergence nature (fast nature).

Mafarja et al. [39] proposed a hybrid meta-heuristic algorithm which combines Grasshopper optimization Algorithm (GOA), Evolutionary Population Dynamics (EPD) algorithm, and selection operator. Where algorithms work concurrently to increase efficiency and to avoid local optimum of (GOA). The proposed algorithm aims to direct the search toward promising areas by removing poor solution. The authors claim the superiority of this approach compared to other Feature Selection mentioned in the literature.

2.2 Metaheuristc for Feature Selection

Feature Selection (FS) is considered a dimensionality reduction (DR) technique used to remove redundant features in dataset. Moreover, FS is an NP-hard problem regarding searching

for the most informative features, where traditional algorithms can not be applied, instead meta-heuristic algorithms are suitably used by approximation.

Stjepan et al. [80] proposed a hybrid genetic algorithm with a neural network (HGA-NN). The proposed approach works in wrapper model, which enhances results by removing not important features in the dataset, and enhancing the initial population by incremental genetic algorithm stages. The algorithm applied to solve credit risk assessment system problem. The authors claim that this approach has been a good addition to data mining techniques.

Authors in [81] proposed a hybrid algorithm which combines Particle swarm optimization (PSO) algorithm with local search strategy to solve (FS) problems. In this model, the local search was included in the PSO to guide the search process to select features through correlation information. The statistical tests has been conducted, and the results showed the superiority of proposed approach in term of accuracy.

Antlion optimization (ALO) is another nature-inspired meta-heuristic algorithm used in FS. Antlion mimics the hunting behavior for antlions in nature. Emary et al. [82] proposed a binary version of the antlion in wrapper model by using transfer functions (v-shaped, s-shaped or basic operators). The authors claim that their approach have the ability to produce optimal solutions regardless of the initial population. Zawbaa et al. [83], proposed another wrapper FS based on (ALO), where the classification accuracy used as objective function.

Nguyen et al. [84] presented a hybrid model based on PSO for FS. Where a local search mechanism works near FS method. The algorithm takes advantage of both wrapper and filter methods. Where classification error is used as fitness function (wrapper), and filter as a measure. The authors showed that, the proposed approach have better performance.

2.3 Parallel Metaheuristic

The use of independent parallelism model in S-metaheuristics and P-metaheuristics algorithms is not new. A famous example for S-metaheuristics is the multistart local search, which is a model that use local search algorithms that works concurrently to find near optimal solution (e.g simulated annealing, local search, tabu search and others) that have its

own independent initial solutions, and possibly different parameters like size of tabu [85, 87]. Several algorithms can be used together (heterogeneous algorithms) to find the solution in the parallel model. This type is called the cooperative Model, where algorithms exchange information among themselves to reach a solution.

2.3.1 Parameter Level

Kennedy [88] applied k-means clustering on PSO to improve the performance using parallel model. Where k-means clustering is used to divide swarm into clusters of particles, and each particle represents a segment in the search space and works in parallel.

Tongchim and Chongstitvatana [89] proposed an adaptive mechanism for parameter adaptation in parallel genetic algorithm. The algorithm aims to adjust parameters during the the search process. Where the populations are divided into smaller sub-populations. The sub-populations are evolve in parallel and independently. After a specific time, some sub-populations exchange information at the a migration process in order to find the solution. The proposed algorithm adjusted four parameters: crossover rate, mutation rate, crossover operator, and mutation operator. The authors claim that, the proposed algorithm outperforms the other algorithms in term of performance under benchmark problems.

2.3.2 Search Level

Alba et al. [90] presented the gradual distributed real-coded GA (GD-RCGA). The proposed model runs eight populations simultaneously in a cubic topology which includes sparse individuals. The topology consists two faces to reflect exploitation and exploration. In this papers, two local area networks has been used to experiments the proposed algorithm. The authors found that this model is more effective and gives better results with continuous optimization. In addition, they found that the asynchronous parallelization outperforms the synchronous parallelization.

2.3.3 Algorithm Level

Ram et al. [91] proposed two distributed algorithms for simulated annealing: clustering algorithm (CA) and genetic clustering algorithm (GCA). The algorithms were applied to solve both traveling salesman problem (TSP) and job shop scheduling problem (JSS). The (CA) uses a cluster of nodes, where each node runs a simulated annealing algorithm. The node exchanging information between each other in order to have a good convergence. The (GCA) starts with genetic algorithm to generate initial population, which is used by distributed simulated annealing nodes over the network. It worth mentioning that, both algorithms were implemented using Distributed Problem Solver platform (DiPS), with 18 network node that run Sun workstations. The authors showed that, the proposed algorithms have very good solutions in term of execution time and solution quality.

2.4 Customer Churn

In this section, we will present the researches conducted to predict customer churn using different methodologies and approaches. Table 2.1 illustrates the information about the prediction approaches used for customer churn and the important results sorted by research year (chronological order).

According to experiment conducted by Qureshi et al. logistic regression achieved good results with 78% accuracy of total number of active churners [15]. In other research, conducted by Nie et al. regression has been combined with decision trees to create a predictive model for telecom customer churn [28], they found that, the graded regression is better than decision trees. Nath and Behara applied Bayesian model to predict customer churn in a wireless company. The result of this research was only 68% of accuracy [29].

Zhang et al. conducted a research based on combination of both $k$nearest neighbor algorithm and logistic regression method as a hybrid system to build binary predictive model. They named the proposed algorithm KNN-LR. Furthermore, they compared the proposed algorithm with the following algorithms: original logistic regression, radial basis function

(RBF) network and C4.5 decision tree. The result of the research was the superiority of the proposed algorithm [30].

Huang Kechadi proposed a novel model. The proposed model combines modified k-means clustering algorithm with a classic rule inductive technique (FOIL). The results of the study showed that the proposed system was outperforming the comparison techniques that were used such: original k-means, decision tree, logistic regression, PART, etc [31].

Sharma and Panigrahi [17] conducted a search for cellular wireless services. They proposed a neural network-based approach for customer churn prediction. Their experiment achieved (92%) of accuracy on dataset obtained from UCI repository.

Decision tree has been carried out to solve customer churn problems. Jahormi et al. developed a predictive model for customer churn in pre-paid mobile telephony organizations using C5.0 decision trees technique with neural network. Their finding was decision trees is better than neural networks regarding performance [21].

Kaur et al. conducted a bank customer churn experiment, to discover the significant features of customers. In their study, they used Naive Bayes, support vector machines (SVM), and J48 decision tree. They concluded that success prediction of churn class is larger than loyal class success prediction [22]. In another study conducted by Soein and Rodpysh. They found that CART decision tree performed better than other techniques (C5.0, QUEST, CHAID, Bayesian networks and Neural networks) [23]. Moreover, a hybrid evolutionary approach has been conducted by Yeshwanth to churn prediction in mobile networks. This approach combines Genetic algorithm with J48 decision. The result of this study showed that (72%) accuracy when applied to a large telecom company [24].

Web Chin-Ping Wei and I-Tang Chiu [65] used the decision tree approach C4.5 for customer retention analysis. Keaveney [66] conducted a survey to discover churn reasons. The study indicates that: Core Service Failure (CSF) or Service Encounter Failure (SEF) will cause the customer to switch the service into another companies. B. Padmanabhan et al. [67] consider that customer response and service quality are the drivers of customer churn. Bloemer et al. [68] assumed a direct relationship between customer satisfaction and the quality of service. Finally, There are several methods to predict customer churn, including: Naive

Bayes and Bayesian network [6] which improved the prediction rates over the C4.5 decision tree, regression analysis [69], neural networks [4, 10], decision trees [6, 69], and support vector machine (SVM) [32].

From the previously studied works. Customers churn prediction is one of the biggest challenges for telecom companies. Telecom companies suffers from the high dimensional dataset due to advances in storage technologies. Moreover, the customers dataset contains unrepresentative (noisy) data, which negatively affect the prediction quality, and require large memory, in addition to the slow execution speed. Data mining is a promising solution used to build predictive model to discover customer churn.

Dimensionality reduction is the key solution to build an accurate customer churn model to avoid problems such as model over fitting. Feature Selection (FS) is used to achieve data reduction by choosing the important features. It is worth mentioning that, the Feature Selection (FS) is an optimization problem. Optimization problems are complex and require intensive resources. In this regards metaheuristic algorithms found to be the natural fit to solve such problems with near optimal solutions. Exploration and exploitation are the core design aspects in metaheuristic algorithms. Exploration is the process to find new promising solutions, on the contrary, exploitation is the process of improving the current promising solution.

Grey Wolf Optimizer (GWO) is a metaheuristic algorithm that mimic the social dominant hierarchy and hunting mechanism of the wolf. Grey Wolf Optimizer (GWO) is the most cited algorithm with more than 2000 citation indexes based on Science Direct website, which motivated us to adopt this algorithm in this thesis.

Finally, the sequential metaheuristics algorithms are considered inappropriate in terms of executing time. Therefore, the use of parallel metaheuristics algorithms is preferred to avoid this problem. In this regard, one can observe that the parallel form of the metaheuristics algorithms with feature selection (FS) has been rarely investigated through the literature reviews to solve the customer churn prediction.

Author	Year	Approach	Obtained Results
Qureshi et al.	2013	Logistic regression	78% accuracy.
Huang & Kechadi	2013	Modified K-means clustering algorithm with classic rule inductive technique (FOIL)	Better than original K-means, decision tree, logistic regression, PART, etc.
Kaur et al.	2013	Naive Bayes, support vector machines (SVM), and J48 decision tree	Success prediction of churn class is larger than loyal class success prediction.
Clement Kirui et al.	2013	Probabilistic Classifiers	Improved prediction rates for all the models used.
I. Brandusoiu & G. Toderean,	2013	Support Vector Machines	The model that uses the polynomial kernel function performs best with 88.56% accuracy. The models RBF, LIN, and POL have 80% accuracy.
Soein & Rodpysh	2012	C5.0, QUEST, CART, CHAID, Bayesian Networks and Neural networks	CART decision tree performed better than other techniques.
Nie et al.	2011	Logistic regression with decision trees	The test result graded regression ahead of decision trees.
Sharma & Panigrahi	2011	Neural Network	92% accuracy.
Yeshwanth	2011	Genetic algorithm with J48 decision	72% accuracy.
B. Padmanabhan et al.	2011	Churn study	Service quality and customer response are two important drivers of churn.
Jahormi et al.	2010	C5.0 decision tree with Neural Network	Decision trees is better than Neural Network.
Zhang et al.	2007	K-nearest neighbor with Logistic Regression	Better than original Logistic Regression, radial basis function and C4.5 decision tree.
Nath & Behara	2003	Bayesian model	68% accuracy.
Web Chin-Ping Wei & I-Tang Chiu	2002	C4.5 decision tree	Used on customer retention analysis and prediction.
Bloemer et al.	1998	Churn study	Assumed a direct relationship between customer satisfaction and the quality of service.
Keaveney	1995	Survey to discover churn reasons	Core Service Failure or Service Encounter Failure will cause the customer to switch the service into another companies.

TABLE 2.1: Literature review summary: prediction approaches used for customer churn.

Chapter 3

Background

In this chapter, a theoretical background about Feature Selection (FS) methods is presented (3.1), followed by a description about the metaheuristics approaches in general (3.2), and illustrating in depth the original Grey Wolf Optimizer and its binary (3.3). Finally presenting the development frameworks that used to write the the proposed algorithms.

3.1 Feature Selection

It is important to choose the right features before creating customer churn prediction model (dimensionality reduction). Large features set may include irrelevant attributes in customer churn. Therefore, increase model complexity, prediction slowness, low accuracy ratio, and over-fitting problem, which make the model dedicated to specific dataset. The features consist mainly of billing data, call statistics, demographic data and others [33, 34].

The Feature Selection (FS) objective is searching for the minimum attributes that meet a certain criterion to build a prediction model that achieves the highest degree of accuracy [35]. It is worth mentioning that Feature Selection is considered an NP-Hard problem [37].

The Feature Selection process consists of four phases [38], as shown in Figure 3.1:

- Feature subset generation: At this stage a partial set of attributes is searched using one of the following search methods: complete, random or heuristic [39]. The process of creation subset may begin with empty set without variables and add them one by one

until decrease the error (forward), full set with all variables and remove them one by one till highest accuracy is reached (backward), or random set to achieve the highest accuracy.

- Evaluation functions: The generated subsets in the previous phases are evaluated by using filter and wrapper models [36]. The evaluation functions measure the quality of generated subset.

- Stopping Criterion: Stop the Feature Selection process if the criterion is met, or prevent complete search.

- Validation: This phase is not considered part of the Feature Selection process, but is used to validate the accuracy of the attributes. It is an iterative procedure to create a classifier from the training data and validate its accuracy, using testing data, till gain the highest accuracy.

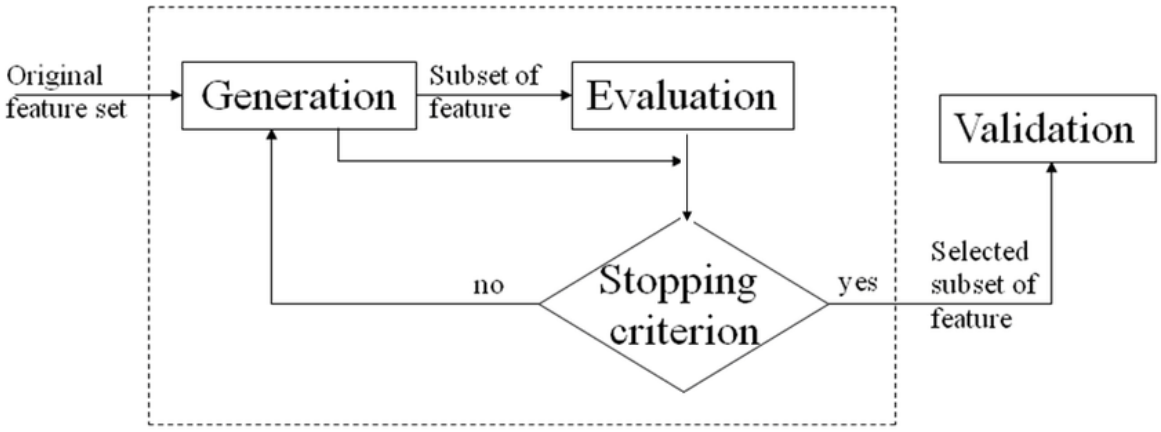


FIGURE 3.1: Feature selection process [38].

3.1.1 Feature subset generation

Feature subset generation, is the process of searching the best subset of features. This process is considered complex. Theoretically speaking, a Feature Selection method must search all possible combinations to find optimal or near-optimal features, on other words,

Feature Selection must search 2^N features, so it grows exponentially when N increases, where 2 is the binary representation of feature (0: no selected, 1:selected), and N is the total number of features [40].

Complete Search

As mentioned before, the complete search searches all possible combinations of features 2^N until reach the optimal subset that achieves the highest accuracy. So this method guarantees the optimal solution, but needs tremendous computational power and long searching process [41] which make this method infeasible for huge datasets.

Random Search

In this type of search method, features are searched randomly, which is also called non-deterministic search [40]. Because of the random nature. The solution (optimal features) could be found during the early stages of the search process which represent the best case. But it may need to visit the whole features just like complete search which represents the worst case.

Heuristic Search

Heuristic search is a technique used frequently when traditional methods are taking too much time to find the solution, or to find near-optimal solution by approximation when traditional methods fail. Heuristic methods use the minimum information available to execute the effective search without the need for all features combinations. There are two types of heuristic methods: general purpose meta-heuristics to solve a wide range of problems, and specified heuristics to solve specific types of problems [40, 42].

Meta-heuristics algorithms proved to be effective in solving optimization problems compared with other approaches like complete search. There are a many meta-heuristics algorithms, including: genetic algorithm [43], gravitational local search [44] and Grey Wolf Optimizer (GWO) algorithm [45], and others.

3.1.2 Feature evaluation functions

Feature evaluation phase comes right after the feature subset generation. In this phase, evaluation functions will be used to measure the goodness of the generated subset. Additionally, the evaluation function result will be compared with previous results, and the best result will be kept, over the course of the iteration. It should be noted that the use of different evaluation functions in the same dataset will generate different results [38].

As mentioned earlier, evaluation functions are categorized into two approaches: filter and wrapper. In filter approach, selected subset is evaluated based on the data regardless of learning algorithm, it uses statistical methods like distance in the evaluation process, unlike the wrapper approach which use a classifier to evaluate a features subset [42].

Filter Approach

Filter approach is highly depends on training data that generated from feature generation phase, instead of using learning algorithms. Moreover, the filter approach does not correlate between features. This approach consists of two phases as shown in Figure 3.2. The first phase uses statistical methods like distance, t-test, F-test, etc. to measure the goodness of the features to be selected. The second phase, which is similar to wrapper approach. Finally, the filter approach is considered fast; because it have non-iterative computation on the dataset. On the other hand, filter fitness functions are often monotonic, so the filter tends to select the full features as the optimal solution [40].

