

# Improving Binary Optimization Algorithms Using Genuine Uniform Initialization

by

Sevda Ebrahimi

A thesis submitted to the  
School of Graduate and Postdoctoral Studies  
in partial fulfillment of the requirements for the degree of

**Master of Applied Science in Electrical and Computer  
Engineering**

Faculty of Engineering and Applied Sciences  
Ontario Tech University  
Oshawa, Ontario, Canada  
December 2023

© Sevda Ebrahimi, 2023

## CERTIFICATE OF APPROVAL

Submitted by Sevda Ebrahimi

In partial fulfillment of the requirements for the degree of

### Master of Applied Science in Electrical and Computer Engineering

Date of Defence: December 11, 2023

Thesis title: Improving Binary Optimization Algorithms Using Genuine Uniform Initialization

The undersigned certify that the student has presented their thesis, that the thesis is acceptable in form and content and that a satisfactory knowledge of the field covered by the thesis was demonstrated by the candidate through an oral examination. They recommend this thesis to the School of Graduate and Postdoctoral Studies for acceptance.

#### Examining Committee:

Shahram

Shahbazpanahi

Digitally signed by Shahram Shahbazpanahi  
DN: cn=Shahram Shahbazpanahi, o=Ontario Tech University,  
ou=Department of Electrical and Computer Engineering,  
email=shahram.shahbazpanahi@utoronto.ca, c=CA  
Date: 2023.12.14 12:16:29 -0500

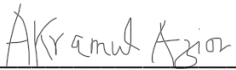
Dr. Shahram Shahbaz Panahi  
Chair of Examining Committee



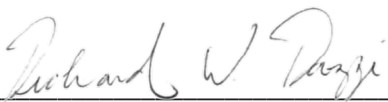
Dr. Masoud Makrehchi  
Research Supervisor



Dr. Shahryar Rahnamayan  
Co-Research Supervisor



Dr. Akramul Azim  
Examining Committee Member




Dr. Richard Pazzi  
Thesis Examiner, Ontario Tech University

- As research supervisor for the above student, the thesis was rendered acceptable without revisions.

\_\_\_\_\_ Dr. Masoud Makrehchi

- As research supervisor for the above student, I read and approved the changes required by the final examiners and recommend the thesis for acceptance:

 \_\_\_\_\_ Dr. Masoud Makrehchi

# Abstract

Population-based metaheuristic algorithms play a crucial role in solving complex optimization problems. The effectiveness of these algorithms is significantly influenced by the initial population of candidate solutions. This thesis investigates the critical aspect of initialization in population-based metaheuristic algorithms. This research studies Uniform Covering (UC) binary initialization method as the substitute for the Bit-string Uniform (BU) binary population initialization method for population initialization step in binary optimization algorithms. BU is the most commonly used random binary population initialization method in the literature, however, this research uncovers the adverse impact of employing this approach on binary optimization algorithms. Study in this thesis reveals that UC method is capable of providing gene-wise uniformity and chromosome-wise uniformity simultaneously, however BU method is not capable of providing chromosome-wise uniformity in the population. Monte-Carlo simulation and mathematical proofs are provided to demonstrate the limitations of the BU initialization in providing the diversity and uniformity in population initialization, meanwhile the effectiveness of the UC method is revealed as the alternative method, aiming to enhance algorithm convergence, robustness, and solution quality. In order to illustrate the effect of the BU and UC initialization on binary optimization algorithms, several experiments are conducted on single-objective and multi-objective combinatorial optimization problems including feature selection and knapsack problems using GA and NSGA-II algorithms representative of the binary optimization problems and binary optimization algorithms respectively. The experiments outcome confirm that BU initialization drastically degrade the performance of the algorithms and UC initialization is the proper way for the random binary population initialization.

**Keywords:** Binary optimization; Uniform population initialization; Multi-objective optimization; Single-objective optimization; Feature selection; Knapsack problem

# Statement of Contributions

The primary contributions of this thesis are as follows:

1. Detailed study on the prevalent binary population initialization method namely Bit-string Uniform (BU) population initialization in meta-heuristic binary optimization algorithms.
2. Deep Investigation on affects of the BU method regarding the uniformity and diversity of the population with Monte-Carlo simulation.
3. Proposal of Uniform Covering (UC) binary population initialization method as a novel binary population initialization in metaheuristic binary optimization algorithms which should be used as the alternative method of BU initialization.
4. Providing a comprehensive experiments to show the superiority of proposed UC method to the traditional BU method in single-objective and multi-objective versions of feature selection and knapsack problems.

## Author's Declaration

I hereby declare that this thesis consists of original work which I have authored. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners. I authorize the University of Ontario Institute of Technology to lend this thesis to other institutions or individuals for the purpose of scholarly research. I further authorize Ontario Tech University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research. I understand that my thesis will be made electronically available to the public.

---

Sevda Ebrahimi

## Acknowledgements

I would like to express my deep gratitude to my research supervisors, Dr. Shahryar Rahnamayan and Dr. Masoud Makrehchi, for their unwavering support, guidance, and expertise throughout this research journey. Their mentorship and invaluable insights have been instrumental in shaping the direction of this work that helped me throughout my education at Ontario Tech University. Additionally, I would like to acknowledge Dr. Azam Asilian Bidgoli whose research in evolutionary algorithms and feature selection was a great help and guidance for this research. Lastly, I would like to express my gratitude to my family and beloved husband Rasul, for providing me with support and encouragement throughout my academic study journey.