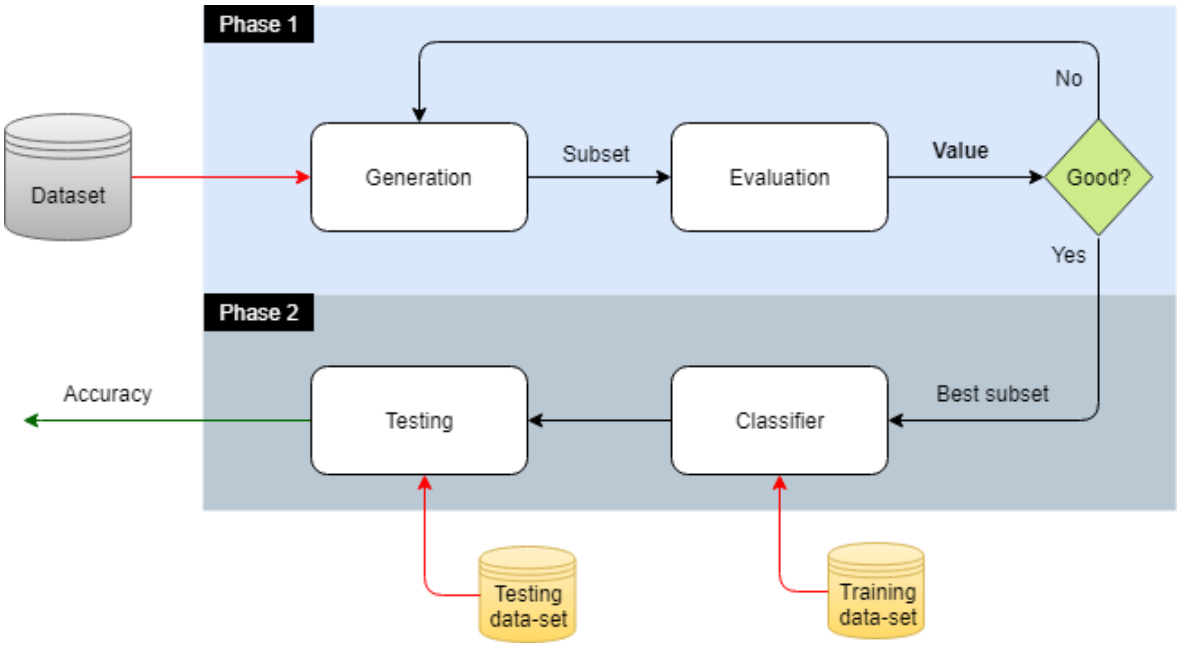


FIGURE 3.2: A filter approach of Feature Selection.

Wrapper Approach

The wrapper approach uses the classifier accuracy to measure the performance of selected subset. This approach aims to maximize predictive accuracy by reducing the error rate. As shown in Figure 3.3, wrapper model consists of two phases: Feature Selection subset phase and learning / testing phase. The first phase is an iterative process that generates subsets, where the accuracy of each subset is calculated. At the end of the process, the highest subset will be selected. In the second phase, the generated subset with highest accuracy that obtained in previous phase will be used in the learning algorithm over the training data, then the result will be compared to a testing dataset for accuracy measurement [40].

Finally, the wrapper approach achieves high degrees of accuracy compared to filter approach; because they are adjusted to specific interactions between the dataset and the classifier. Wrappers use use cross-validation technique to avoid over-fitting problem. But wrappers are slow; since a classifier must be trained for each feature. In addition, wrappers are tied to classifier bias, which make the solution lack of generality.

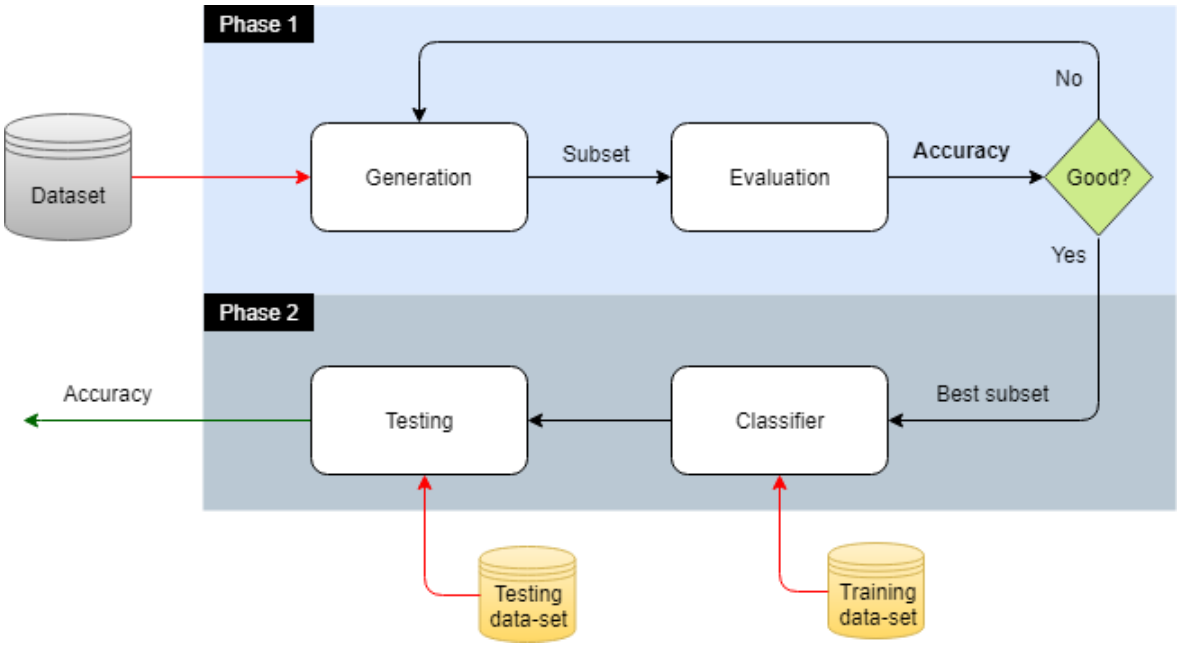


FIGURE 3.3: A wrapper approach of Feature Selection.

3.1.3 Feature stopping Criterion

Feature stopping criterion is a necessary step, in order to prevent the algorithm from entering an infinite loop; which may reduce computer resources, especially main memory, and causes the algorithm crash. The algorithm stops searching in general according to occur one of the following conditions [40]:

- Reach the loop limit of the algorithm (maximum iteration).
- The search execution time is finished, in case using time instead of iterations.
- Reach the complete search.
- Obtain an acceptable degree of feature subset quality (accuracy).

It is worth mentioning that, the stopping condition is also important to prevent optimization algorithms (e.g GWO) from reaching the minimum features that represent the highest accuracy values. In the case of a customer churn dataset in telecom companies, it is possible to reduce the dataset from tens of features into couple of features, which cause to create unrealistic and wrong predictive model.

3.2 Metaheuristics

Optimization methods are found in different subjects, including: engineering, computing and even everyday life, where maximizing or minimizing are being used to solve problems. For example, companies use optimization methods to maximize their sales while minimizing losses [47]. In a practical, it is known that finding the optimal solution or complete search is considered an expensive process and time consuming. Accepting a relatively reasonable and not optimal solution is therefore the way out of the impasse, and here comes the metaheuristic algorithms ways to do this work [46]. It is worth mentioning that the study of optimization problems is in fact very old and goes back to the Greek era [47]. The word ”metaheuristic” is originally Greek. It consists of two parts: the first part is ”meta” which means ”upper level methodology”, and the second one is ”heuristic” which means exploring new ways (strategies) to solve problems; The first use of this term was by F.Glover in 1986 [46].

Metaheuristic search methods can be defined as ”upper level general methodologies (templates) that can be used as guiding strategies in designing underlying heuristics to solve specific optimization problems” [46]. Another definition for meta-heuristic was mentioned in the literature as follows: ”is a group of techniques that guides the search process. The main objective of this method is to effectively explore the search space to find the best solutions” [48].

Based on the literature, there are many optimization algorithms such as: Grey Wolf Optimizer (GWO) [45], whale optimization algorithm [49], ant colony optimization [50], and others.

Exploration and exploitation are the core design aspects in metaheuristic algorithms. Exploration is the process to find new promising solutions which has not been inferred yet, the high exploration leads to fast convergence, but reduce the quality of results. On the contrary, exploitation is the process of improving the current promising solution, the high exploitation leads to local optima problem; which is the best solution within neighbors. To overcome this problem, a hybrid combination between different optimization algorithms is used [41].

The following sub-sections show two types of metaheuristic algorithms, namely single-based and population-based. Single-based metaheuristics improves one solution over the course of the iteration. Whereas, population-based improves a set of solutions over the course of iterations to find near optimal solution [46].

3.2.1 Single-Based Meta-heuristic Algorithms

Single-Based metaheuristic (S-metaheuristics) is an iterative procedure that ends with a given stopping criteria and consists of two steps as shown in figure 3.4. The first step is candidate solutions generation from the current solution; usually through transformations. The second step is the replacement, where a solution is chosen from the set of candidate solutions and replace the current solution [46].

S-metaheuristics can be used without the use of memory, so above procedures relies completely on the current solution. In contrast, memory can be utilized to store the search history. Finally, S-metaheuristics algorithms are exploitation oriented, making them vulnerable to local optima. A famous example of S-metaheuristics is simulated annealing [41].

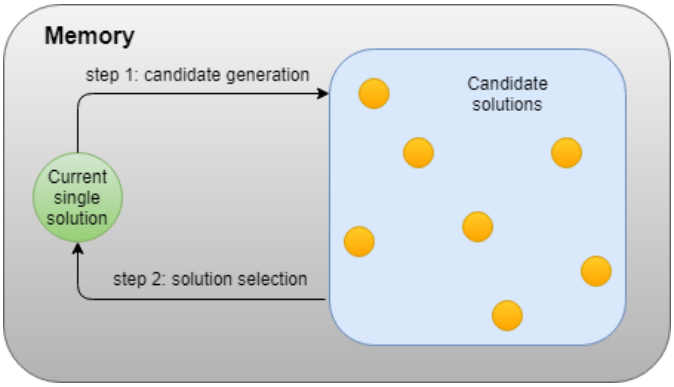


FIGURE 3.4: Main principle of the single-based solution algorithm.

3.2.2 Population-Based Meta-heuristic Algorithms

Population-based metaheuristics (P-metaheuristics) can be seen as a general state of (S-metaheuristics), it is also an iterative procedure that stops when a given condition is satisfied. P-metaheuristics consists of two phases: the generation and the replacement. This procedure

begins from an initial population of solutions, then generate a new population and finally, replace the current solution iteratively [46], as shown in figure 3.5.

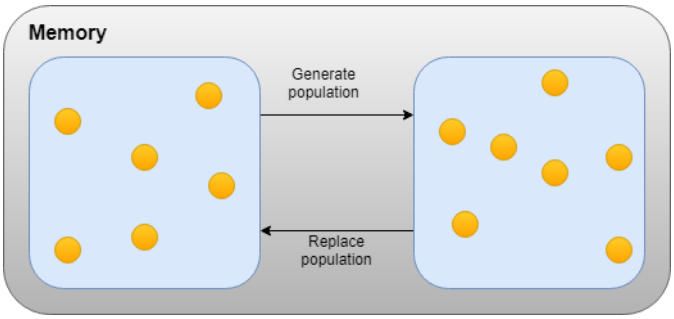


FIGURE 3.5: Main principle of the population-based solution algorithm.

In practice, it is important to diversify the initial population to avoid premature convergence problem. Following are strategies for generating new population:

- Evolution-based: this strategy uses different operators like mutation and crossover on the current population in order to generate a new one.
- Blackboard-based: this strategy utilizes the solutions available in the current population to generate new one.

In addition, population replacement includes two strategies:

- Replaces the population with a new generated one.
- Chooses the best solution from both the old and new generated populations to find the subsequent solution.

The stopping criteria in the population-based algorithm is either static or adaptive. In the static, the stopping criteria is known and predefined in the algorithm. In the adaptive, the stopping occurs based on the state of the algorithm [46].

P-metaheuristics may be memoryless, in this case, both generation and replacement phases depend on the current population. Memory can also be used, where searching history are stored in memory and used to create new populations (solutions) [46]. Algorithm 2 illustrates P-metaheuristics pseudocode. Furthermore, population-based algorithms is exploration oriented.

Algorithm 2 P-metaheuristics pseudocode

1: **Initialize:**
   $P = P_0$; //Generation of the initial population
   $t = 0$;
2: **Outputs:**
   Best solution(s) found.
3: **repeat**
4: Generate(P'_t); // Generation a new population
5: P_{t+1} = Select-Population($P_t \cup P'_t$) // Select new population
6: **until** Stopping criteria satisfied.

The majority of P-metaheuristics algorithms are nature-inspired. There are many nature-inspired P-metaheuristics such as evolutionary algorithms (EAs), particle swarm optimization (PSO) and bee colony (BC) [46].

3.3 Swarm Intelligence Algorithms

Swarm intelligence algorithms are inspired by the social behavior of species that compete for foods such as ants, bees, fish, birds and wolves [46].

In terms of Feature Selection (FS), many algorithms have been applied in this field, but the meta-heuristic algorithms have proven to be the best. Examples of these algorithms are particle swarm optimization [51], dolphin algorithm [52], and gery wolf optimizer [45]. The following subsections describe Grey Wolf Optimizer (GWO) (3.3.1) in the details, and its binary version (3.3.2).

3.3.1 Gery Wolf Optimizer (GWO)

Grey wolves live in a pack with group size between (5 - 12) on average. They live in social dominant hierarchy as shown in Figure (3.6).

The pack of Grey wolf consists the following according to [45]:

1. The alpha wolves are leading the pack, making decisions, and their decisions are dictated to the pack.

2. The betas are the second level of hierarchy that probably be a candidate to alpha, the betas are assisting the alpha in making decision.

3. The Omega are the lowest level in the hierarchy, they have to submit to all upper levels.

4. The deltas have to Obeys alphas and betas, but they control the omega. This category of wolves contains: scouts, sentinels, hunters, caretakers, and elders (experienced wolves) who considered to be alpha or beta candidate.

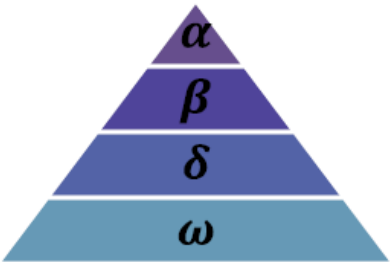


FIGURE 3.6: Grey wolf Hierarchy (dominance increasing from down top) [45].

The Grey wolf starts hunting by Tracking and chasing the prey, next the wolves pursuing and encircling the prey. Encircling behavior could be mathematically modeled using the following equations :(3.1), (3.2), (3.3), and (3.4). Finally the wolves attack towards the prey. The best solution based on Grey wolf mathematical model is: alpha (α), beta (β) and delta (δ) respectively; which they guide the hunting, and finally omega (ω) follow these three candidates.

$$\vec{X}(t+1) = \vec{X}_p(t) + \vec{A}.\vec{D} \tag{3.1}$$

Where: $\vec{D}$ is defined in equation (3.2), t is the iteration number, $\vec{A}$ and $\vec{C}$ are coefficient vectors, $\vec{X}_p$ is the prey position, and $\vec{X}$ is the gray wolf position.

$$\vec{D} = |\vec{C}.\vec{X}_p(t) - \vec{X}(t)| \tag{3.2}$$

A 2D position space and selected possible neighbors are illustrated in Figure (3.7)(a) that present the effects of equations (3.1) and (3.2), as shown in the figure the Grey wolf can

update its position (X, Y) randomly based on the position of the prey (X*, Y*); where $\vec{A}$ and $\vec{C}$ vectors are adjusted to reach any place around the best agent. On the other hand a 3D position space is illustrated in Figure (3.7)(b).

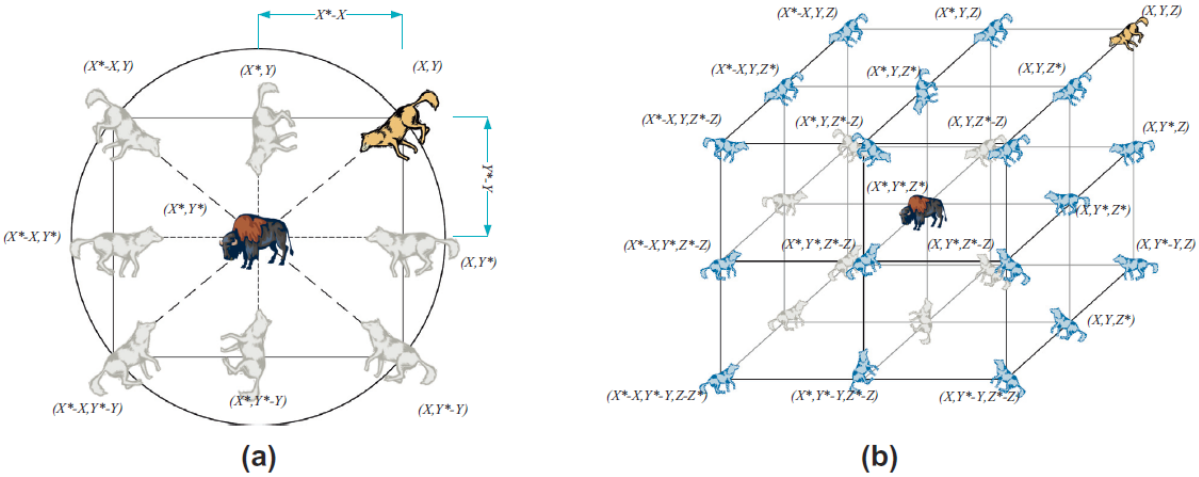


FIGURE 3.7: wolves position vectors in 2D and 3D [45].

The $\vec{A}$, $\vec{C}$ vectors are calculated as in equations (3.3) and (3.4).

$$\vec{A} = 2a.\vec{r}_1 - a \tag{3.3}$$

$$\vec{C} = 2\vec{r}_2 \tag{3.4}$$

where: (a) is linearly decreased from 2 to 0 over the course of iterations according to the equation (3.5), and r_1 , r_2 are random vectors in [0, 1].

$$a = 2 - t\frac{2}{MaxItr} \tag{3.5}$$

Where: (t) is the iteration number and MaxIter is the total number of iteration allowed for the optimization.

Equation (3.6) represents updating wolves positions:

$$\vec{X}_{t+1} = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3} \tag{3.6}$$

Where: $\vec{X}_1$, $\vec{X}_2$, $\vec{X}_3$ are defined as in equations (3.7), (3.8), and (3.9) respectively.

$$\vec{X}_1 = |\vec{X}_\alpha - \vec{A}_1.\vec{D}_\alpha| \tag{3.7}$$

$$\vec{X}_2 = |\vec{X}_\beta - \vec{A}_2.\vec{D}_\beta| \tag{3.8}$$

$$\vec{X}_3 = |\vec{X}_\delta - \vec{A}_3.\vec{D}_\delta| \tag{3.9}$$

Where: $\vec{X}_\alpha$, $\vec{X}_\beta$, $\vec{X}_\delta$ are the first three best solutions in the current iteration t, $\vec{A}_1$, $\vec{A}_2$, $\vec{A}_3$ are defined as in equation (3.3), and $\vec{D}_\alpha$, $\vec{D}_\beta$, $\vec{D}_\delta$ are defined using equations (3.10), (3.11), and (3.12) respectively.

$$\vec{D}_\alpha = |\vec{C}_1.\vec{X}_\alpha - \vec{X}| \tag{3.10}$$

$$\vec{D}_\beta = |\vec{C}_2.\vec{X}_\beta - \vec{X}| \tag{3.11}$$

$$\vec{D}_\delta = |\vec{C}_3.\vec{X}_\delta - \vec{X}| \tag{3.12}$$

Where: $\vec{C}_1$, $\vec{C}_2$, and $\vec{C}_3$ are defined as in equation (3.4).

The location of prey is estimated by alpha, beta, and delta as shown in figure (3.8). Alpha will get exploration when $|A| > 1$ and exploitation when $|A| < 1$. Finally, the algorithm is terminated by end of iterations, or by reach minimum or maximum solution. Algorithm (3) represents the pseudo code for GWO algorithm.

3.3.2 Binary Gery Wolf Optimizer (BGWO)

In the Feature Selection (FS) we use a binary number to either select the attribute (1) or not select (0), in this case we need to modify the original version of Grey wolf, which uses a continues values in the search space to restricted binary (0, 1).

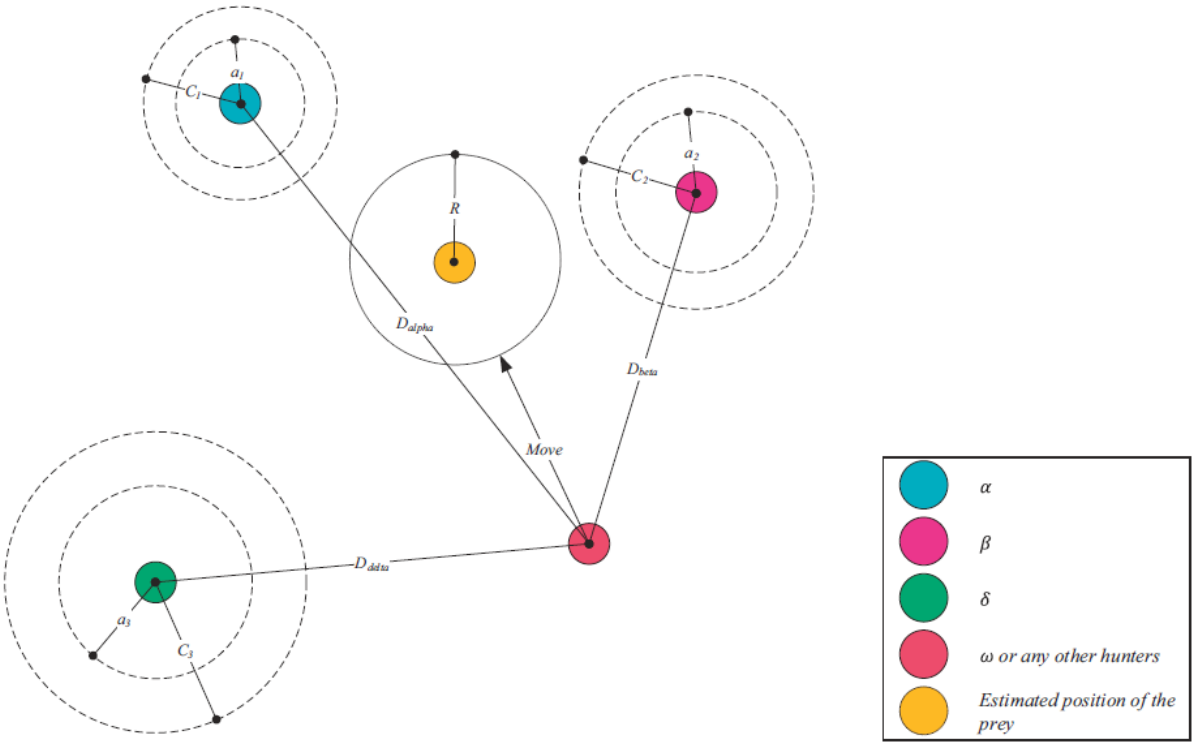


FIGURE 3.8: Position updating in GWO [45].

The binary version of the algorithm starts by initializing a two dimensional matrix that represents the positions of search agents (wolf position) filled with zeros and ones randomly. then we update the position of α, β and δ as the original Grey wolf algorithm as well as updating (a) parameter which is linearly decreased in interval [2,0]. The last step in the main course of the algorithm iteration is updating the wolves positions by using Sigmoid function as shown in equation 3.13, where $Sigmoid(x, a, c)$ is a mapping on a vector x, and depends on two parameters $a = 10.0$ and $c = 0.5$. Figure 3.9 illustrate Sigmoid function.

$$Sigmoid(x, a, c) = \frac{1}{1 + \exp^{-a(x-c)}} \tag{3.13}$$

Then the position of wolves α, β and δ are calculated based on 3.13 as shown in equations 3.14, 3.15 and 3.16 respectively.

$$Sigmoid(-A1 * D\_alpha, 10, 0.5) \tag{3.14}$$

Algorithm 3 Grey Wolf Optimizer (GWO) algorithm pseudo code

Initialize:
    GWO population $x_i(i = 1, ..., n)$
    n=0
Outputs:
    X_α //best solution
while n $<$ Max_iterations **do**
    for all agent $\in$ population **do**
        adjust agent boundaries
        calculate the objective function
        update X_α, X_β, X_δ
    end for
    calculates (a) parameter by equation(3.5)
    for all agent $\in$ population **do**
        calculate (A) parameter by equation(3.3)
        calculate (C) parameter by equation(3.4)
        update current agent position by equation (3.6)
    end for
    n = n + 1
end while

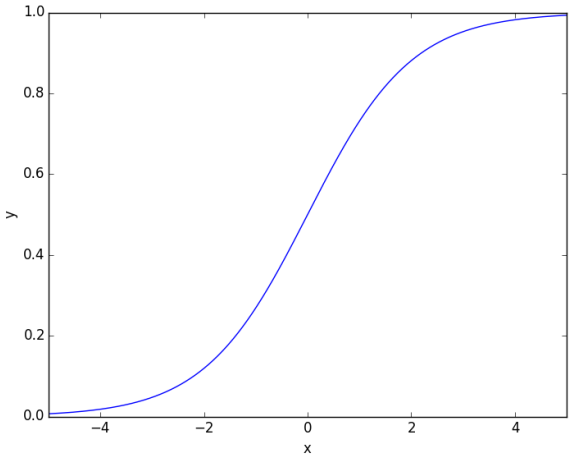


FIGURE 3.9: Sigmoid function.

$$Sigmoid(-A1 * D\_beta, 10, 0.5) \tag{3.15}$$

$$Sigmoid(-A1 * D\_delta, 10, 0.5) \tag{3.16}$$

It is worth to mention here that we have to make sure that wolves positions must be zero or one when updating wolf position as shown in algorithm 4.

Algorithm 4 Binary version of updating wolf position

```
s = Sigmoid(...) //as shown in equations 3.14, 3.15 and 3.16
if s <random then
    v = 0
else
    v = 1
end if
if wolf_current_position at iteration + v >1 then
    wolf_position = 1
else
    wolf_position = 0
end if
```

Since we do not have an idea about the location of the optimum solution, we will take a crossover operator between the three wolves instead of the average as the continues version of the algorithm as shown in function 5.

Algorithm 5 Binary BGWO cross over function

```
procedure CROSSOVER(x1, x2, x3)
    r = random
    if r <0.333 then
        return x1
    else
        if r <0.666 then
            return x2
        else
            return x3
        end if
    end if
end procedure
```

A KNN classifier is used as a wrapper approach in the fitness function to evaluate the solution in Feature Selection (FS) is a combination of minimum number of features and highest accuracy. So the fitness function will be a minimization problem as shown in equation 3.17.

$$fitness = (1 - SzW) * (1 - accuracy) + SzW * \frac{features\_sum}{dimension} \tag{3.17}$$

Where: SzW is a constant with a value 0.01, *accuracy* is the KNN accuracy which is, the ratio between the number of correct predictions over the total number of instances in testing training set, $features\_sum$ is a sum of ones in the current search agent, *dimension* is the size of search agent.

3.4 Development Frameworks

In this section we will present the development frameworks that used in development process of our proposed algorithms. Both OpenMP and Shark will be discuss in details at section 3.4.1 and section 3.4.2.

3.4.1 OpenMP

In this thesis, OpenMP framework version 2 is utilized to parallelize the sequential (BGWO). OpenMP is an acronym for Open Multi-Processing, its an open specification for multi processing API for defining multi threaded shared programs. This framework supports C/C++ and Fortran programming langaues. In addition, OpenMP consists of three main parts: runtime library, compiler directives (pragma) and environmental variables [53].

OpenMP is able to parallelize a sequential code, by adding simple directives. First, the code starts with one master thread, and then a number of threads are created (fork) in the parallel regain. Once the the parallel tasks are finished, all created threads will be destroyed (join), as shown in figure 3.10.

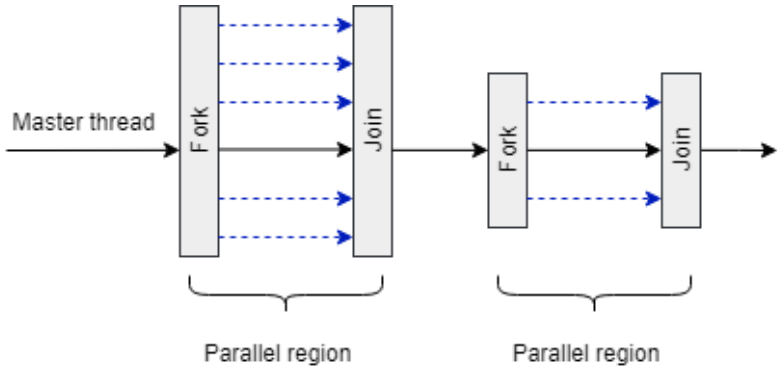


FIGURE 3.10: OpenMP execution model.

Habbas et al. pointed to the importance of using OpenMP for applications parallelization. Following are benefits of using OpenMP:

- No Message passing required, also it is not used in our thesis.
- OpenMP directives may be incorporated incrementally.
- Easy to transfer sequential code into parallel.
- Generates small code size (lightweight).
- It helps to write a parallel code that is easy to read.
- Portability, because of its standard, the code written using OpenMP can be ported to different platforms e.g. Windows, linus, MacOSX, etc.

On the other side, OpenMP has few drawbacks, which is not scalable as MPI, and dedicated to shared memory systems exclusively. Such drawbacks is not important in our thesis, since we are targeting Intel machine (shared memory system).

3.4.2 Shark

Shark is used to facilitate our parallel BGWO, in order to take advantage of its helpers functions, and algorithms. SHARK is a fast, modular, portable and feature-rich open-source C++ machine learning library [58].

The main purpose of using Shark is to speed up the programming process and save time. For example, instead of implementing: parallel KNN classifier, dataset reader, classifier evaluation, etc. Shark bundled with lots of such procedures. In addition, Shark will use available processors internally to speed up the performance. So, Shark is implicitly have an iteration-level parallel model, this seems obvious in calculating KNN distances. Figure 3.1, shows the list of Shark functions used in BGWO.

Function name	**Description**
importCSV	Import dataset from CSV based file
shuffle	shuffle the dataset
splitAtElement	split the dataset into training and testing
makeIndependent	copy the dataset into process memory
KNN	calculate K nearest knebiour
loss	calculate the data loss (error rate)

TABLE 3.1: Functions used from Shark framework.

Chapter 4

Parallel Metaheuristics

Optimization problems are complex, NP-hard, and require intensive resources (CPU time and/or memory consumption) in diverse domains of Real-life applications. Metaheuristics found to be the natural fit to solve such problems with near optimal solutions; to reduce the computational complexity of the search process [46].

On one hand, the search space is huge in complex optimization problems, which means more resources will be acquired by the fitness function of metaheuristics algorithms. Thus, the sequential metaheuristics algorithms are considered inappropriate in terms of executing time. Therefore, the use of parallel metaheuristics algorithms is preferred to avoid this problem [46].

On the other hand, the technological advancement in the manufacture of processors (e.g. multicore processors), networks (e.g. Infiniband) and storage media (e.g. solid-state drive), moreover, the cost/performance ratio is constantly decreasing. These factors played an important role to adapt parallel mechanisms in metaheuristics algorithms [46].

Following are the advantages of using parallel and distributed computing in meta-heuristics design [46]:

- Reduce the search time: optimization problems need big number of iterations to reach the solution, so using parallelizing in meta-heuristic will reduce the search time in exploration and exploitation by splitting the iterations among processors.

- Enhance solution quality: by passing messages between processes in parallel environment, this kind of communication will change meta-heuristics searching behavior in the landscape which usually leads to better convergence and reduced search.
- Improve the robustness: the strength of the parallel metaheuristic is its ability to solve multiple problems efficiently. Also in term of meta-heuristic parameter sensitivity.
- Solve large-scale problems: the ability of parallel meta-heuristic to solve a wide range of problems that cannot be solved by a sequential method. In addition, improving the accuracy (e.g. increasing the number of iterations will increase the chance to have good results).

Finally, the parallel meta-heuristic depends on target architecture, parallel programming language, middle-ware, and the employed techniques; such as shared memory (e.g. POSIX threads, OpenMP), remote procedural call (e.g. ONCRPC, protobuf/ZMQ).

4.1 Parallel and Distributed Architectures

Parallel and Distributed Architectures have many taxonomies based on: traditional Flynn classification, memory organization, processor granularity, processor synchronization, and interconnection architecture.

According to Flynn [46], parallel architectures are based on two criteria: the number of instructions, and the number of data. Following are the four classes for parallel architectures:

- SISD (single instruction stream-single data stream): it is the traditional type of computers, where a single processing unit (mono-processor) receives a single stream of instructions that operate on a single stream of data, as shown in figure 4.1. It must be noted that this type has been used less recently and is on its way to extinction.

FIGURE 4.1: SISD computers.

- MISD (multiple instruction streams-single data stream): a multiple instruction streams are executed on a single data stream. This type of parallel consists of N processors, each with its own control unit, share a common memory, as shown in figure 4.2. In practice, this type does not exist.

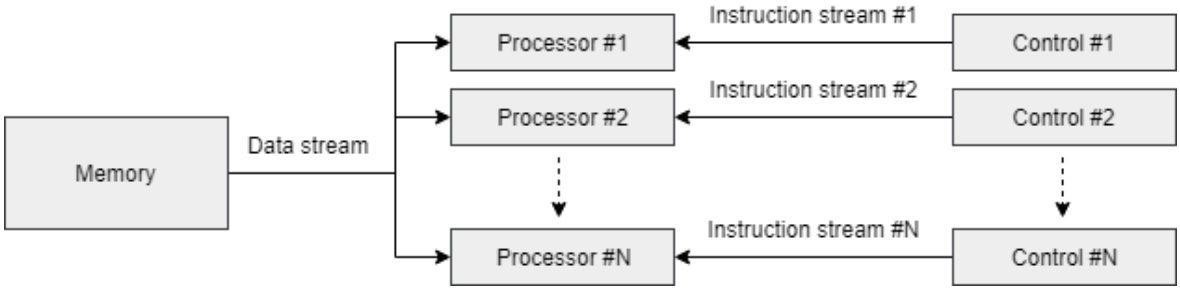


FIGURE 4.2: MISD computers.

- SIMD (single instruction stream-multiple data streams): in this type of parallel, where N processors operate under the control of a single instruction stream issued by a central control unit, and each processor has its own data streams, as shown in figure 4.3. SIMD is simple and scalable, but it is expensive and requires specific programming model.

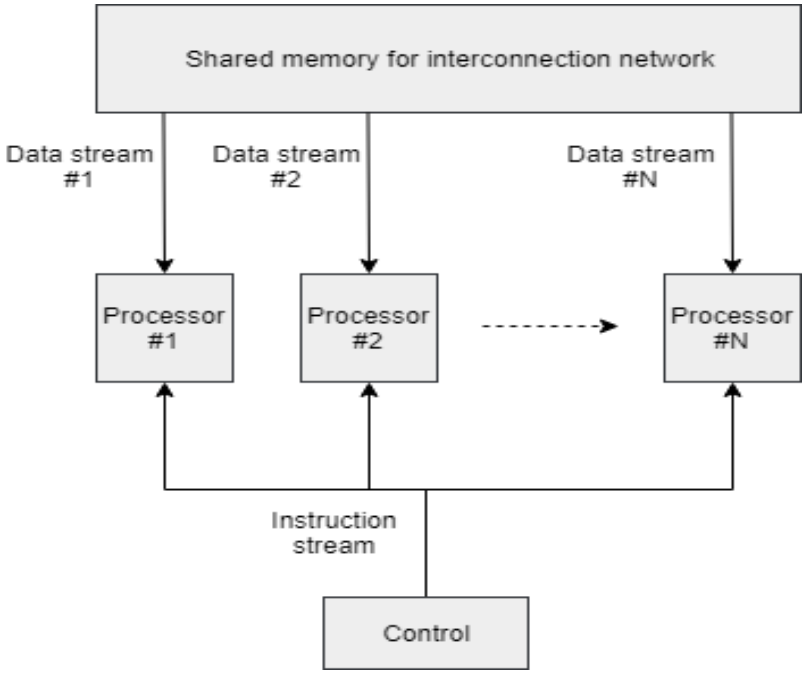


FIGURE 4.3: SIMD computers.