# Contents

<b>Certificate of Approval</b>	<b>ii</b>
<b>Abstract</b>	<b>iii</b>
<b>Statement of Contributions</b>	<b>iv</b>
<b>Dedication</b>	<b>v</b>
<b>Acknowledgment</b>	<b>vi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	3
1.2 Objectives . . . . .	4
1.3 Thesis Outline . . . . .	6
<b>2 Background Review</b>	<b>7</b>
2.1 Literature Review . . . . .	7
2.2 Evolutionary Feature Selection Algorithms . . . . .	11
2.2.1 Single-objective Feature Selection . . . . .	13
2.2.2 Multi-objective Feature Selection . . . . .	17
2.3 0-1 Knapsack problem . . . . .	20
2.3.1 Single-objective 0-1 Knapsack Problem . . . . .	20
2.3.2 Multi-objective 0-1 Knapsack Problem . . . . .	21
2.4 Summary . . . . .	22

<b>3</b>	<b>Proposed Initialization for Population-based Binary Optimization Algorithms</b>	<b>24</b>
3.1	Introduction . . . . .	24
3.2	Uniform Covering Initialization . . . . .	24
3.3	Mathematical and Monte-Carlo based Investigation . . . . .	29
3.4	Summary . . . . .	38
<b>4</b>	<b>Experimental Results and Analysis</b>	<b>40</b>
4.1	Introduction . . . . .	40
4.2	Multi-objective Problems . . . . .	41
4.2.1	Multi-objective Feature Selection . . . . .	41
4.2.2	Multi-objective 0-1 Knapsack Problem . . . . .	50
4.3	Single-objective Problems . . . . .	58
4.3.1	Single-objective Feature Selection . . . . .	58
4.3.2	Single-objective 0-1 Knapsack Problem . . . . .	60
4.4	Summary . . . . .	64
<b>5</b>	<b>Conclusion and Future Direction</b>	<b>66</b>
5.1	Conclusion Remarks . . . . .	66
5.2	Future Direction . . . . .	68
	<b>Bibliography</b>	<b>69</b>
	<b>Appendices</b>	<b>76</b>
	<b>Appendix A Optimization Algorithms</b>	<b>77</b>
A.1	Genetic Algorithm (GA) . . . . .	78
A.2	Non-dominated Sorting Genetic Algorithm II (NSGA-II) . . . . .	79



# List of Tables

4.1	Detailed information about used datasets in multi-objective feature selection . . . . .	43
4.2	Mean value of minimum errors for 31 runs and the corresponding average number of features for the solution with minimum error in BU and UC method. Results are compared based on t-test with p-value of 0.05. . . . .	48
4.3	The average ranking score for Pareto front solutions regarding first objective function ( $f_1$ ), second objective function ( $f_2$ ) and in general, average score of two objective functions defined as Ave-rank score for BU and UC method with different number of items including 50, 1000, and 5000. . . . .	53
4.4	Detailed information about used datasets in single-objective feature selection . . . . .	59
4.5	Mean and variance of best solution (minimum error for KNN classifier) over 31 runs for the datasets . . . . .	59

# List of Figures

2.1	General Framework for Evolutionary Algorithm. . . . .	9
2.2	The GA-based feature selection flowchart. . . . .	14

3.1	If the gene-wise uniformity is fulfilled for an specific allele (gene index in a chromosome), the counts of the number of ones and zeros that have appeared in the population must be almost in equilibrium. For population of size $N$ which gene-wise uniformity is fulfilled, total number of ones that appeared in each allele must be almost $N/2$ , obviously the rest of the other $N/2$ are initialized with zeros. . . . .	25
3.2	If the chromosome-wise uniformity is fulfilled for a population which its chromosomes have $M$ dimensions, the frequency of seeing each number in the set including sum of ones for each chromosome must be almost the same. For population of size $M$ which chromosome-wise uniformity is fulfilled, distribution of the numbers of $\{0, \dots, M\}$ is uniform. . . . .	26
3.3	For BU initialization method, Bernoulli probability distribution with $p = 0.5$ is used equally for the bitstrings/chromosomes of the population to fill them with 1/0. . . . .	27
3.4	In UC initialization method, for initializing each chromosome a separate probability distribution will be chosen to fill the bitstrings. The $p$ parameter is selected randomly and uniformly in $[0,1]$ , then each bit will be defined 1 based on probability of $p$ . . . . .	28
3.5	Distribution of genes in a population with 1000 individuals and 80 genes initialized with UC method (a) and BU method (b). . . . .	31
3.6	Distribution of chromosomes with total number of ones in a population with 1000 individuals and 80 genes initialized with UC method (a) and BU method (b). . . . .	32

3.7	A 4-dimensional hypercube as a 4-dimensional binary search space with its total candidate solutions represented as vertices. Red vertices in BU method have higher chance in comparison to other vertices to participate in initial population and the initial population is biased to this specific region of the search space. . . . .	33
3.8	Ratio of accessible search space of BU method to UC method. Total available search space in BU method is approximately $C(n, \frac{n}{2})$ , where the available search space for UC method is $2^n$ . . . . .	35
3.9	Histogram of pair-wise Hamming distances in BU initialization (a) and UC initialization (b) for population size of $N = 100$ and dimension of $n = 100$ . . . . .	38
4.1	Pareto fronts (31 runs) of Madelon dataset after (a) 200 iterations, (b) 300 iterations, (c) 500 iterations. . . . .	45
4.2	Final Pareto front after non-dominated sorting of total points in objective space for Madelon dataset . . . . .	46
4.3	Pareto fronts (31 runs) of TOX-171 after (a) 50 iterations, (b) 200 iterations, (c) 400 iterations. . . . .	47
4.4	Final Pareto front after non-dominated sorting of total points in objective space for TOX-171 dataset . . . . .	48
4.5	Pareto fronts (31 runs) of Arcene dataset after (a) 50 iterations, (b) 100 iterations, (c) 250 iterations. . . . .	49
4.6	Final Pareto front after non-dominated sorting of total points in objective space for Arcene dataset . . . . .	50
4.7	Pareto fronts distribution (31 runs) in the objective space including: 1-total profit of the items inside the knapsack and 2-number of the items inside the knapsack for (a) 15 iterations, (b) 25 iterations, (c) 60 iterations with 50 population size, for knapsack problem with $n=50$ . . . . .	54