- MIMD (multiple instruction streams-multiple data streams): In MIMD design, there are multiple independent processors, where each processor can execute different instructions on different data, as shown in figure 4.4. Standard component (e.g network, processors) are employed in this type of parallel.

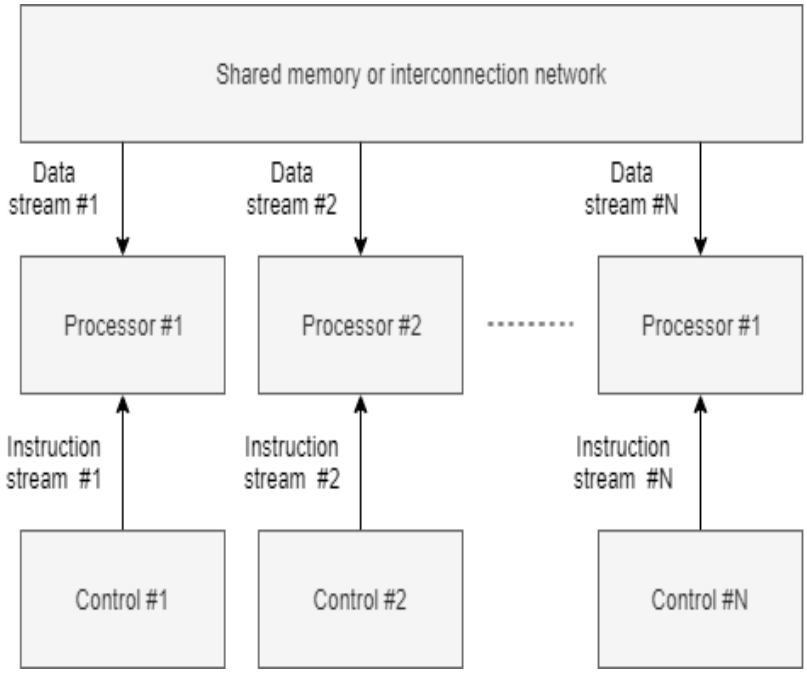


FIGURE 4.4: MIMD computers

Taxonomy based on memory organization consists of: distributed memory and shared memory (e.g. UMA, NUMA). In distributed memory, each processor has its own memory, and the communication preformed between processors by message passing. In shared memory, a single address is shared by all processors. Another taxonomy, which is processor granularity, it contains three sub-type: coarse grained (limited powerful processors), fined grained (many small processors) and medium grained (hybrid between coarse and fined). The taxonomy based on processor synchronization could be asynchronous with independent clocks for each processor, fully synchronous with global clock for all processors and bulk-synchronous which is considered a hybrid between the previous types. Finally taxonomy based on interconnection architectures, where the network is utilized statically (e.g. point-to-point), or dynamically (e.g. using switches, crossbar, buses) [46].

4.1.1 Dedicated Architectures

In addition to above parallel mechanisms, it is possible to take advantage of some dedicated hardware to implement parallel metaheuristic algorithms. Field programmable gate arrays (FPGAs) and graphical processing unit (GPU) are well known examples of dedicated hardware. As mentioned before, metaheuristic algorithms required heavy resources, so dedicated hardware can assist CPU to do other jobs by executing different tasks on their boards. For example calculating KNN into GPU and its results goes to CPU to calculate fitness function [46].

4.2 Parallel Programming Environments and Middlewares

Middleware is a software layer located between the operating system kernel and user application, which facilitates the development of software through hiding distributed applications complexity and providing a generic interface to make applications portable and reusable. The parallel architectures mentioned in the previous section play an important role in choosing the parallel programming paradigms. The parallel programming paradigm is divided into two types: shared memory and distributed memory, as shown in figure 4.5.

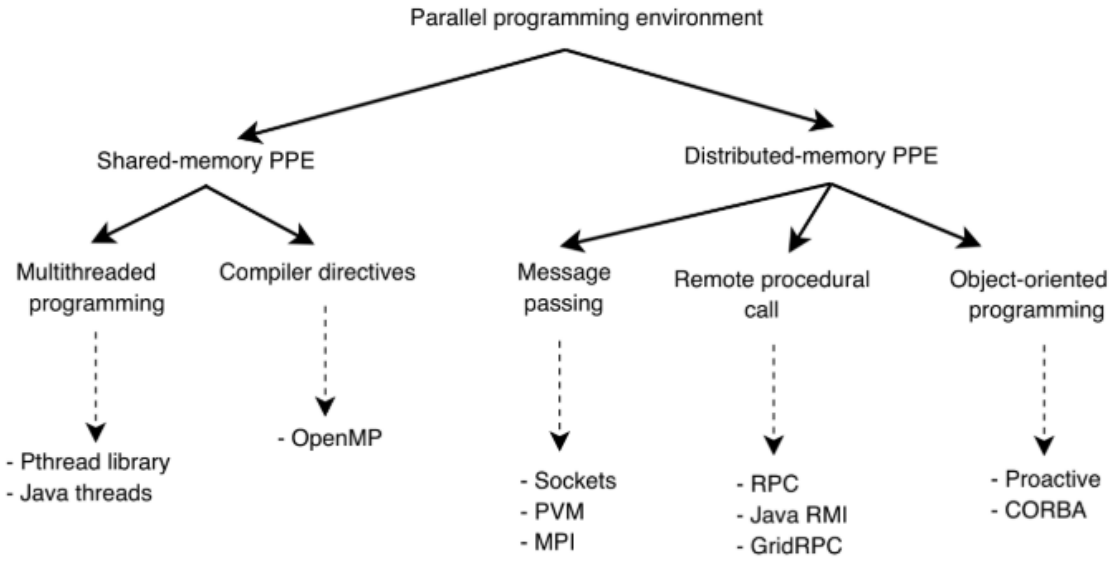


FIGURE 4.5: Main parallel programming languages, programming environments and middlewares.

The shared memory paradigm allows all processes to share the same data. Hence, the programmer is responsible to handle data access among processes (synchronization conditions) to avoid data corruption or deadlock by using mutex or semaphores. Following are alternatives to program such architectures:

- Multithreading: Thread is the smallest sequence of instructions inside the process. Multiple threads can be executed in parallel by a scheduler and they share the same memory space of the process.

- Compiler directives: Compiler directives (pragma) specifies how a compiler should process its input. OpenMP (open multiprocessing) is a set of compiler directives that supports multi-platform shared memory multiprocessing programming. OpenMP is portable and scalable, it supports multiple programming languages to develop a parallel system such: C, C++, and Fortran. In addition, its support different operating systems such Linux and Microsoft Windows [53].

  By using OpenMP directives, developers can easily transform sequential algorithms into parallel ones. Furthermore, OpenMP can be controlled by system environments to change its behavior (e.g. change number of threads).

Distributed memory has the following three paradigms:

- Message passing: Is the most common paradigm of parallel architectures, where processes or threads exchange information using synchronous or asynchronous messages. Network socket is considered the most famous environment for message passing paradigm.

- Remote procedure call (RPC): Is a protocol that allows program to request function located in another computer over the network regardless of understanding the network details

- Object-oriented models: Its an object oriented extension of the RPC. Java RMI (remote method invocation) is one example of this paradigm.

4.3 Performance Evaluation

The performance in the parallel algorithms is measured by the execution time exactly as in the sequential algorithms, but in addition, the number of processors and the characteristics of parallel architecture are taken into consideration. The scalability of parallel algorithms means getting performance is proportional to the number of processors. To evaluate the scalability of parallel algorithms both speedup and efficiency are utilized [54].

The speedup S_N is defined as the time T_1 it takes to complete a program with one processor divided by the time T_N it takes to complete the same program with N processors, as shown in equation 4.1. Wall clock time can be used to calculate the whole program time including I/O operations.

$$S_N = \frac{T_1}{T_N} \tag{4.1}$$

Average speedup is used in case of stochastic metaheuristics that depends on solution quality to stop execution, as shown in equation 4.2.

$$S_N = \frac{Average(T_1)}{Average(T_N)} \tag{4.2}$$

The efficiency E_N using N processors is defined as the speedup S_N divided by the number of processors N, as shown in equation 4.3. Efficiency indicates how N processors are utilized while the program is running.

$$E_N = \frac{S_N}{N} \tag{4.3}$$

Finally, it is important to use the same random seed -which is used to generate random numbers- in order to have a deterministic random generation cross different runs to calculate the performance.

4.4 Parallel Design Of Metaheuristics

This section presents the three main levels used in designing a parallel metaheuristics. The levels are algorithmic level, iteration level and solution level. As shown in figure 4.6.

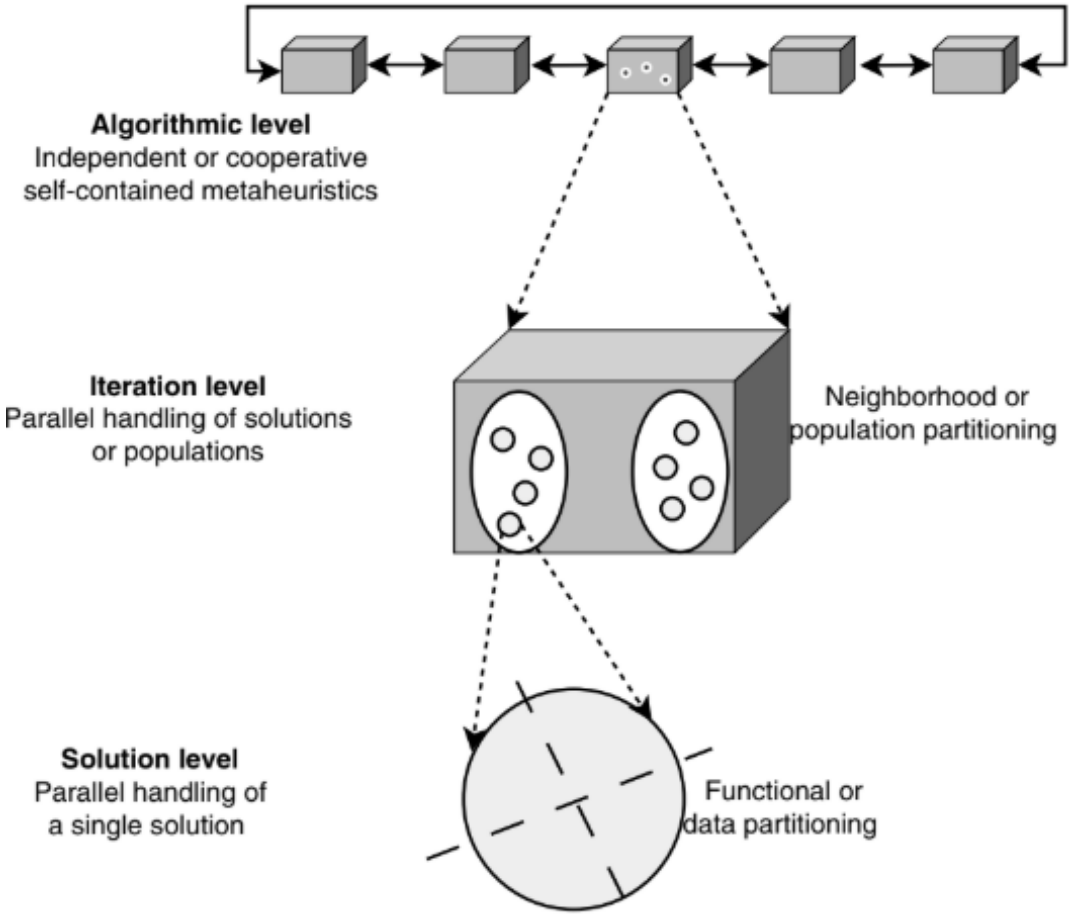


FIGURE 4.6: Combination of the three parallel hierarchical models of metaheuristics [46].

4.4.1 Algorithmic-Level Parallel Model

At this level, independent or collaborative metaheuristics are used. In case of independent, research is equivalent to the sequential search in terms of quality of the result, but less in terms of time. In the case of a collaborative model, the metaheuristics affect each other, sharing together to improve the result [46].

Independent Algorithmic-Level Parallel Model

In this type, a group of independent metaheuristics coexist in parallel to solve a specific problem. Each single metaheuristic may differ than others in solution initialization, parameter settings, fitness function, stopping criteria, and others [46]. This type is simple and easy to implement. It consists of a master and a number of workers. A master provides the workers with the necessary parameters to start the process of finding the best solution, as shown in figure 4.7

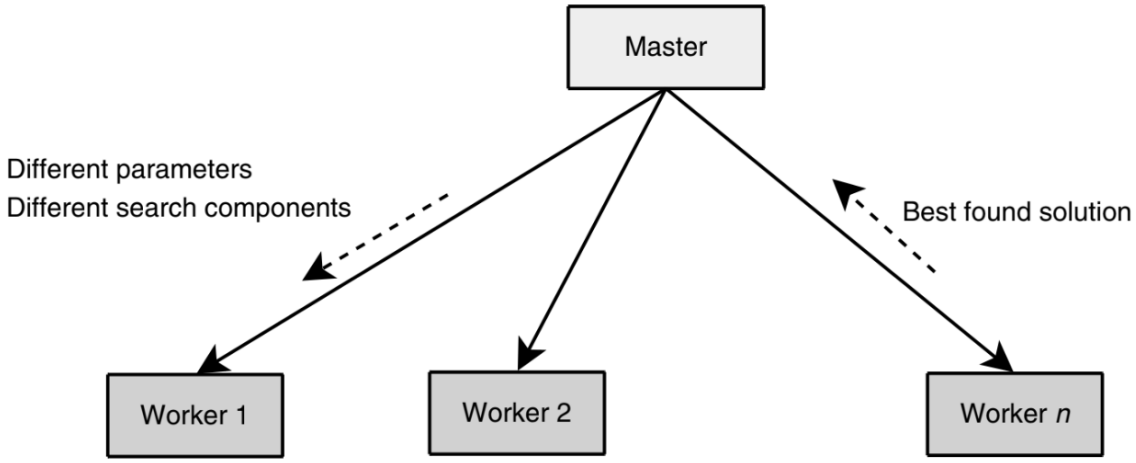


FIGURE 4.7: The parallel independent algorithmic-level model for metaheuristics [46].

Cooperative Algorithmic-Level Parallel Model

In this parallel type, metaheuristics (homogeneous, or heterogeneous) are cooperate together to get the best solution. To design such model, different considerations must be taken first, figure 4.8 illustrates the major questions in designing an algorithmic-level model for metaheuristics.

Regarding to choose the appropriate time to exchange messages between metaheuristic (when?), there are several ways to do this: periodic (after a fixed number of iterations), Probabilistic, and adaptive (run-time based on a search characteristics). Different topologies are used to exchange information between metaheuristic (where?) such: ring, 2D mesh, hypercube of order 3, and complete graph. As to the content of exchanging messages between

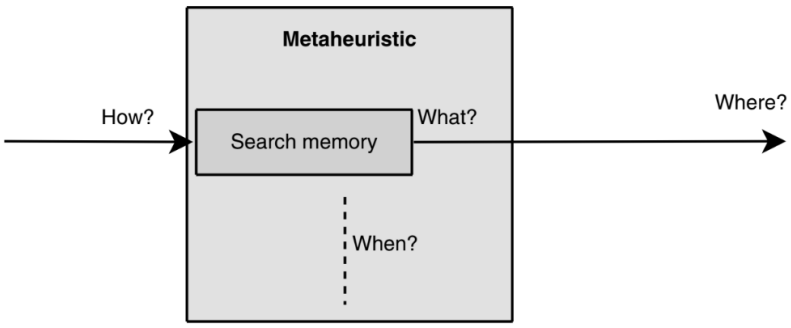


FIGURE 4.8: Design questions involved by the parallel algorithmic-level model for metaheuristics [46].

metaheuristics (what?). The content may includes elite solution (local best) or search memory (any element found in search memory). Finally, the integration policy (how?) deals with received information, where local variables are replaced by global best [46].

4.4.2 Iteration-Level Parallel Model

It is well known that the metaheuristic, both types S-metaheuristics and P-metaheuristics, require enormous computational resources through the large number of iteration cycles to reach near optimal solutions. To speed up metaheuristic algorithms, iteration-level parallel model is deployed. In this model, the problems are independent, so its easy to split the iterations into a number of parallel cycles to work together [46].

On one hand, S-Metaheuristics can take advantage of the parallel model in both generation and evaluation of the neighborhood to speed up the searching process. In this model, the iteration is divided into several segments (partitions) each one runs on different processor. In general, the number of segments is equal, and the maximum number of segments is equal to number of solutions [46].

To select the improved neighbor from parallel partitions, two strategies are utilized:

- Best improving neighbor: a synchronous search (wait all partitions till finish), where the parallel partitions are searched to find the best neighbor.
- First improving neighbor: asynchronous search (no need to explore all partitions), where the exploration stops once the improved neighbor is found.

On other hand, P-metaheuristics can be optimized in term of execution time by using iteration level mode of parallel, since P-metaheuristics has independant set of solution, which make it easy to split the iteration over a number of partitions. For example, Grey Wolf Optimizer (GWO) consists of population of agents (solutions), each solution can run on processor. To select the best solution both synchronous and asynchronous are utilized [46].

4.4.3 Solution-Level Parallel Model

Evaluating the fitness (objective) function in metaheuristicsn is the frequently used and the most costly operation. Solution-level parallel model, is a problem-dependent aims to speed up the search and reduce the time needed for the evaluation by applying the parallelization of the evaluation of a single solution [46].