4.8	Pareto fronts distribution (31 runs) in the objective space including: 1- total profit of the items inside the knapsack and 2-number of the items inside the knapsack for (a) 200 iterations, (b) 400 iterations, (c) 600 iterations with 100 population size, for knapsack problem with $n=1000$ . .	56
4.9	Pareto fronts distribution (31 runs) in the objective space including: 1- total profit of the items inside the knapsack and 2-number of the items inside the knapsack for (a) 1000 iterations, (b) 1500 iterations, (c) 2000 iterations with 100 population size, for knapsack problem with $n=5000$ .	57
4.10	Performance plot of GA algorithm with BU and UC initialization in solving single-objective knapsack problem for $n=1000$ and $n=5000$ (a) and (C). In the knapsack problem with $n=1000$ and $n=5000$ , the optimization is stopped after 300 and 3000 iterations respectively. (b) and (d) present the minimum constraint violation (average of 31 runs)in the population over function evaluation. . . . .	62

## List of Abbreviations and Symbols

**AC** Ant Colony

**ABC** Artificial Bee Colony

**ALO** Ant Lion Optimizer

**BU** Bit-string Uniform

**Customer Relationship Management** CRM

**DE** Differential Evolution

**EA** Evolutionary Algorithm

**FS** Feature Selection

**GA** Genetic Algorithm

**MOACO** Multi-Objective Ant Colony Optimization

**MODE** Multi-Objective Differential Evolution

**MOGA** Multi-Objective Genetic Algorithm

**MOPSO** Multi-Objective Particle Swarm Optimization

**NSGA** Non-dominated Sorting Genetic Algorithm

**NSGA-II** Non-dominated Sorting Genetic Algorithm-II

**PRNG** Pseudo-Random Number Generators

**PSO** Particle Swarm Optimization

**PSOIn1** Particle Swarm Optimization Initialization 1

**PSOIn2** Particle Swarm Optimization Initialization 2

**PSOIn3** Particle Swarm Optimization Initialization 3

**PSOIn4** Particle Swarm Optimization Initialization 4

**QRNG** Quasi-Random Number Generators

**QRS** Quasi-Random Sequence

**SPEA2** Strength Pareto Evolutionary Algorithm

**UC** Uniform Covering

**UED** Uniform Experimental Design

# Chapter 1

## Introduction

Optimization, as a fundamental task in mathematics and computer science, is the process of finding the best possible solution from a set of feasible alternatives. It plays a crucial role in various real-world applications across diverse domains, such as engineering, economics, machine learning, and etc. The need for improving solutions has become significantly important in the face of complex and large-scale problems that are often encountered in today's data-driven and interconnected world.

Evolutionary algorithms (EAs) as population-based metaheuristic optimization algorithms are a class of optimization techniques known for their remarkable ability to efficiently navigate complex solution spaces and find near-optimal solutions in diverse problem domains [1]. These algorithms, inspired by natural phenomena or heuristic principles, exhibit the power to handle high-dimensional and non-convex optimization problems that traditional exact methods often struggle to solve [2, 3]. Their capacity to efficiently explore solution spaces and offer viable solutions makes them a valuable asset in the toolkit of optimization techniques, particularly for real-world problems where exact solutions may be elusive.

Evolutionary algorithms are composed of various operators and components that gen-

erally drive the search for optimal solutions. These algorithms typically include elements such as population initialization, selection mechanisms, crossover (recombination), mutation, and fitness evaluation. The population initialization phase constructs the foundation by generating an initial set of candidate solutions. Selection mechanisms determine which individuals should contribute to the next generation, often based on their fitness values. Crossover and mutation operations facilitate the exploration of new solutions by combining and altering existing individuals. The fitness evaluation assesses the quality of these solutions according to the optimization criteria. Each of these components plays a crucial role in the algorithm's overall performance and effectiveness, with their interactions organized to achieve the desired balance between exploration and exploitation of the solution space.

The population initialization phase in population-based optimization algorithms has an important role in shaping the trajectory and effectiveness of the entire optimization process. This crucial step establishes the initial individuals of candidate solutions which the algorithm will evolve over successive iterations. The quality and diversity of this initial population significantly influence the algorithm's ability to explore the solution space thoroughly. Well-designed initialization strategies not only help in speeding up convergence but also aid in preventing premature convergence to suboptimal solutions. Moreover, a carefully created population initialization can enhance the algorithm's robustness and adaptability across various problem domains. Approximately, the quality and diversity of the initial population defines the stage for the entire optimization journey. So, emphasizing on the profound importance of this phase is important in achieving successful and efficient optimization outcomes.

Various population initialization techniques have been introduced in the literature. Population initialization techniques, regardless of the type of the problems and algorithms, can be characterized into three categories including: randomness, compositionality, and generality [4]. Regarding randomness, a series of numbers can be perceived as



completely deterministic to truly random and the classification method tries to categorize the population from this point of view. Regarding the compositionality, it is defined as the number of independent procedures engaged in a population initialization method, and finally, generality of the population initializer relates to the variety of the domains that it can be applied to. It is worth mentioning that each categorization criteria characterizes the method from a distinctive and independent perspective. As an example, whether a technique is random or not, is not contingent on whether it is compositional and/or general. This thesis focuses solely on population initialization, particularly from the point of view of randomness and merely in binary population initialization for binary optimization algorithm.

## 1.1 Motivation

Looking at the literature and studies conducted in random population initialization, we can notice that the main concern for the researchers is to generate random numbers using several tools in the population initialization step, which they including the properties of: unpredictability, incompressibility and irregularity. Various methods and tests are introduced for this goal and to provide and measure such properties for these individuals in the population and sequence of numbers. The studies mainly focus on the generating random number with stochastic or deterministic techniques [5]. In most of studies for stochastic techniques the afford is to overcome the disability of deterministic machines (i.e., digital computers) and lack of efficient methods to sample random numbers from physical phenomena (e.g., radioactive decay or atmospheric noise) and provide some algorithm to tackle these challenges. Random number generators including Pseudo-Random Number Generator (PRNG) and Quasi-Random Number Generators (QRNG) are known as the most commonly used population initialization methods in EAs, believing that initialization is simple with these methods and at same time satisfies the uniformity condition in

population [6]. However, this study uncovers that the way (strategy) of filling a string of zero and ones in binary population initialization step has huge effect in uniformity and distribution of the candidate solution in the search space. This thesis provides the mathematical investigation and proof for the above mentioned issue that appears for random binary population initialization methods and simultaneously, suggests an alternative method for random binary initialization which is not paid much attention due to its vanished capability of expanding search space because of the mapping from binary to integer, real, etc., search spaces exist in non-binary optimizations.

## 1.2 Objectives

The performance of population-based metaheuristic optimization algorithms, including EAs, are highly affected by the way its each operator is defined and its parameter settings. Population initialization is a fundamental component of optimization algorithms, and its quality and effectiveness can significantly impact the algorithm's overall performance, efficiency, and ability to find high-quality solutions [7]. However, in binary optimization algorithms, it seems that initialization step has not been investigated and valued as it is needed. Reviewing the existing literature, one can find notable research studies that have been carried out and achieved significant results in population initialization in general, but when it comes to random binary population initialization, the traditional method of standard zero/one equiprobable choice for every bit in each individual of the population is the most commonly used method for initialization and its capability in generating the qualified individuals in binary search space that are representative of candidate solution has not been questioned for binary optimization to the best of our knowledge. In this thesis, we have specifically targeted this method in binary optimization algorithm to see if it really provides the uniformity in the search space and how it affects the performance of the binary optimizers. This method has been introduced in the literature as a popular

method due to its simplicity and uniformity. Only a few studies have discovered the effect of the bias to the specific region of the binary search space with this initialization method, however they have not studied and discovered how this steering of the algorithm toward certain region can affect the optimization process for the binary and non-binary algorithms separately. We have discovered that the bias to the specific region in binary optimization problems kills the uniformity and diversity in the population. Several experiments, besides the mathematical investigation are conducted to confirm the above mentioned studies. We have also considered the NP-hard problems with the possible search space of  $2^n$  (for  $n$  dimension) including evolutionary feature selection problem and knapsack problem in both single-objective version and multi-objective version to be solved using binary Genetic Algorithm (GA) and Non-dominated Sorting Genetic Algorithm-II (NSGA-II) as the case studies to validate our research and investigation.

The main goals of this thesis are as the follows:

1. Detailed study on random binary population initialization methods.
2. Providing mathematical investigation and Monte-Carol simulation for investigating uniformity and diversity of the Bitstring uniform (BU) and Uniform Covering (UC) binary initialization methods.
3. Conducting comprehensive experiments in single-objective, multi-objective optimization algorithms for solving feature selection and knapsack problems that are the representative of combinatorial optimization problems for comparing BU and UC initialization techniques.

The study focuses on using the proper strategy of random binary bitstring generation in binary population initialization for binary optimization algorithms.

## 1.3 Thesis Outline

This thesis comprises of five chapters, that are organized as follows:

**Chapter 2** presents a background review that has relevance to the research, including the concept of binary population initialization, definition and concept of the bit-wise and chromosome-wise uniformity in a binary population. In addition, this chapter provides an introduction and background review of single-objective population-based algorithm, including Genetic Algorithm (GA) and also multi-objective population-based algorithm, including, Non-dominated Sorting Genetic Algorithm II (NSGA-II). In addition, an introduction to single-objective and multi-objective evolutionary feature selection problem and also single-objective and multi-objective knapsack problems as the combinatorial optimization problem is provided. **Chapter 3** proposes an alternative binary population initialization for the most commonly used random binary population initialization in the literature and proves its superiority by providing the mathematical evidence and Monte-Carlo simulation **Chapter 4** investigates the effectiveness of the alternative method through a series of comprehensive experimental analyses on combinatorial optimization problems including feature selection and knapsack problems.

**Chapter 5** includes the conclusion and suggests future research directions.

# Chapter 2

## Background Review

### 2.1 Literature Review

In this chapter, relevant studies to the random binary population initialization are reviewed regardless of the type of algorithm or application. In addition, the concept of uniformity has been described and considered from bit-wise and chromosome-wise uniform distribution point of view.

A binary optimization problem is a type of mathematical optimization problem in which the goal is to find the optimal combination of binary decision variables to maximize or minimize an objective function while satisfying a set of constraints. Binary optimization problems have wide applications in various fields, including computer science, operations research, engineering, finance, and logistics. Some example of application of binary optimization algorithms are:

**Code Optimization in Compilers:** Compilers often perform binary optimizations to generate more efficient machine code from high-level programming languages. This includes techniques like constant folding, loop unrolling, and dead code elimination.

**Resource Allocation in Cloud Computing:** In cloud computing environments, binary optimization can be applied to allocate resources efficiently. This includes assign-

ing virtual machines to physical servers in a way that minimizes energy consumption, maximizes resource utilization, and meets performance requirements.