Following are three models used in solution-level parallel:

- Functional decomposition: Is a synchronous model, where the evaluation function (fitness) is divided into several sub functions that executed in parallel. The results of the sub functions are then collected as a single result (reduction operation).
- Data partitioning: In some problems, data are huge. Therefore fitness function can not deal with such data due to memory limitation. Instead, a distributed databases is used to store the data. So, similar instances of fitness functions are performed on different data partitions.
- Hybrid model: It is a model that combines both functional decomposition and data partitioning.

4.4.4 Main Properties of Parallel Metaheuristics

Parallel metaheuristic is measured by maximum number of parallel processes regardless the parallel architecture. In addition to the number of processors, there are a number of important issues that affect the parallel metaheuristics, for example: asynchronous communications, processors integrate different parallel elements integration (e.g. GPU, DMA, FPU),

input/output (I/O) operations, and scheduling strategies. Following are scheduling strategies [46]:

- Static scheduling: in this type, the number of parallel tasks (jobs) and data location is defined during the compile time. This type is more suitable with homogeneous and non-shared data heterogeneous metaheuristics.
- Dynamic scheduling: it is similar to the static scheduling except that the data determines at run time. Thus, load balancing is essential for shared architectures.
- Adaptive scheduling: in this type tasks are created automatically during the node hibernation, and tasks are killed when nodes becomes non idle. Finally, adaptive load balancing must be handled.

Chapter 5

The Implementation of Parallel BGWO

Umayaparvathi and Iyakutti [56] proposed a five-phase model to construct a customer churn prediction, as shown in figure 5.1. The process of building the model starts with data gathering from telecom companies. The second phase, is data preparation, where the obtained data are often incomplete or contain irrelevant items. Then derived variables, where data expert extract the attributes from the data. In customer churn, derived variables are not important, with an assumption that data are arranged in database with columns that represent the attribute name, and rows that represent the customers. Extracting variables is the next phase, which determine the minimal set of variable using several techniques (e.g. Feature Selection algorithms) to build accurate model with reasonable time. Finally model construction which used to guess the customer churn.

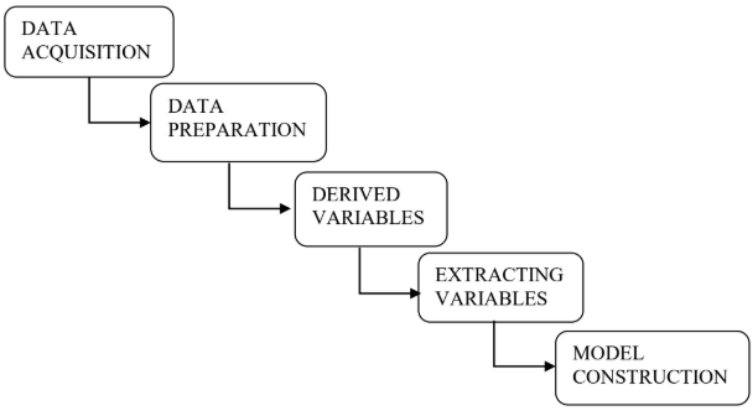


FIGURE 5.1: Data modelling steps [56].

The following sections deal with data acquisition 5.1 and processing 5.2 as an important steps before building a correct churn model [55], as well as presenting the proposed approach 5.3.

5.1 Data Acquisition

Because its difficult to obtain real customer dataset from local telecommunications companies, due to data privacy (confidentiality) or competitive considerations, we used Kaggle [1] dataset repository instead. Kaggle is a cloud-based workbench for data science, owned by Google. The dataset includes 7043 customers and 20 customer properties, in addition a labeled propriety (churn/ not churn). Table 5.1 represents the dataset properties, which includes information about:

- Churn: Customers status who left or still in company within the last month.
- Services requested by the customer: phone, multiple lines, tech support, online security, streaming TV and movies, device protection, internet, and online backup.
- Customers account information: contract, payment method, paperless billing, monthly charges, total charges and how long the customer spent in the company.
- Customers demographic information: gender, and if they have dependents and/or partners.

[1] https://www.kaggle.com

Field name	Field type	Field values	Description
customerID	string	unique values	
gender	string	Male, Female	Whether the customer is a male or a female
SeniorCitizen	numeric	0, 1	Whether the customer is a senior citizen or not
Partner	string	No, Yes	Whether the customer has a partner or not
Dependents	string	No, Yes	Whether the customer has dependents or not
tenure	numeric	0 to 72	Number of months the customer has stayed with the company
PhoneService	string	No, Yes	Whether the customer has a phone service or not
MultipleLines	string	No, Yes	Whether the customer has multiple lines or not
InternetService	string	DSL, Fiber optic, No	Customers internet service provider
OnlineSecurity	string	Yes, No, No internet service	Whether the customer has online security or not
OnlineBackup	string	Yes, No, No internet service	Whether the customer has online backup or not
DeviceProtection	string	Yes, No, No internet service	Whether the customer has device protection or not
TechSupport	string	Yes, No, No internet service	Whether the customer has tech support or not
StreamingTV	string	Yes, No, No internet service	Whether the customer has streaming TV or not
StreamingMovies	string	Yes, No, No internet service	Whether the customer has streaming movies or not
Contract	string	Month-to-month, One year, Two year	The contract term of the customer
PaperlessBilling	string	No, Yes	Whether the customer has paperless billing or not
PaymentMethod	string	Electronic check, Mailed check, Bank transfer, Credit card	The customers payment method
MonthlyCharges	numeric	18.3 to 119	The amount charged to the customer monthly
TotalCharges	numeric	18.8 to 8.68k	The total amount charged to the customer
churn	string	No, Yes	Whether the customer churned or not

TABLE 5.1: Churn in Telecom's dataset from Kaggle [2].

[2]https://www.kaggle.com/blastchar/telco-customer-churn

5.2 Preprocessing

Customer dataset is sometimes incomplete or includes irrelevant data (noisy), and such data affects the accuracy of the intended customer model. The database must contain fields that are directly related to call charges, call usage (e.g. total day calls) [55]. In this section, we will present the data cleaning process that used in the customer dataset.

5.2.1 Data Cleaning

In this step, both features, and instances are cleaned (eliminated) from the dataset, in order to build accurate predictive model.

Following are some customer fields that should be neglected when building the predictive model, since they do not have predictive value:

- Unique values, including unlock codes, customer addresses. In this regards, we removed 'customerID' column from the dataset.
- Common values such geographical.
- Fields with only one value.
- Fields with repetitive 'null' value. These fields are exist in prepaid customers, where the customer is not obliged to give personal information such as age, sex, etc. In our case 'TotalCharges' have empty value, so we removed the correlated row.

Furthermore, the customer instance in the dataset will be eliminated if he/she have zero volumes (no transaction) within specific time interval (depends on telecom company, but usially 2-4 months), or the customer have a recordings for one month, and other months were missing.

5.3 The Proposed Approach

In this section, we will present multiple offspring sampling technique (5.3.1) and its interaction with the proposed Grey Wolf Optimizer (GWO) (5.3.2 and 5.3.3). We selected the

Grey Wolf Optimizer (GWO) algorithm as the most cited paper with more than 2000 citation indexes based on Science Direct website [3], in addition to the ease of converting the algorithm into parallel form using OpenMP framework. Finally, The GWO algorithm relies on the hierarchical nature to search for a solution, represented by wolves, which provides an opportunity to obtain a more accurate results.

Multiple Offspring Sampling (MOS) is a technique that used different update functions to advance Grey Wolf position to better location. We developed two parallel algorithms based on the original Gray Wolf optimizer. The first algorithm runs an independent (GWO) instances (5.3.2), and selects the best solution at the end of the algorithm. The second algorithm is an improvement on the first algorithm, where the cooperation is applied between the GWO instances (5.3.3). In this regards, the master thread stops all (GWO) instances after a certain iterations to collect alpha values from workers, and then populate the best alpha over all instances. It is worth mentioning that, the master thread will make mutation on some instances to make sure not all instances go toward the same location in the search space.

5.3.1 Multiple Offspring Sampling (MOS)

Experience has shown that evolutionary algorithms (e.g. genetic algorithm) can solve many problems. But this does not mean that they can solve all the problems. For example, changing dataset or parameters may affect the algorithm's success. In this regard, The No Free Lunch Theorem (NFL) means "any two algorithms are equivalent when their performance is averaged across all possible problems" [59]. This theory leads us to conclude that a hybrid of several algorithms or several parameters can guarantee the solution in different situations, this principle is called (MOS).

Multiple offspring Sampling (MOS) is a technique to create new solutions (individuals) using certain evolutionary algorithm, encoding, operators (if required), and parameters [61].

Following are MOS taxonomies based on the above definition:

- Algorithm-based MOS: create new individuals by using different algorithms (e.g. GAs, EDAs).

[3] https://www.sciencedirect.com/science/article/abs/pii/S0965997813001853

- Coding-based MOS: solution representation can be done using different codings (genotypes).
- Operator-based MOS: this type is usually used with genetic algorithms, where multiple operators are applied to the solution simultaneously.
- Parameter-based MOS: different evolutionary parameters are used, such as crossover, selection, mutation, etc.

Table 5.2 presents the update functions used in this thesis.

Id	Function Name	Equation
1	Linear [45]	$Fx = 2 - current\_itr \frac{2}{total\_itr}$
2	Inertia strategy [96]	$Fx = 0.9 - (0.9 - 0.4)(\frac{current\_itr}{total\_itr})^{\frac{1}{\pi^2}}$
3	Nonlinear improved [96]	$Fx = 0.3 \times 1.0002^{-current\_itr}$
4	Decreasing inertia weight [96]	$Fx = (\frac{2}{current\_itr})^{0.3}$
5	Natural exponent ineria weight [95]	$Fx = 0.4 + (0.9 - 0.4) \times e^{-10 \times \frac{current\_itr}{total\_itr}}$
6	Oscillating inertia weight [94]	$\text{Fx} = \begin{cases} \frac{1.2}{2} + \frac{0.6}{2} \times & \\ \quad cos(2\pi current\_itr \frac{34}{3total\_itr}) & current\_itr < 3\frac{total\_itr}{4} \\ 0.3 & elsewhere \end{cases}$
7	PSO with Sugeno inertia weight [93]	$Fx = 1 - \frac{current\_itr}{total\_itr} \div 1 + 2\frac{current\_itr}{total\_itr}$
8	Logarithm decreasing inertia weight [92]	$Fx = 0.9 + 1.3 \times \log_{10}(1 + 10\frac{current\_itr}{total\_itr})$

TABLE 5.2: Update functions used in parallel MOS BGWO.

Combining several algorithms together (hybridization) leads to the following cases [60]:

- Collaborative algorithms: improves the performance of the best algorithm when it is used individually.
- Competitive selection: select the best algorithm in performance.

5.3.2 Independent parallel BGWO

MOS are used to generate new individuals, which leads to an improvement in the solution. We have taken advantage of the MOS principles but with a slight modification, where we pass

a different update functions (table 5.2) to different instances of BGWO, in order to move the solution (exploitation or exploration). In other words, All BGWO instances are running in parallel, where each instance have different update function. At the end of BGWO iteration, we select the best solution (the highest accuracy) generated from the BGWO instances. Figure 5.2 illustrates the flow diagram for this algorithm. Finally, we named the proposed approach as piMBGWO (parallel-independent-MOS-BGWO).

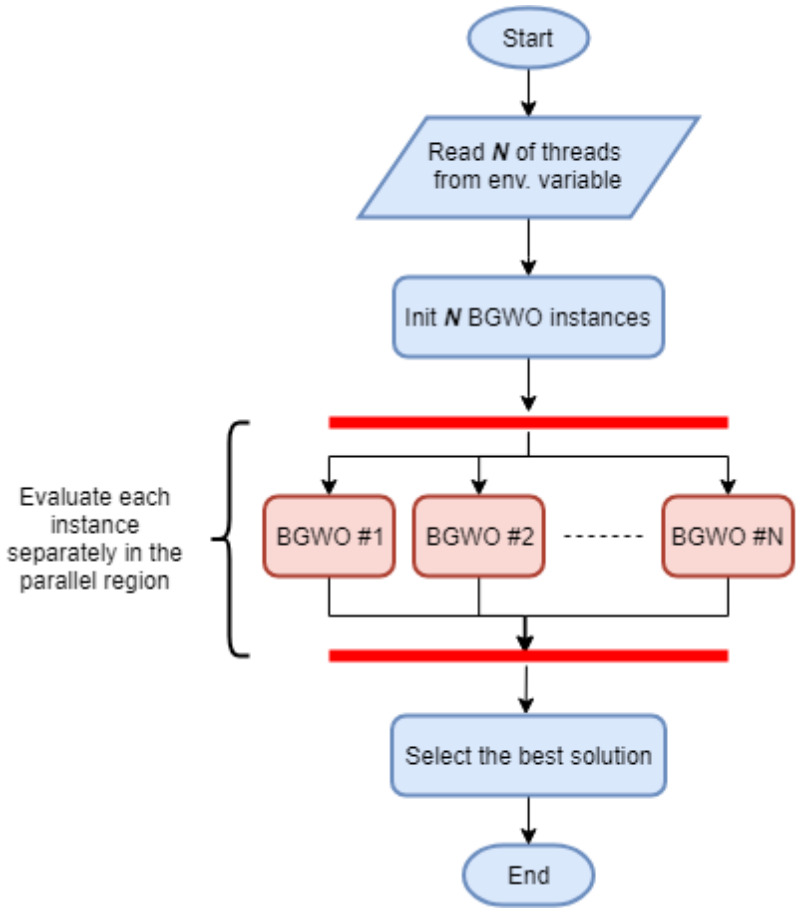


FIGURE 5.2: Parallel independent MOS BGWO flowchart.

5.3.3 Cooperative parallel BGWO

This algorithm is an improvement to the previous algorithm. Where a master thread runs the independent BGWO algorithms a specified number of iterations and then collects an alpha value. The master thread stops the independent algorithms every 25% (K) percentage of

the total number of iterations. For example, if we assume that the number of iterations is 100, the master thread will stop the independent BGWO 4 times to collect alpha values. We performed a number of experiments to calculate the value of (K) on the UCI datasets, and concluded that 25% is the best value. Table 5.3 shows cooperative parallel BGWO settings. As with the first approach, each independent instance of BGWO algorithm running an update function that differ from its counterparts as shown in figure 5.3.

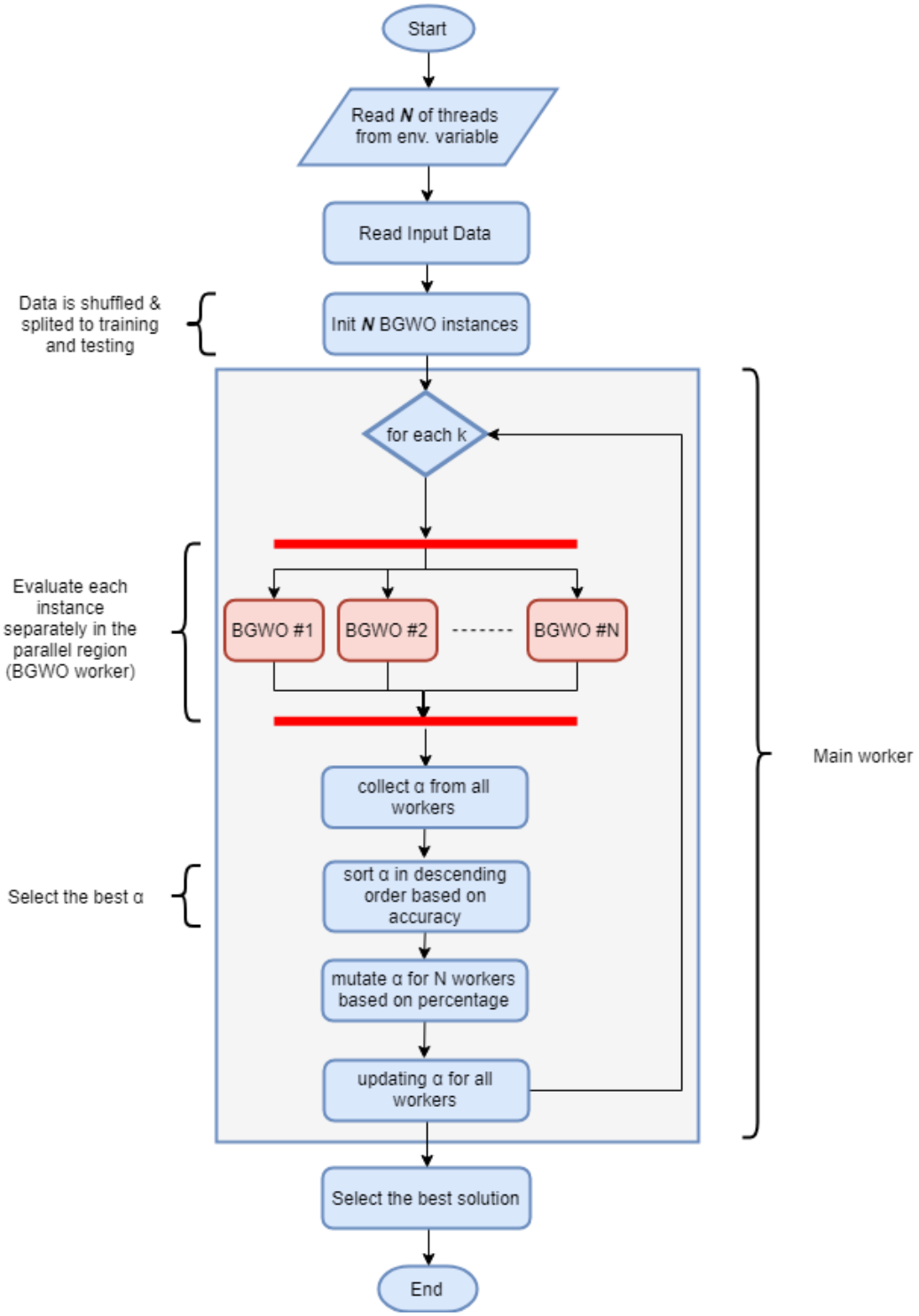


FIGURE 5.3: cooperative parallel BGWO flowchart.