**Financial Portfolio Optimization:** Investors use binary optimization techniques to optimize their investment portfolios. The goal is to select the best combination of assets that maximizes returns while minimizing risks, considering factors such as volatility, correlation, and historical performance.

**Job Scheduling in Manufacturing:** In manufacturing processes, binary optimization is applied to schedule production jobs on machines. This helps in minimizing production time, reducing idle time, and optimizing the use of manufacturing resources.

**Circuit Design:** Binary optimization is used in electronic circuit design to optimize the placement and routing of components on a chip. This helps in minimizing the physical size of the circuit, reducing power consumption, and improving overall performance.

Solving binary optimization problems can be computationally challenging, as the discrete nature of binary variables often leads to combinatorial explosions. Various algorithms and techniques, such as branch and bound, branch and cut, integer linear programming solvers, and metaheuristic, are used to tackle these problems and find near-optimal or optimal solutions. The choice of method depends on the specific problem and its size and complexity.

Population-based algorithms are a class of optimization techniques that are particularly effective for solving binary optimization problems [8]. These algorithms maintain a population of candidate solutions (often represented as binary strings) and iteratively evolve this population to find better solutions. Among these algorithms, EAs are successfully used for solving binary optimization problems for many decades, as one of the population-based meta-heuristic algorithms [9]. Each EA has its main four components including: population initialization, crossover, mutation, and selection steps. performance of an EA is highly affected by the way each one of its four steps are defined and implemented. Among these steps, despite some of the great researches that have been

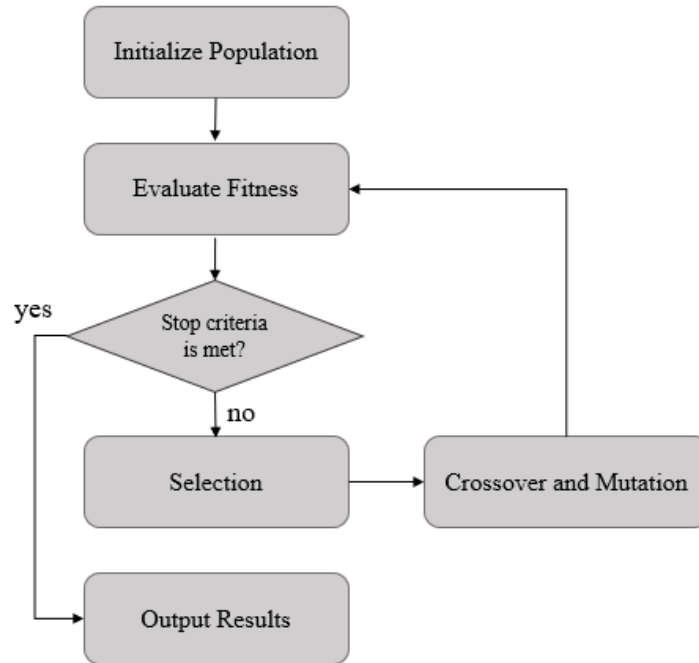


Figure 2.1: General Framework for Evolutionary Algorithm.

conducted and attained great results [4, 10], it appears that initialization step has not been investigated and valued as much as it is needed in binary optimization algorithms. Figure 2.1 presents a a general framework for evolutionary algorithms.

The main goal of the population initialization step that serve as the starting point for the evolutionary process, is to provide a diverse set of candidate solutions that uniformly covers the search space as much as possible. For this goal, generally in EAs, randomness is often introduced in the initialization process to ensure diversity and uniformity among the initial population [11].

The population initialization techniques, can be categorized into three categories based on randomness, compositionality, and generality. Regarding randomness, it considers if a series of numbers can be perceived completely deterministic or random. Regarding compositionality, it is described as the number of independent procedures participated in a population initialization method, and finally for generality, it considers how a population initialization technique can relate to various domain and applications. The last two categories are out of scope of this study, but the effect of randomness on uniformity and

diversity of population particularly in binary optimization is the main concern in this thesis, so we have focused on randomness category. Reviewing the literature in regard to the randomness category, the random population initialization methods are divided into stochastic and deterministic methods [12]. Pseudo-Random Number Generators (PRNGs) and Chaotic Number Generators are two main stochastic techniques that are commonly used for random population initialization specially in EAs. Simultaneously, Uniform Experimental Design (UED) and Quasi-Random Sequence (QRS) are two main divisions for deterministic approaches and again popular for EAs random population initialization in absence of prior knowledge about the problem [4]. Research in this field mainly discuss the uniformity from the point that if the generated numbers are truly random, whether they are statistically/computationally random or not, or providing the tools and test to assess these type of concepts and definitions and mainly talk about the systems itself rather than providing merely a strategy that can focus on producing the random numbers in a way that fulfill the randomness and uniformity in distribution of candidate solutions in a search space. It is believed that randomly generated individuals, regardless of its strategy and application, can help explore a broader portion of the solution space and provide the uniformity for the candidate solutions in the search space. In studies [13, 14], also application of the Gaussian or normal distribution is proposed to have a biased population and scatter the candidate solutions according to these distributions when it is preferred and based on problem's needs. However, in this situation, we encounter a fundamental question: Are these randomly generated individuals in the initial population capable of covering the search space uniformly and do they provide diversity in the initialized population, or uniformity in distribution of the candidate solutions for binary population? If we look at the literature, there are plenty of various and advanced methods and techniques that are introduced to enhance the quality of the candidate solutions in population initialization step in general, however when it comes to binary population initialization, these topics are rarely discussed and to the best of



knowledge the effect of random number generators both systematically and strategically have not been studied. Meanwhile, in binary optimization algorithms, the most popular and commonly used method for random binary population initialization is using the standard 0/1 equiprobable choice for every bit in each chromosome using random number generators formulated as follows:

$$x = \begin{cases} 1 & rand(0,1) \leq 0.5 \\ 0 & rand(0,1) > 0.5 \end{cases} \quad (2.1)$$

Where  $x$  is representative of any bit in any chromosome/bitstring in a population. This method of random binary population initialization is widely used and is the main method for random binary population initialization.