After collecting alpha values, the master thread sort alpha values by descending accuracy, then selects the heights value (the first item). The master thread then distributes the highest

value to all BGWO instances. It is worth mentioning that the phase of collecting alpha values consists of: BGWO current accuracy, alpha score and alpha position (current solution). Finally, the master thread apply mutation on BGWO instances to prevent them go towards same location in the search space. Algorithm 6 presents the mutation function followed in this approach. Where the basic idea in this function is flipping alpha position by applying (xor) operator randomly based on mutation rate. Finally, we named the proposed approach as pcMBGWO (parallel-Cooperative-MOS-BGWO).

Parameter	**Value**
K percentage	0.25
Mutation rate	0.02 [4]
Mutation percentage	0.3 [4]

TABLE 5.3: Cooperative parallel BGWO settings.

Algorithm 6 Cooperative parallel BGWO mutation function

```
1:  procedure MUTATION(alpha_pos, mutation_rate)
2:      nmu = ceil(mutation_rate * size_of_alpha_pos)
3:      result = alpha_pos
4:      i = 0
5:      while i < nmu do
6:          j = random_number_between(0, size_of_alpha_pos)
7:          result[j] = alpha_pos[j] xor 1
8:          i = i + 1
9:      return result
10: end procedure
```

[4] https://fr.mathworks.com/matlabcentral/fileexchange/52856-binary-and-real-coded-genetic-algorithms

Chapter 6

Experiments

In this chapter, we will discuss the tools and libraries used in the algorithm development process as described in section experiment setup (6.1). In addition, we will explain the benchmark datasets used to evaluate the proposed algorithms as shown in section benchmark datasets (6.2). Finally, we illustrate the collected results and provide a discussion as described in section results and discussion (6.3).

6.1 Experiment Setup

The proposed algorithms were developed under Debian 64bit operating system (version 8.1.0), with tool-chains (development tools) as shown in table 6.1.

Tool name	Version	Description
g++	5.4.0	C++ compiler (standard 0x11)
ld	2.26.1	GNU linker
make	4.1	GNU make (builder)
Shark	3.1.0	Machine learning library
OpenMP	2.0	Shared memory multi-threading library
Python	2.7	Python scripting language to calculate Wilcoxon (p_value)
Scipy	1.2.2	Open-source Python library used for scientific computing (Wilcoxon)
IBM SPSS	22	Statistical analysis software (Friedman test)

TABLE 6.1: Tool-chains used in the development.

However, we used Intel(R) processor i7-4770 to have good infrastructure for parallelization, and total memory of 16 GB. Table 6.2 presents the used processor specifications.

Property name	Property value
Vendor	Intel(R) Core^{TM}
Model	i7-4770
Frequency	3.40 GHz
CPU cores	4
Siblings (Threads)	8
Cache	8 MB
Instructions width	64 bits

TABLE 6.2: CPU specifications used in development.

Finally, table 6.3 shows the parameters tuning used in the proposed algorithm. It is worth mentioning that, we used four threads in the proposed algorithm to match the CPU threads that used in the experiment as shown in table 6.2.

Parameter	**Value**
KNN threads	4
BGWO threads	4
Population size	10 [45]
Iterations	100 [45]
Training percent	80% [45]
Testing percent	20% [45]
Number of run (average)	20 [45]
α in pBGWO	[2,0] decreasing, equations 5.2
KNN classifier	K=5 [39]
KNN distance metric	Euclidean [64]

TABLE 6.3: Parameters tuning for the proposed algorithm.

6.2 Benchmark Datasets

We evaluated the proposed algorithms on 18 well known datasets from UCI repository [62], as shown in table 6.4.

Dataset	# Features	# Instances	# Classes	Domain
Breastcancer	9	699	2	Biology
BreastEW	30	569	2	Biology
CongressEW	16	435	2	Politics
Exactly	13	1000	2	Biology
Exactly2	13	1000	2	Biology
HeartEW	13	270	2	Biology
IonosphereEW	34	351	2	Electromagnetic
KrvskpEW	36	3196	2	Game
Lymphography	18	148	2	Biology
M-of-n	13	1000	2	Biology
PenglungEW	325	73	2	Biology
SonarEW	60	208	2	Biology
SpectEW	22	267	2	Biology
Tic-tac-toe	9	958	2	Game
Vote	16	300	2	Politics
WaveformEW	40	5000	3	Physics
WineEW	13	178	3	Chemistry
Zoo	16	101	6	Artificial

TABLE 6.4: List of used datasets in the experiments.

6.3 Results and Discussion

We conducted a set of experiments to measure the effectiveness of the proposed algorithms (pMBGWO), (piMBGWO) and (pcMBGWO). The proposed algorithms consist of four instances running in parallel by OpenMp framework, and each instance run its own update function. Table 6.5 illustrates the selected update functions based on their ranks. The four update functions are: natural exponent ineria weight (5), inertia strategy (2), nonlinear

improved (3) and oscillating inertia weight (6) respectively as show in table 5.2.

The selected update functions are derived by the following procedure:

1. Apply all update functions on each dataset using (piMBGWO) algorithm. The result will be a huge matrix with all possible combinations.
2. Sorting each dataset by fitness value in ascending order.
3. Creating a table the includes the update function id ranks as appeared in each dataset. For instance, the update function number (7) has a rank (1) in Breastcancer dataset, and update function number (2) has a rank (4) in the same dataset, etc.
4. Calculate the average rank of each update function.
5. Sorting the average rank in ascending order.
6. Collect the first four update functions, since we are using four threads in our experiment.

Dataset	Function id / Rank							
	5	**2**	**3**	**6**	**8**	**7**	**4**	**1**
Breastcancer	2	4	8	3	6	1	5	7
BreastEW	2	5	6	1	4	3	8	7
Exactly	5	1	3	4	2	6	8	7
Exactly2	3	4	5	2	8	1	6	7
HeartEW	2	5	1	3	7	8	4	6
Lymphography	1	3	6	4	8	7	2	5
M-of-n	2	5	3	1	4	7	8	6
PenglungEW	4	5	1	6	2	3	7	8
SonarEW	2	1	3	6	7	4	5	8
SpectEW	3	8	6	5	2	4	1	7
CongressEW	2	3	1	7	4	5	6	8
IonosphereEW	4	6	3	2	5	1	8	7
KrvskpEW	6	1	2	5	3	4	8	7
Tic-tac-toe	5	1	8	6	7	3	2	4
Vote	7	4	3	2	1	6	8	5
WaveformEW	4	6	1	3	2	5	8	7
WineEW	2	3	7	4	1	8	6	5
zoo	2	5	3	7	4	6	8	1
Average	**3.222**	**3.889**	**3.889**	**3.944**	4.278	4.556	6.000	6.222

TABLE 6.5: The update functions rank (ascending order by average column).

We compared the original grey wolf algorithm (BGWO) with (pBGWO), (piMBGWO) and (pcMBGWO) in terms of fitness, accuracy, number of features and execution time (in seconds); as showed in tables [6.6, 6.7, 6.8 and 6.9]. In addition, we plotted the tables as graphs as shown in figure [6.1].

Dataset	BGWO	pBGWO	piMBGWO	pcMBGWO
Breastcancer	0.023	0.024	0.023	**0.021**
BreastEW	0.031	0.026	0.024	**0.021**
CongressEW	0.037	0.023	0.024	**0.021**
Exactly	0.208	0.102	0.080	**0.079**
Exactly2	0.210	0.206	0.207	**0.203**
HeartEW	0.092	0.075	0.073	**0.070**
Iono-sphereEW	0.069	0.052	0.049	**0.039**
KrvskpEW	0.032	0.028	0.024	**0.020**
Lymphogra-phy	**0.085**	0.078	0.103	0.098
M-of-n	0.063	0.034	0.020	**0.018**
penglungEW	0.214	0.190	0.186	**0.143**
SonarEW	0.022	0.025	0.019	**0.010**
SpectEW	0.125	0.103	0.105	**0.093**
Tic-tac-toe	0.098	0.101	0.098	**0.096**
Vote	0.037	0.024	0.025	**0.023**
WaveformEW	0.199	0.190	0.186	**0.178**
WineEW	**0.004**	0.005	0.005	**0.004**
Zoo	0.003	0.004	0.003	**0.002**

TABLE 6.6: Comparison between BGWO, pBGWO, piMBGWO and pcM-BGWO in terms of fitness average.

Dataset	BGWO	pBGWO	piMBGWO	pcMBGWO
Breastcancer	0.976	0.979	**0.981**	0.980
BreastEW	0.972	0.980	0.981	**0.982**
CongressEW	0.957	0.978	0.979	**0.989**
Exactly	0.752	0.880	0.900	**0.986**
Exactly2	0.768	0.792	0.791	**0.793**
HeartEW	0.894	0.926	0.926	**0.930**
Iono-sphereEW	0.936	0.953	0.954	**0.964**
KrvskpEW	0.974	0.979	0.982	**0.987**
Lymphogra-phy	0.913	**0.922**	0.902	0.900
M-of-n	0.921	0.963	0.980	**0.997**
penglungEW	0.790	0.810	0.813	**0.867**
SonarEW	0.983	0.980	0.986	**0.994**
SpectEW	0.872	0.899	0.896	**0.910**
Tic-tac-toe	0.841	0.897	**0.904**	0.901
Vote	0.962	**0.978**	0.977	0.976
WaveformEW	0.803	0.811	0.816	**0.825**
WineEW	0.999	**1.000**	**1.000**	**1.000**
Zoo	**1.000**	**1.000**	**1.000**	0.995

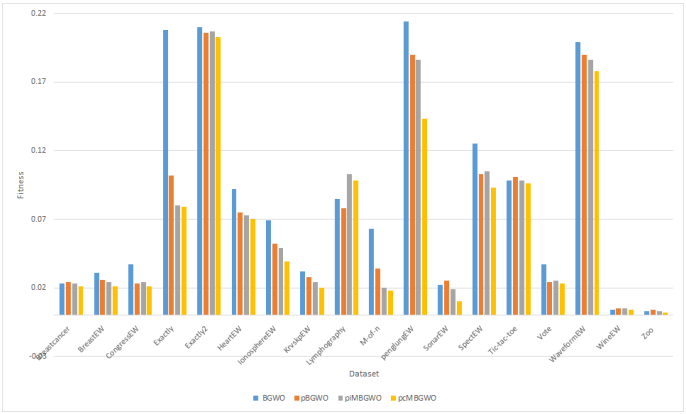
TABLE 6.7: Comparison between BGWO, pBGWO, piMBGWO and pcM-BGWO in terms of accuracy average.

Dataset	BGWO	pBGWO	piMBGWO	pcMBGWO
Breastcancer	5.15	4.6	4.6	**3.7**
BreastEW	17.45	17	15.4	**12.95**
CongressEW	7.8	7.1	5.95	**5.2**
Exactly	9.7	7.9	7.45	**6.85**
Exactly2	6.75	6	5.5	**4.9**
HeartEW	8.6	7.75	7.85	**6.8**
IonosphereEW	17.9	19.15	17.35	**12.9**
KrvskpEW	28.1	27.6	24.75	**22.6**
Lymphography	7.9	10.05	9.85	**6.8**
M-of-n	9.1	8.15	7.2	**6.15**
penglungEW	187.6	183.85	161.25	**127.25**
SonarEW	31.2	36.15	29.9	**23**
SpectEW	14.75	13.25	12	**10.35**
Tic-tac-toe	9	8.8	9	**7.85**
Vote	7.9	5.95	5.95	**4.25**
WaveformEW	30.55	29.2	26.5	**25.5**
WineEW	5.1	7	6.3	**4.8**
Zoo	5	7	5.05	**3.85**

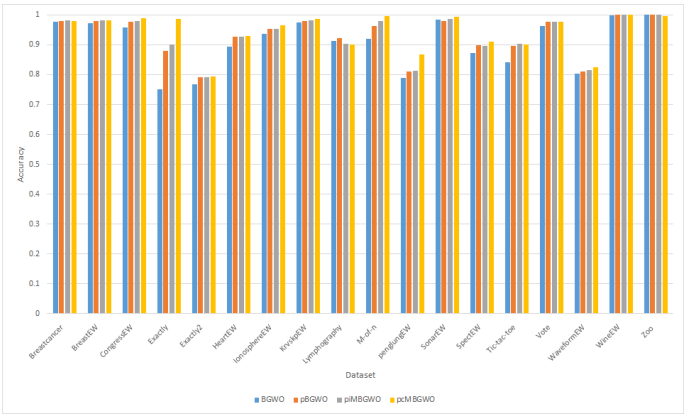
TABLE 6.8: Comparison between BGWO, pBGWO, piMBGWO and pcMBGWO in terms of number of features average.

Dataset	BGWO	pBGWO	piMBGWO	pcMBGWO
Breastcancer	**0.455**	0.543	0.535	8.278
BreastEW	**0.576**	0.660	0.598	6.480
CongressEW	**0.298**	0.413	0.392	6.009
Exactly	**1.009**	1.131	1.092	18.945
Exactly2	**0.908**	1.032	0.979	22.637
HeartEW	**0.136**	0.187	0.181	2.335
IonosphereEW	**0.295**	0.414	0.397	4.347
KrvskpEW	**16.123**	20.274	16.698	170.443
Lymphography	**0.059**	0.112	0.113	1.210
M-of-n	**1.000**	1.142	1.096	20.552
penglungEW	**0.210**	1.704	1.755	2.229
SonarEW	**0.199**	0.402	0.368	2.355
SpectEW	**0.179**	0.237	0.225	2.841
Tic-tac-toe	**0.842**	1.173	1.117	18.111
Vote	**0.156**	0.307	0.261	3.408
WaveformEW	**37.650**	49.043	44.964	379.654
WineEW	**0.074**	0.115	0.114	1.522
Zoo	**0.039**	0.094	0.093	0.844

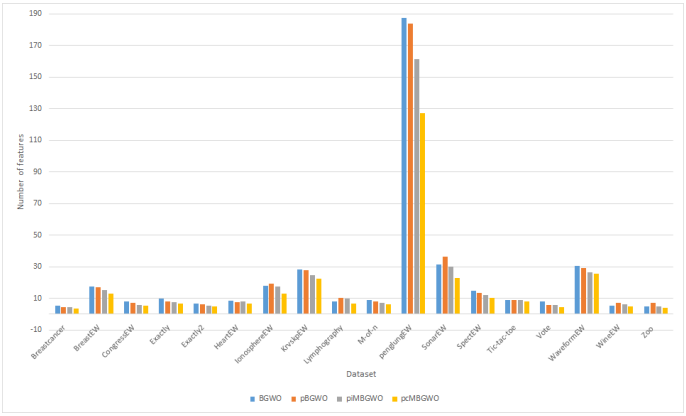
TABLE 6.9: Comparison between BGWO, pBGWO, piMBGWO and pcMBGWO in terms time (seconds).



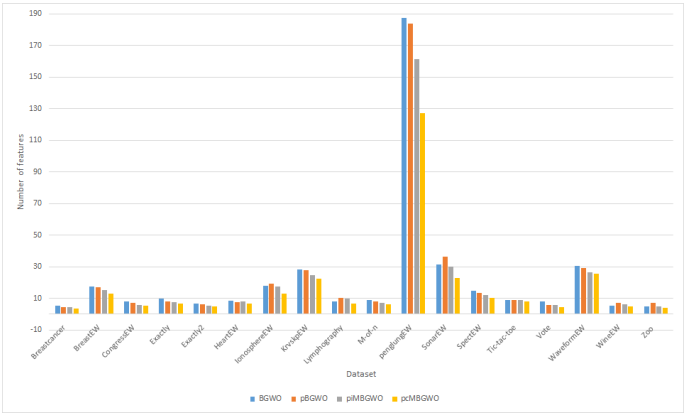
(A) Fitness



(B) Accuracy



(C) Features



(D) Time (seconds)

FIGURE 6.1: Graph comparison between BGWO, pBGWO, piMBGWO and pcMBGWO.