In what follows, we have discussed the algorithms and problems that have been used about initialization methods in binary optimization problems at this thesis. We have used GA and NSGA-II algorithm for solving single-objective and multi-objective feature selection problem and single-objective and multi-objective knapsack problems [15, 16]. Accordingly, these topics are explained for comprehensive understanding of the next chapters.

## 2.2 Evolutionary Feature Selection Algorithms

Feature selection is well-known and critical pre-processing step in data mining and machine learning tasks with various application in real-world such as:

**Medical Diagnosis:** In healthcare, feature selection is crucial for identifying the most relevant biomarkers or clinical variables for diseases.

**Image and Speech Recognition:** In computer vision and speech processing, feature selection helps extract essential information from large datasets.

**Text Classification and Natural Language Processing:** Feature selection is

employed in text mining and NLP applications to identify the most informative words or features in documents.

**Customer Relationship Management (CRM):** Feature selection is applied in CRM systems to identify key customer attributes affecting purchasing behavior and satisfaction.

The main goal in feature selection problems is to select a subset or subsets of the most relevant features (attributes) from a larger set of the features within a dataset. There are quite very rich and abundant studies and researches in the literature in feature selection [17, 18]. Feature selection is considered as a combinatorial optimization problem which can be solved by binary variants of metaheuristic algorithms. There are of course many other methods for solving feature selection problems and each one of them has its own advantages and disadvantages, for an example, exhaustive methods can be used to find the best feature subset(s) [19]. However, the search space of dataset including  $n$  features, is  $2^n - 1$  which its computation time will exponentially increase in case  $n$  is a large number, which these days it is quite common to have such data with huge number of features and also huge number of samples. Greedy search, random search, etc. are other methods that can be used for feature selection problems. In most of the feature selection algorithms, they still have enormous complexity, high computational costs, and they also suffer from premature convergence, so it has become an active field of research and study.

Recently, evolutionary feature selection algorithms have received much attention due to the powerful search ability of evolutionary computation (EC) techniques. EAs demonstrate advantages over traditional methods on solving feature selection problems considering them as both a single-objective problem or a multi-objective problem, requiring less domain specific information. In evolutionary feature selection methods, the performance of a machine model is define as an objective function, and then the evolutionary algorithm tries to improve the performance of the model just by providing smaller number

of features as much as possible by eliminating irrelevant and redundant information.

### 2.2.1 Single-objective Feature Selection

During the initial phase of research on feature selection, it was usually studied as a single objective optimization problem. From this point of view, the optimizer's main goal is to enhance the performance of machine learning model as an objective function. Single-objective feature selection can be mathematically described as follows:

Imagine that dataset  $S$  includes  $d$  number of features. The working mechanism of feature selection algorithm is to select the relevant features from among  $d$  features with respect to the fact that  $J$  is the single objective function to be optimized, which is typically a performance metric (e.g., accuracy, F1-score, or a relevant criterion). Given a dataset  $S = \{f_1, f_2, f_3, \dots, f_d\}$ , the objective is to select the best subset from  $S$  for optimizing  $J$ . Let us call the extracted subset  $Q = \{f_1, f_2, f_3, \dots, f_n\}$  where  $n < d$ , and  $f_1, f_2, f_3, \dots$  represent the features of any subset extracted. Accordingly, the objective is to find best  $Q$  which gets the fittest value for  $J(Q)$ .

Solving feature selection problem with EAs, the characteristics of these algorithms remains the same including: population initialization, crossover, mutation and selection, and apparently, there will be a criteria to stop the search algorithm from proceeding at a right time. For binary optimization algorithm, binary population initialization will take place by defining one as presence of a corresponding feature and zero as absence of the feature. Genetic algorithm (GA) [20], ant colony optimization (ACO) [21], differential evolution (DE) [22], artificial bee colony (ABC) [23], particle swarm optimization (PSO) [24] are commonly used population-based metaheuristic algorithms to solve the single objective feature selection problems. Both binary and continuous versions of these algorithms such as DE and PSO are applicable for solving the feature selection problems, a good example could be continuous version of PSO that has been introduced for feature selection problems that uses a threshold value to choose a particular feature using

a continues representation, however the continues versions are out of scope of this study, but some of these studies has been presented in the following when different population initialization techniques for feature selection are discussed.

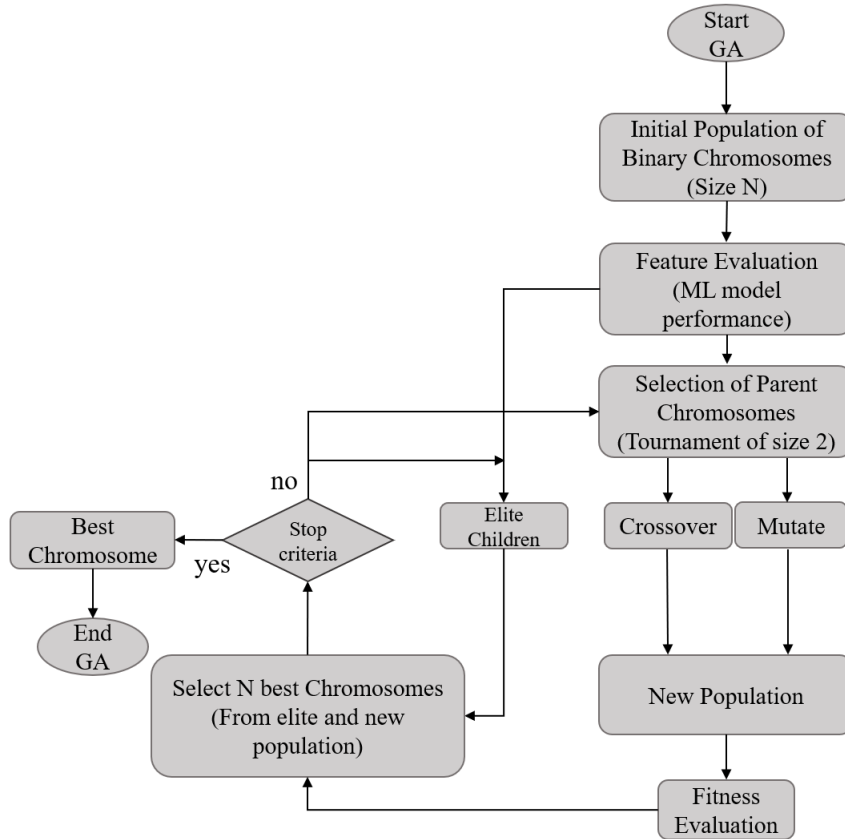


Figure 2.2: The GA-based feature selection flowchart.

### Genetic Algorithm:

In this thesis, we have use binary GA as the optimizer for single-objective feature selection problem. GA-based feature selection algorithm has five important components including: chromosome encoding, population initialization, fitness evaluation, genetic operators including (crossover and mutation, selection) and criteria to stop the GA. Figure 2.2 shows the flowchart of a GA-based feature selection algorithm with some modification from [25, 26].

A subset of features, is defined as a chromosome in the population that is representative of presence or absence of features by assigning 1 or zero to the coresponding feature

within the chromosome respectively. Again for feature selection problem, in population initialization step, as previously mentioned, the prevalent binary population initialization is most of the time with BU initialization method in almost most of these algorithms including the GA.

In fitness evaluation step, each feature subset's quality is assessed using a fitness function tied to the performance of a machine learning model. Subsets that contribute to better model performance receive higher fitness scores.

Genetic operator are responsible for generating offspring for continuing the evolutionary process. In crossover operation, selected feature subsets undergo crossover operations, wherein attributes from two or more subsets are combined to create new subsets. Crossover introduces diversity into the population and can lead to the discovery of improved feature combinations. There are various type of crossover in the literature that can be applied to the algorithm such as single point crossover, two point crossover, uniform crossover, etc. that can be used in GA. In mutation operation, some feature subsets may undergo mutation, introducing small, random changes to individual features. It means that that an arbitrary bit(s) in a chromosome will be flipped from its initial (1 to 0 and vice versa) state based on the mutation coefficient. Mutation helps explore new areas of the search space, preventing the algorithm from getting stuck in local optima. Finally, selection step selects the feature subsets based on their fitness value for reproduction.

And in the end, with defining a criterion, the process continues over multiple generations until the termination criterion is met, such as a maximum number of generations or convergence to a satisfactory solution.

### **Initialization for Feature Selection**

BU initialization is the prevalent binary population initialization method in GA-based feature selection method and of course in many other algorithms with binary population initialization. A few studies have proposed methods to improve the quality of the population generated with BU initialization method in feature selection that are mentioned in

what follows. In [27] preserving non-necessary number of the features even after performing GA-based feature selection have been investigated when using the BU initialization method. Author in this study, proposes that setting a constrain on maximum number of total features in feature subset would be beneficial to the algorithm. In [28] three initialization methods have been proposed namely PSOn1, PSOn2 and, PSOn3 using small initialization, large initialization and the mixed initialization. Small initialization, initializes each particle using a small number of features and large initialization, initializes each particle using a large number of features. Small and large initialization schemes in this paper are inspired by forward selection and backward selection respectively. In mixed initialization, which is a combination of small and large initialization, some particles are initialised using a small number of features and rest of the particles are initialised using large feature subsets. Another study, proposed a new initialization method for PSO algorithm in feature selection problem which is called hybrid initialization strategy [29]. In this method, which is introduced as PSOn4 population initialization method, half of the particles start the process of initialization with small initialization and remaining with the large initialization. In [30], an unsupervised feature selection mechanism is proposed for feature selection problem which instead of predefined probability distribution such as Bernoulli probability, a function based on feature score is introduced same as the probability distribution for generating 0 and 1 representative of absence or presence of a feature. The segmented initialization mechanism is an other method of population initialization for feature selection problems which uses the Bernoulli probability distribution with three different  $p$  values including 0.25, 0.5, 0.75 for binary population initialization that divides total population into three section and initialize each section with corresponding  $p$  value [31]. In [32], a function is introduced for generation of 0 and 1 bitstring in multi-objective gray wolf optimization feature selection based on the fact that for most dataset optimal number of features are less than half of the total features. So, a function similar to previously mentioned probability functions with  $p$  selected from  $\{0.1,$

0.2, 0.3, 0.4, 0.5} and feature positioning based on population factor strategy is proposed. In [33], the method of small, medium, and large initialization has been applied to binary ant lion optimizer (ALO) for feature selection problem and has achieved improvements in feature selection problem. Another study [34] a Gaussian probability density function is introduced for 0 and 1 bitstring generation in feature selection problem with considering the fact that the parameters of the Gaussian function is generated using the feature score and mutual information. In [35] A multi-objective evolutionary algorithm is introduced for large-scale Feature Selection (FS), which utilizes an interval-based initialization approach and a self-adjusting crossover operator. The novel interval-based initialization technique restricts the number of selected features in a solution to enhance the diversity of the initial population within the search space, and also reduces the similarity of the initial population within the decision space. The key procedures involve establishing different intervals based on the number of selected features and evaluating the similarity of each interval solution using Jaccard similarity. Subsequently, intervals exhibiting high similarity will be initialized with a reduced number of solutions.