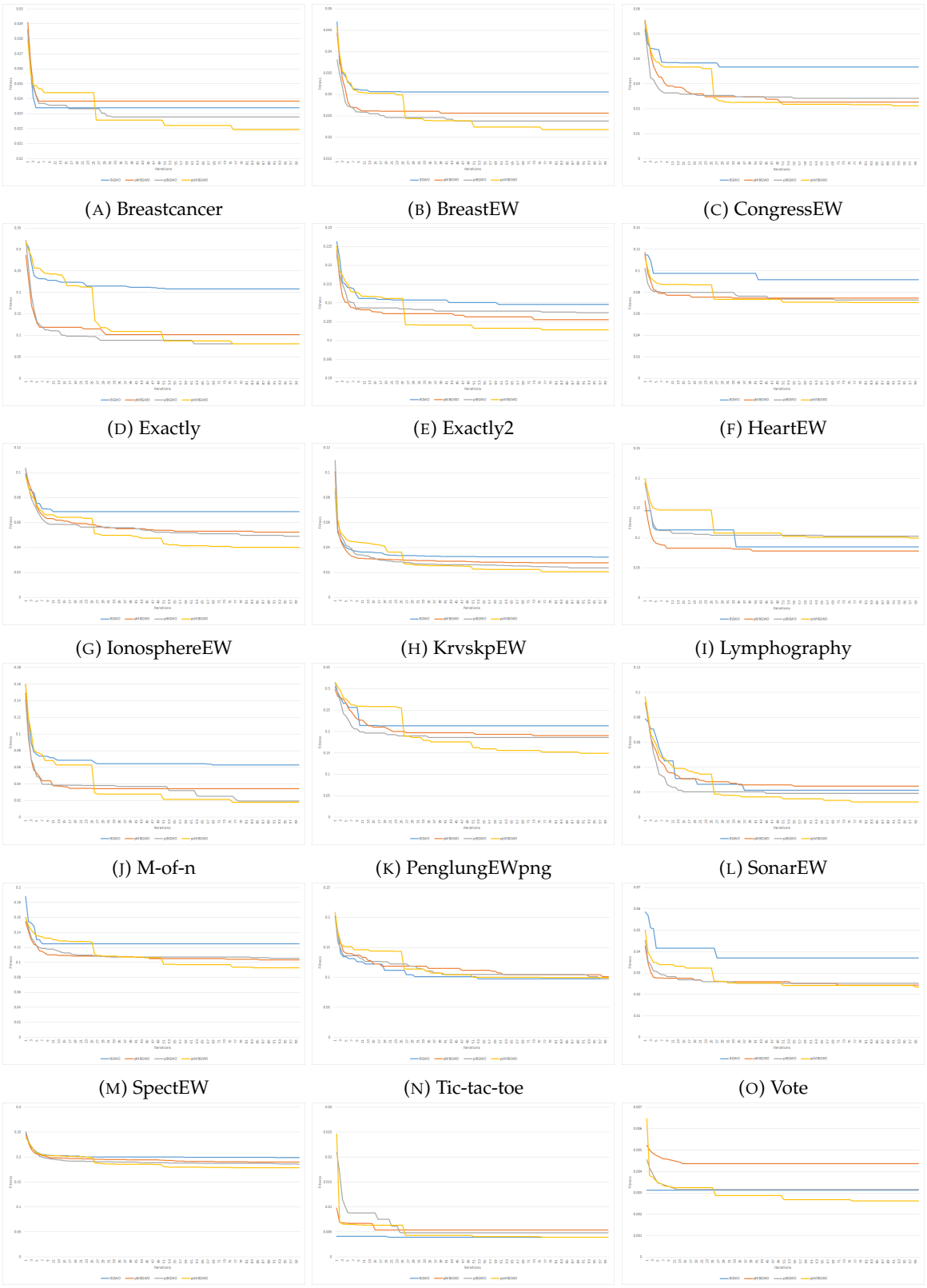
By reviewing the results above, we observe the superiority of the proposed models: co-operative (pcMBGWO) and independent (piMBGWO) over the original algorithm (BGWO) in terms of execution fitness average, accuracy average and number of features average.

Algorithms can be arranged in terms of fitness average as follows: pcMBGWO > piMBGWO > pBGWO > BGWO. Where (pcMBGWO) outperforms the other algorithms in 17 out of 18 datasets. Based on this arrangement, the third research question (1.3) RQ 3 was answered. The (pcMBGWO) also outperforms other algorithms regarding accuracy average by 13 out of 18 datasets.

Moreover, the (pcMBGWO) outperforms other algorithms regarding number of features average cross all datasets. In this regards, we obtained a results with high accuracy with minimum number of features, and so we answered the research question (1.3) RQ 2.

On the other hand, the results showed that the (pcMBGWO) algorithm takes more time compared to other algorithms due to the time spent in collecting the information from the workers, calculating the best alpha value from workers, applying mutation and distribution of the best solution to all workers. It is worth mentioning that the parallel mechanism using OpenMp (pcMBGWO) has achieved a superior time in both algorithms: pBGWO and piMBGWO. For example the execution time in the Breastcancer using the original algorithm is 0.455 seconds, and if the algorithm is executed four times in succession, the time becomes 1.82 second. However, using a OpenMp mechanism, the time of pBGWO and piMBGWO is 0.543 seconds, 0.535 seconds respectively.

In addition, the fitness convergence curve has been plotted for all algorithms over the 18 datasets as shown in figure [6.2]. The graph show the superiority of the proposed algorithm (pcMBGOW). As mentioned earlier in this thesis (subsection 2), Grey Wolf Optimizer (GWO) counters bias towards the exploitation, so it suffers from stacking at the local optima. As shown in the Yellow Line, the proposed algorithm has overcome the problem of local optima by conducting evaluation every 25% of the iterations as mentioned in (section 5.3.3).



(A) Breastcancer (B) BreastEW (C) CongressEW

(D) Exactly (E) Exactly2 (F) HeartEW

(G) IonosphereEW (H) KrvskpEW (I) Lymphography

(J) M-of-n (K) PenglungEWpng (L) SonarEW

(M) SpectEW (N) Tic-tac-toe (O) Vote

(P) WaveformEW (Q) WineEW (R) Zoo

FIGURE 6.2: Fitness convergence curve for benchmark datasets.

Statistically speaking, we defined the null hypothesis H_0 as follows: There are no differences between the proposed approaches and the original approach; $\alpha = 5\%$. In this regard. Both F-test, and Wilcoxon test has been conducted on collected results to validate the proposed algorithms.

We applied the Friedman test (F-test) to determine the rank of the algorithms in terms of fitness (table 6.6), accuracy (table 6.7), number of features (table 6.8) and execution time (table 6.9). Friedman test is a non-parametric statistical test used to detect differences in treatments across multiple test attempts. Tables (6.10, 6.11, 6.12 and 6.13) represent the rank of algorithms according to the F-test. The F-test results showed that the proposed algorithm (pcMBGWO) surpassed (rank 1) the other algorithms BGWO, pMBGWO and piBGWO in term of fitness, accuracy and number of features. But the proposed algorithm comes last (rank 4) in term of execution time due to threading overhead other factors as mentioned above.

Algorithm	BGWO	pMBGWO	piBGWO	pcMBGWO
Mean rank (F_test)	3.44	2.92	2.50	1.14
Overall rank	4	3	2	**1**

TABLE 6.10: F-test ranking for BGWO, pBGWO, piMBGWO and pcMBGWO in term of fitness.

Algorithm	BGWO	pMBGWO	piBGWO	pcMBGWO
Mean rank (F_test)	3.72	2.58	2.08	1.61
Overall rank	4	3	2	**1**

TABLE 6.11: F-test ranking for BGWO, pBGWO, piMBGWO and pcMBGWO in term of accuracy.

Algorithm	BGWO	pMBGWO	piBGWO	pcMBGWO
Mean rank (F_test)	3.53	3.11	2.36	1.00
Overall rank	4	3	2	**1**

TABLE 6.12: F-test ranking for BGWO, pBGWO, piMBGWO and pcMBGWO in term of number of features.

Algorithm	BGWO	pMBGWO	piBGWO	pcMBGWO
Mean rank (F_test)	1.00	2.89	2.11	4.00
Overall rank	**1**	3	2	4

TABLE 6.13: F-test ranking for BGWO, pBGWO, piMBGWO and pcMBGWO in term of time (second).

In order to reject the null hypothesis and support our approaches, we have to calculated the probability value (P-value). So the proposed approaches pBGWO, piMBGWO and pcMBGWO will be valid if $p\_value < \alpha$. To calculate the (P-value) we used the Wilcoxon test because of the nature of the random and binary data (not having data that is normally distributed). Tables 6.14, 6.15 and 6.16 compare pcMBGWO versus BGWO, pMBGWO and piBGWO in term of accuracy, number of features and fitness using Wilcoxon test (P-value). Its worth mentioning that, F-test has been calculated using IBM SPSS, and Wilcoxon test (P-value) has been calculated using python/Scipy library.

Dataset	pcMBGWO versus		
	BGWO	pMBGWO	piBGWO
Breastcancer	**4.38E-02**	1.07E-01	6.17E-01
BreastEW	**7.00E-04**	5.78E-02	2.48E-01
CongressEW	**1.20E-04**	**1.24E-02**	**2.17E-03**
Exactly	**8.66E-05**	**2.10E-04**	**1.38E-03**
Exactly2	**2.99E-04**	7.15E-01	3.02E-01
HeartEW	**1.04E-04**	6.50E-01	5.86E-01
IonosphereEW	**2.34E-04**	**2.63E-03**	**4.73E-03**
KrvskpEW	**1.40E-04**	**8.56E-04**	**2.51E-03**
Lymphography	3.41E-01	**2.36E-02**	7.98E-01
M-of-n	**8.76E-05**	**4.14E-04**	**6.44E-03**
PenglungEW	**1.20E-03**	**6.19E-03**	**1.78E-02**
SonarEW	**6.66E-03**	**1.34E-03**	**3.48E-02**
SpectEW	**1.13E-04**	**3.58E-02**	**1.52E-02**
Tic-tac-toe	**1.18E-03**	6.80E-01	5.81E-01
Vote	**1.76E-03**	4.39E-01	7.63E-01
WaveformEW	**1.20E-04**	**8.80E-05**	**3.83E-04**
WineEW	3.17E-01	NaN	NaN
zoo	3.17E-01	3.17E-01	3.17E-01

TABLE 6.14: Wilcoxon test (P-value) for pcMBGWO versus BGWO, pBGWO, piMBGWO in term of accuracy.

Dataset	pcMBGWO versus		
	BGWO	pMBGWO	piBGWO
Breastcancer	2.18E-01	4.05E-01	5.31E-01
BreastEW	**5.88E-04**	**2.75E-04**	9.19E-02
CongressEW	**1.30E-02**	2.18E-01	3.56E-01
Exactly	**7.88E-05**	**1.43E-04**	**1.57E-03**
Exactly2	1.46E-01	5.41E-01	9.38E-01
HeartEW	**2.96E-03**	3.03E-01	3.78E-01
IonosphereEW	**3.70E-04**	**1.25E-03**	**3.89E-03**
KrvskpEW	**1.25E-03**	**3.25E-03**	2.91E-01
Lymphography	1.68E-01	**4.75E-03**	7.16E-02
M-of-n	**2.79E-04**	**1.26E-03**	**3.71E-02**
PenglungEW	**8.82E-05**	**8.82E-05**	**1.62E-04**
SonarEW	**4.09E-04**	**8.60E-05**	**7.76E-03**
SpectEW	**1.22E-04**	**2.46E-04**	**1.36E-02**
Tic-tac-toe	NaN	3.17E-01	NaN
Vote	**4.41E-03**	1.08E-01	4.19E-01
WaveformEW	**1.06E-03**	**2.57E-02**	6.17E-01
WineEW	**7.24E-03**	**4.89E-02**	6.70E-01
zoo	7.26E-01	**6.85E-04**	8.75E-01

TABLE 6.15: Wilcoxon test (P-value) for pcMBGWO versus BGWO, pBGWO, piMBGWO in term of number if features.

Dataset	pcMBGWO versus		
	BGWO	pMBGWO	piBGWO
Breastcancer	**3.51E-02**	**1.57E-02**	1.76E-01
BreastEW	**8.92E-04**	**5.11E-03**	**2.42E-02**
CongressEW	**2.89E-03**	5.57E-01	1.13E-01
Exactly	**1.82E-04**	3.51E-01	8.52E-01
Exactly2	**5.73E-03**	1.79E-01	1.79E-01
HeartEW	**1.62E-04**	1.42E-01	4.07E-01
IonosphereEW	**2.92E-04**	**4.04E-03**	**6.93E-03**
KrvskpEW	**1.40E-04**	**2.54E-04**	**2.34E-03**
Lymphography	4.94E-01	4.46E-01	2.27E-01
M-of-n	**1.16E-03**	**2.16E-02**	6.79E-01
PenglungEW	**1.16E-03**	**2.49E-03**	**3.18E-03**
SonarEW	**8.37E-04**	**2.92E-04**	**2.82E-03**
SpectEW	**8.67E-05**	**6.22E-04**	**4.05E-03**
Tic-tac-toe	**7.46E-04**	**7.82E-04**	**7.46E-04**
Vote	**1.45E-03**	2.55E-01	8.89E-02
WaveformEW	**1.03E-04**	**8.92E-04**	**2.65E-03**
WineEW	**1.33E-02**	**2.85E-04**	**8.39E-04**
zoo	**1.56E-03**	**1.48E-03**	**1.85E-02**

TABLE 6.16: Wilcoxon test (P-value) for pcMBGWO versus BGWO, pBGWO, piMBGWO in term of fitness.

An overview of P-values as shown in the tables (6.14, 6.15 and 6.16) look like less than α, which means that there are a meaningful differences, so the null hypothesis can be rejected and we confirmed the proposed approaches to be statistically significant. Hence, we claim that, the parallel mechanism using OpenMp over MOS technique gives better results than the original algorithm in both independent and cooperative models. It is worth mentioning that the "NaN" value appeared in some of previous tables, and this value means that some

data samples (algorithms) are equal or has the same average, and sometimes means (division by zero).

After we confirmed that the proposed algorithm (pcMBGWO) gives the best results, we compared it against the following algorithms in terms of accurecy over the UCI dataset (6.4): Whale Optimization Algorithm (WOA), Genetic Algorithm (GA), Particle Swarm Algorithm, and Dragonfly Algorithm as shown in table [6.17].

Dataset	pcMBGWO	WOA	GA	PSO	DF
Breastcancer	**0.980**	0.957	0.968	0.967	0.963
BreastEW	**0.982**	0.955	0.939	0.933	0.961
CongressEW	**0.989**	0.929	0.932	0.928	0.967
Exactly	**0.986**	0.757	0.674	0.688	0.98
Exactly2	**0.793**	0.698	0.746	0.73	0.745
HeartEW	**0.930**	0.763	0.78	0.787	0.83
IonosphereEW	**0.964**	0.89	0.814	0.819	0.93
KrvskpEW	**0.987**	0.915	0.92	0.941	0.953
Lymphography	**0.900**	0.785	0.696	0.744	0.877
M-of-n	**0.997**	0.854	0.861	0.921	0.992
PenglungEW	**0.867**	0.729	0.584	0.584	0.845
SonarEW	**0.994**	0.854	0.754	0.737	0.915
SpectEW	**0.910**	0.787	0.793	0.822	0.853
Tic-tac-toe	**0.901**	0.751	0.719	0.735	0.788
Vote	**0.976**	0.938	0.904	0.904	0.958
WaveformEW	**0.825**	0.712	0.773	0.762	0.75
WineEW	**1.000**	0.928	0.937	0.933	0.98
Zoo	**0.995**	0.964	0.855	0.861	0.958

TABLE 6.17: The proposed approach pcMBGWO versus WOA, GA, PSO and DF in term of accuracy over UCI dataset.

Obviously, the proposed algorithm (pcMBGWO) outperforms the competing metaheuristic algorithms by a noticeable difference as shown in table [6.17]. So, we confidently used the proposed algorithms on telecom customers dataset as described in section 5.1 in order to reduce the table dimension by extracting the most important attributes (Feature Selection). Then, we used these features to predict the customer churns. In this regard, we used the data mining algorithm K-Nearest Neighbors (KNN) on the extracted attributes for churn prediction, then we compared the results against the non-trimmed table (without Feature Selection) using the following machine learning algorithms: K-Nearest Neighbors (KNN), Random Forests (RF), Decision Tree (CART) and Gaussian (NB) as shown in table 6.18. The results showed the superiority of (pcMBGWO) approach in term of accuracy. With these results we answered the following research questions (1.3): RQ 1, RQ 4.

Category	Algorithm name	Accuracy
Grey Wolf Optimizer	BGWO	0.775
	pBGWO	0.783
	piMBGWO	0.784
	pcMBGWO	**0.806**
Machine learning	KNeighbors (K-NN)	0.748
	Random Forests (RF)	0.762
	Decision Tree (CART)	0.725
	Gaussian (NB)	0.701

TABLE 6.18: Comparing the proposed approaches with selected machine learning techniques on telecom customers dataset.

It is worth mentioning that the accuracy of predicting the customers churn on telecom dataset based on reduced number of attributes by applying the proposed algorithms: BGWO (0.775), pBGWO (0.783), piMBGWO (0.784) and pcMBGWO (0.806) were significantly better than predicting the customers churn using the whole attribute set using machine learning algorithms; specifically the K-Nearest Neighbors (KNN) algorithm (0.748).

The K-Nearest Neighbors (KNN) algorithm is known to have the following issues: struggles to give the accurate prediction in high dimensional dataset, does not predict accurately on imbalanced data and sensitive to outliers. In this regards, the proposed algorithms solved the high dimensional dataset problem by reducing the number of attributes (features), and create a table with fewer features while maintaining a high accuracy.

Chapter 7

Conclusion and Future Work

In this thesis, two parallel models of Grey Wolf Optimizer (GWO) based on Multiple-offspring-sampling technique was proposed. The parallel models (piMBGWO & pcMBGWO) used OpenMP parallel framework. A well known benchmarks from UCI repository were utilized to validate the proposed approach. We conducted a number of experiments on the proposed approaches, in addition a comparison experiments with original and parallel version of GWO as well. The proposed approaches provides promising results to be used as customer churn prediction model. Finally We recommend using parallel independent model to get competitive results in term of time and accuracy. Furthermore, we recommend using the cooperative model, which outperforms the independent model in accuracy if the execution time is not important in solving a problem.

As a future work, we intend to use dedicated hardware (CUDA/NVIDIA GPU) for metaheuristic search instead of OpenMP, since we anticipate to deal with enormous size of customer's valuable and hidden data stored in telecom companies' warehouses (big-data). Thus we will be able to build a model that predict customer churn using couple thousands of GPU cores, which is definitely will boost the search time. On other hand, using ensemble model of different metaheuristics search algorithms such: BGWO, genetic algorithm, PSO, Whale, etc. will increase the search capabilities, since each algorithm have its own characteristics, e.g GWO have more exploitation (local optima), on the other hand, WOA have more exploration. Of course, all these algorithms will run in parallel environment.

References

[1] J. Hadden, A. Tiwari, R. Roy, and D. Ruta, "Computer assisted customer churn management: State-of-the-art and future trends," Comput. Oper. Res., vol. 34, no. 10, pp. 2902–2917, Oct. 2007.

[2] Amal M. Almana et al Int. Journal of Engineering Research and Applications, ISSN : 2248-9622, Vol. 4, Issue 5(Version 6), May 2014, pp.165-171

[3] Lin, W.C., Tsai, C.F. and Ke, S.W., (2014), "Dimensionality and data reduction in telecom churn 22 prediction", Kybernetes, 43(5), pp.737-749.

[4] A. Sharma and P. Prabin Kumar, "A Neural Network based Approach for Predicting Customer Churn in Cellular Network Services," Int. J. Comput. Appl., vol. 27, no. 11, pp. 26–31, 2011.

[5] W. Verbeke, D. Martens, C. Mues, and B. Baesens, "Building comprehensible customer churn prediction models with advanced rule induction techniques," Expert Syst. Appl., vol. 38, no. 3, pp. 2354–2364, Mar. 2011.

[6] C. Kirui, L. Hong, W. Cheruiyot, and H. Kirui, "Predicting Customer Churn in Mobile Telephony Industry Using Probabilistic Classifiers in Data Mining," IJCSI Int. J. Comput. Sci. Issues, vol. 10, no. 2, pp. 165–172, 2013.

[7] B. Huang, M. T. Kechadi, and B. Buckley, "Customer churn prediction in telecommunications," Expert Syst. Appl., vol. 39, no. 1, pp. 1414–1425, Jan. 2012.

[8] C.-S. Lin, G.-H. Tzeng, and Y.-C. Chin, "Combined rough set theory and flow network graph to predict customer churn in credit card accounts," Expert Syst. Appl., vol. 38, no. 1, pp. 8–15, Jan. 2011.

[9] Shin-Yuan Hung , David C. Yen , H. Wang, Applying data mining to telecom, Expert Systems with Applications 31 (2006) 515–524, Elseiver.

[10] V. Lazarov and M. Capota, "Churn Prediction," Bus. Anal. Course. TUM Comput. Sci., 2007.

[11] R. Mansouri, M. Saraee, and R. Amirfattahi, "Application of Data Mining in Predicting Cell Phones Subscribers Behavior Employing the Contact Pattern," in 2010 International Conference on Data Storage and Data Engineering (DSDE), 2010, pp. 63–68.

[12] O. R. Zaïane, "Introduction to Data Mining." CMPUT690 Principles of Knowledge Discovery in Databases Chapter, pp. 1–15, 1999.

[13] C. Shearer, "The CRISP-DM model: The New Blueprint for Data Mining," J. Data Warehous., vol. 5, no. 4, pp. 13–22, 2000.

[14] Z. Kasiran, Z. Ibrahim, and M. S. M. Ribuan, "Mobile phone customers churn prediction using elman and Jordan Recurrent Neural Network," in Computing and Convergence Technology (ICCCT), 2012 7th International Conference on, 2012, pp. 673–678.

[15] S. A. Qureshi, A. S. Rehman, A. M. Qamar, A. Kamal, and A. Rehman, "Telecommunication subscribers' churn prediction model using machine learning,"in Digital Information Management (ICDIM), 2013 Eighth International Conference on, 2013, pp. 131–136.

[16] W. H. Au, K. C. C. Chan, and Y. Xin, "A novel evolutionary data mining algorithm with applications to churn prediction," Evol.Comput. IEEE Trans., vol. 7, no. 6, pp. 532–545, 2003.

[17] A. Sharma and P. K. Panigrahi, "A Neural Network based Approach for Predicting Customer Churn in Cellular Network Services," Int. J. Comput. Appl., vol. 27, no.11, 2011.

[18] V. Lazarov and M. Capota, "Churn Prediction," TUM Comput. Sci., 2007.

[19] S. A. Qureshi, A. S. Rehman, A. M. Qamar, A. Kamal, and A. Rehman, "Telecommunication subscribers' churn prediction model using machine learning," in 2013 Eighth International Conference on Digital Information Management (ICDIM), 2013, pp. 131–136.

[20] Q. Shen, H. Li, Q. Liao, W. Zhang, and K. Kalilou, "Improving churn prediction in telecommunications using complementary fusion of multilayer features based on factorization and construction," in The 26th Chinese Control and Decision Conference (2014 CCDC), 2014, pp. 2250–2255.

[21] A. T. Jahromi, M. Moeini, I. Akbari, and A. Akbarzadeh, "A Dual-Step Multi-Algorithm Approach For Churn Prediction In Pre-Paid Telecommunications Service Providers,"RISUS. J. Innov. Sustain., vol. 1, no. 2, 2010.

[22] M. Kaur, K. Singh, and N. Sharma, "Data Mining as a tool to Predict the Churn Behaviour among Indian bank customers, "Int. J. Recent Innov. Trends Comput. Commun., vol. 1, no. 9, pp. 720–725, 2013.

[23] R. A. Soeini and K. V. Rodpysh, "Evaluations of Data Mining Methods in Order to Provide the Optimum Method for Customer Churn Prediction: Case Study Insurance Industry," 2012 Int. Conf. Inf. Comput. Appl. (ICICA 2012), vol. 24, pp. 290–297, 2012.

[24] V. Yeshwanth, V. V. Raj, and M.Saravanan, "Evolutionary Churn Prediction in Mobile Networks USing Hybrid Learning," in Twenty-Fourth International FLAIRS Conference, 2011, pp. 471–476.

[25] L. Breiman. Random forests. Machine learning, 45(1):5–32, 2001.

[26] N. Singh, R.S. Raw, and R.K. Chauhan, "Data mining with regression technique," J. Information Systems and Communication, vol. 3, no. 1, pp. 199–202, 2012.

[27] Hosmer DW Jr, Lemeshow S, Sturdivant RX (2013) Applied Logistic Regression. Third Edition. New Jersey: John Wiley Sons

[28] G. Nie, W. Rowe, L. Zhang, Y. Tian, and Y. Shi, "Credit card churn forecasting by logistic regression and decision tree," Expert Syst. Appl., vol. 38, no. 12, pp. 15273–15285, 2011.

[29] S. V Nath and R. S. Behara, "Customer churn analysis in the wireless industry: A data mining approach," in Proceedings-Annual Meeting of the Decision Sciences. Institute, 2003.

[30] Y. Zhang, J. Qi, H. Shu, and J. Cao, "A hybrid KNN-LR classifier and its application in customer churn prediction," in 2007 IEEE International Conference on Systems, Man and Cybernetics, 2007, pp. 3265–3269.

[31] Y. Huang and T. Kechadi, "An effective hybrid learning system for telecommunication churn prediction,"Expert Syst. Appl., vol. 40, no. 14, pp. 5635–5647, Oct. 2013.

[32] I. Brandusoiu and G. Toderean, "Churn Prediction in the Telecommunications Sector using Support Vector Machines," Ann. ORADEA Univ. Fascicle Manag. Technol. Eng., no. 1, 2013.

[33] Jiayin Qi, Li Zhang, et al. ADTreesLogit model for customer churn prediction. Annual of operation research, 2009, 168(1): 247-265.

[34] Yangming Zhang, Jiayin Qi, Huaying Shu, Jiantong Cao. A hybrid KNN-LR classifier and its application in customer churn prediction. 2007 IEEE International Conference on Systems, Man and Cybernetics, Oct. 7-10, 2007,3265-3269.

[35] Jin Zhang, D. "Research on Rough Set Theory Based Data Mining Algorithm.," Unpublished doctoral dissertation, Northwestern Polyteehnieal University, China. December, 2005.

[36] R.Kohavi and B.Frasea.. "Useful Feature Subsets and Rough set reducts," International Workshop on Rough Sets and Soft Computing (RSSC), pp. 310-317, 1994.

[37] A. M. Taha and A. Y. Tang, "Bat algorithm for rough set attribute reduction," Journal of Theoretical and Applied Information Technology, vol. 51, no. 1, pp. 1–8, 2013.

[38] M. Dash and H. Liu, "Feature selection for classification," Intelligent data analysis, vol. 1, no. 3, pp. 131–156, 1997.

[39] M. Mafarja, I. Aljarah, A. A. Heidari, A. I. Hammouri, H. Faris, A.-Z. Ala'M, and S. Mirjalili, "Evolutionary population dynamics and grasshopper optimization approaches for feature selection problems," Knowledge-Based Systems, 2017.

[40] H. Liu and H. Motoda, Feature selection for knowledge discovery and data mining. Springer Science Business Media, 2012, vol. 454.

[41] M. M. Mafarja and S. Mirjalili, "Hybrid whale optimization algorithm with simulated annealing for feature selection," Neurocomputing, vol. 260, pp. 302–312, 2017.

[42] M. Mafarja and S. Mirjalili, "Whale optimization approaches for wrapper feature selection," Applied Soft Computing, vol. 62, pp. 441–453, 2018.

[43] J. H. Holland, "Genetic algorithms," Scientific american, vol. 267, no. 1, pp. 66–73, 1992.

[44] B. Webster and P. J. Bernhard, "A local search optimization algorithm based on natural principles of gravitation," Tech. Rep., 2003.

[45] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," Advances in engineering software, vol. 69, pp. 46–61, 2014.

[46] Talbi E (2009) Metaheuristics: from design to implementation. John Wiley Sons, Hoboken

[47] Yang, X.S.: Engineering Optimization: An Introduction with Metaheuristic Applications. Wiley, Chichester (2010)

[48] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," ACM computing surveys (CSUR), vol. 35, no. 3, pp. 268–308, 2003.

[49] S. Mirjalili and A. Lewis, "The whale optimization algorithm," Advances in Engineering Software, vol. 95, pp. 51–67, 2016.

[50] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization-artificial ants as a computational intelligence technique," IEEE computational intelligence magazine, vol. 1, no. 4, pp. 28–39, 2006.

[51] J. Kennedy and R. Eberhart, "particle swarm optimization," in proceedings of the 1995 ieee international conference on neural networks, vol. 4, perth, australia, ieee service center, piscataway, nj, 1995, pp." 1942.

[52] A. Kaveh and N. Farhoudi, "A new optimization method: dolphin echolocation," Advances in Engineering Software, vol. 59, pp. 53–70, 2013.

[53] B. Chapman, G. Jost, R. VanderPas, and D. J. Kuck. Using OpenMP: Portable Shared Memory Parallel Programming. MIT Press, 2007.

[54] V. Kumar, A. Grama, A. Gupta, and G. Karypis. Introduction to Parallel Computing: Design and Analysis of Algorithms. Addison-Wesley, 1994.

[55] M. Berry and G. Linoff. Mastering Data Mining. John Wiley and Sons, New York, USA, 2000.

[56] V. Umayaparvathi and K. Iyak, "Applications of Data Mining Techniques in Telecom, " in 2012 International Journal Of Computer Applications (0957-8887), vol 42, No.20, Mar.

[57] Habbas Z., Krajecki M., Singer D. [2000], Domain Decomposition for Parallel Resolution of Constraint Satisfaction Problems with OpenMP, In: Proceedings of The Second European Workshop on OpenMP, Edinburgh, Scotland, 1-8.

[58] Christian Igel, Verena Heidrich-Meisner, and Tobias Glasmachers. Shark. Journal of Machine Learning Research 9, pp. 993-996, 2008.

[59] Wolpert D, Macready W (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput l(l):67-82.

[60] LaTorre A (2009) A framework for hybrid dynamic evolutionary algorithms: multiple offspring sampling (mos). Ph.D. thesis, Universidad Politecnica de Madrid (November 2009).

[61] LaTorre, Antonio, Santiago Muelas, and José-María Peña. "A MOS-based dynamic memetic differential evolution algorithm for continuous optimization: a scalability test." Soft Computing 15, no. 11 (2011): 2187-2199.

[62] D. Dheeru and E. Karra Taniskidou, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[63] Amin A, Anwar S, Adnan A, Nawaz M, Alawfi K, Hussain A, Huang K. Customer churn prediction in the telecommunication sector using a rough set approach. Neurocomputing. 2017 May 10;237:242-54.

[64] E. Emary, H. M. Zawbaa, and A. E. Hassanien, "Binary grey wolf optimization approaches for feature selection," Neurocomputing, vol. 172, pp. 371–381, 2016.

[65] C. Wei, I. Chiu, Turning telecommunication call details to churn prediction :a data mining Approach expert System with applications (2002).

[66] S. M. Keaveney, "Customer switching behavior in service industries: An exploratory study," J. Mark., vol. 59, no. 2, pp. 71–82, 1995.

[67] B. Padmanabhan, A. Hevner, C. Michael, and S. Crystal, "From information to operations: Service quality and customer retention," ACM Trans. Manag. Inf. Syst., vol. 2, no. 4, 2011.

[68] J. Bloemer, K. de Ruyter, and P. Peeters, "Investigating drivers of bank loyalty: the complex relationship between image, service quality and satisfaction," Int. J. bank Mark., no. 16, pp. 276–286, 1998.

[69] S. A. Qureshi, A. S. Rehman, A. M. Qamar, A. Kamal, and A. Rehman, "Telecommunication subscribers' churn prediction model using machine learning," in Eighth International Conference on Digital Information Management (ICDIM 2013), 2013, pp. 131–136.

[70] S.Kirkpatrick, C.D.Gelatt, and M.P.Vecchi, Science 220, 671 (1983).

[71] J.H. Holland, Adaptation in Natural and Artificial Systems, Univ. of Michigan Press, Ann Arbor, MI. 1975.

[72] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, New York, 1989.

[73] L. Davis (Ed.), Handbook of Genetic Algorithms, Van Nostrand Reinhold, New York, 1991.

[74] Z. Michalewicz, Genetic AlgorithmsData Structures" Evolution Programs, Springer, New York, 1992.

[75] J.L.R. Filho, P.C. Treleaven, C. Alippi, Genetic algorithm programming environments, IEEE Comput. 27 (1994) 28-43.

[76] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Technical report-tr06, Erciyes university, engineering faculty, computer engineering department, Tech. Rep., 2005.

[77] R. Y. Nakamura, L. A. Pereira, K. Costa, D. Rodrigues, J. P. Papa, and X.-S. Yang, "Bba: a binary bat algorithm for feature selection," in Graphics, Patterns and Images (SIBGRAPI), 2012 25th SIBGRAPI Conference on. IEEE, 2012, pp. 291–297.

[78] A. Kaveh and N. Farhoudi, "A unified approach to parameter selection in meta-heuristic algorithms for layout optimization," Journal of Constructional Steel Research, vol. 67, no. 10, pp. 1453–1462, 2011.

[79] E. Zorarpacı and S. A. Ozel, "A hybrid approach of differential evolution and artificial bee colony for feature selection," Expert Systems with Applications, vol. 62, pp. 91–103, 2016.

[80] S. Oreski and G. Oreski, "Genetic algorithm-based heuristic for feature selection in credit risk assessment," Expert systems with applications, vol. 41, no. 4, pp. 2052–2064, 2014.

[81] P. Moradi and M. Gholampour, "A hybrid particle swarm optimization for feature subset selection by integrating a novel local search strategy," Applied Soft Computing, vol. 43, pp. 117–130, 2016.

[82] E. Emary, H. M. Zawbaa, and A. E. Hassanien, “Binary ant lion approaches for feature selection,” Neurocomputing, vol. 213, pp. 54–65, 2016.

[83] H. M. Zawbaa, E. Emary, and B. Parv, “Feature selection based on antlion optimization algorithm,” in Complex Systems (WCCS), 2015 Third World Conference on. IEEE, 2015, pp. 1–7.

[84] H. B. Nguyen, B. Xue, I. Liu, and M. Zhang, “Filter based backward elimination in wrapper based pso for feature selection in classification,” in Evolutionary Computation (CEC), 2014 IEEE Congress on. IEEE, 2014, pp. 3111–3118.

[85] R. M. Aiex, S. L. Martins, C. C. Ribeiro, and N. R. Rodriguez. Cooperation multi-thread parallel tabu search with an application to circuit partitioning. In Solving Irregularly Structured Problems in Parallel, LNCS Vol. 1457 Springer, 1998, pp. 310–331.

[86] T. G. Crainic and M. Gendreau. Cooperative parallel tabu search for capacitated network design. Journal of Heuristics, 8:601–627, 2002.

[87] E.-G. Talbi, Z. Hafidi, and J. M. Geib. A parallel adaptive tabu search approach. Parallel Computing, 24:2003–2019, 1996.

[88] J. Kennedy. Stereotyping: Improving particle swarm performance with cluster analysis. In International Conference on Evolutionary Computation, 2000, pp. 1507–1512.

[89] S. Tongchim and P. Chongstitvatana. Parallel genetic algorithm with parameter adaptation. Information Processing Letters, 82(1):47–54, 2002.

[90] E. Alba, F. Luna, A. J. Nebro, and J. M. Troya. Parallel heterogeneous genetic algorithms for continuous optimization. Parallel Computing, 30:699–719, 2004

[91] D. J. Ram, T. H. Sreenivas, and K. G. Subramaniam. Parallel simulated annealing algorithms. Journal of Parallel and Distributed Computing, 37:207–212, 1996.

[92] Gao, Yue-lin, Xiao-hui An, and Jun-min Liu. ”A particle swarm optimization algorithm with logarithm decreasing inertia weight and chaos mutation.” In 2008 International Conference on Computational Intelligence and Security, vol. 1, pp. 61-65. IEEE, 2008.

[93] Hu, Hongping, Yanping Bai, and Ting Xu. "Improved whale optimization algorithms based on inertia weights and theirs applications." International journal of circuits, systems and signal processing 11 (2017): 12-26.

[94] Kentzoglanakis, Kyriakos, and Matthew Poole. "Particle swarm optimization with an oscillating inertia weight." In GECCO'09-11th Annual conference on Genetic and evolutionary computation. 2009.

[95] Chen, Guimin, Xinbo Huang, Jianyuan Jia, and Zhengfeng Min. "Natural exponential inertia weight strategy in particle swarm optimization." In 2006 6th World Congress on Intelligent Control and Automation, vol. 1, pp. 3672-3675. IEEE, 2006.

[96] Mafarja, Majdi, Radi Jarrar, Sobhi Ahmad, and Ahmed A. Abusnaina. "Feature selection using binary particle swarm optimization with time varying inertia weight strategies." In Proceedings of the 2nd International Conference on Future Networks and Distributed Systems, p. 18. ACM, 2018.