### 2.2.2 Multi-objective Feature Selection

Feature selection problems, inherently have characteristics of multi-objective optimization problems with two conflicting objectives. The main two conflicting objectives comprises of:

- Maximizing the performance of the machine learning model
- Minimizing the size of the feature subset to overcome the curse of dimensionality

at the same time. Multi-objective feature selection can be mathematically explained as follows:

Imagine that dataset  $S$  includes  $d$  number of features. The working mechanism of feature selection algorithm is to select the relevant features from among  $d$  features with

respect to the fact that  $\mathbf{J}=\{J_1, J_2, \dots, J_m\}$  that is the set of objective functions, to be optimized. Let us have a dataset  $S = \{f_1, f_2, f_3, \dots, f_d\}$ , to be a binary vector of length  $d$ , where  $f_i$  is 1 if feature  $i$  is selected and 0 if it is not selected, let us name the extracted subset  $Q = \{f_1, f_2, f_3, \dots, f_n\}$  where  $n < d$ . Considering the above mentioned two main objectives,  $J_1$  would be the performance of a machine learning model to be maximized, and  $J_2$  would be the number of members in subset  $Q$  to be minimized at the same time.

The challenges and difficulties emerged for solving multi-objective feature selection algorithms can be successfully tackled using multi-objective evolutionary algorithms.

There are various evolutionary multi-objective optimization algorithms in the literature that can be used for solving multi-objective feature selection problems. These algorithms aim to find Pareto-optimal solutions, which represent trade-offs between multiple conflicting objectives and despite the single-objective optimization algorithms which provide one solution, they provide a set of solutions that can be chosen based on the preference. Here is a list of some popular evolutionary multi-objective optimization algorithms for multi-objective feature selection:

Non-Dominated Sorting Genetic Algorithm (NSGA-II): NSGA-II is a widely used multi-objective optimization algorithm [36]. It employs non-dominated sorting and crowding distance to maintain diversity in the Pareto front.

Multi-Objective Particle Swarm Optimization (MOPSO): MOPSO extends the concept of particle swarm optimization to solve multi-objective problems [37]. It uses particles to explore the search space and maintain a diverse set of solutions.

Strength Pareto Evolutionary Algorithm (SPEA2): SPEA2 is another popular algorithm that combines an external archive with elitism and tournament selection to find Pareto-optimal solutions [38].

Multi-Objective Genetic Algorithm (MOGA): MOGA applies genetic algorithms to multi-objective problems [39]. It uses techniques like Pareto dominance and fitness sharing to evolve diverse and high-quality solutions.



Multi-Objective Differential Evolution (MODE): MODE adapts the differential evolution algorithm to handle multi-objective optimization problems [40]. It uses mutation, crossover, and selection to find Pareto-optimal solutions.

Multi-Objective Ant Colony Optimization (MOACO): MOACO extends ant colony optimization to multi-objective optimization [41]. It employs multiple pheromone matrices to guide ants in the search for Pareto-optimal solutions.

Hybrid Algorithms: Many researchers have developed hybrid algorithms that combine elements from different evolutionary multi-objective optimization methods to enhance their performance on specific feature selection problems [18].

These algorithms offer various approaches and trade-offs in terms of exploration and exploitation of the Pareto front, diversity maintenance, and convergence speed. The choice of algorithm often depends on the specific characteristics of the multi-objective feature selection problem and the goals of the optimization process.

### **NSGA-II algorithm:**

In this thesis, NSGA-II algorithm is used for solving multi-objective feature selection problem and multi-objective knapsack problem, consequently for comprehensive understanding of this thesis, it has been explained in detail in what follows. The NSGA-II is a powerful multi-objective optimization algorithm that is widely used for solving problems with multiple conflicting objectives. NSGA-II is an extension of the original Non-dominated Sorting Genetic Algorithm (NSGA), designed to provide improved performance in terms of convergence and diversity maintenance. In NSGA-II algorithm, first, a random binary population is generated. At the same time the initial population is sorted based on nondominated sorting method. Then, for generating the offspring population, the binary tournament selection, recombination and mutation are used. Then parent and offspring population are merged in order to perform the nondomination sorting. The total population is sorted according to nondomination sorting relation, and new population is formed by gathering the solution from the first front and then the next

fronts until the current parent size exceeds the population size. Each individuals will be assigned a fitness value based on the front they belong to. In addition to the fitness value, a new parameter named crowding distance is calculated for each individuals which is a metric for describing how close an individual is close to its neighbors in its specific belonging front. Large average crowding distance value indicates that the individual, when selected in the population will be better for the diversity of the population. Parents are selected from the population based on their front rank and crowding distance using the tournament selection mechanism. Then, the selected population generates the offspring by using the crossover and mutation operators. The population with the current population and current offspring, will again be sorted based on nondomination sorting and best  $N$  individuals are selected which  $N$  is predefined population size by the user (see Appendix A) [42]. NSGA-II is particularly well-suited for solving multi-objective feature selection problems, where the goal is to find a set of features that optimizes multiple objectives, such as classification accuracy, model complexity, and computational efficiency, while considering the trade-offs between these objectives, however, in our study we have considered only two conflicting objectives which they will be explained in detail in following chapters . In the following section, Knapsack problem has been explained for comprehensive understanding of the studies in this thesis.

## **2.3 0-1 Knapsack problem**

### **2.3.1 Single-objective 0-1 Knapsack Problem**

The knapsack problem is a classic optimization problem in computer science, mathematics, and operations research. It's a problem of combinatorial optimization problem with many real-world applications, including resource allocation, portfolio optimization, project scheduling, tourist trip planning, DNA sequencing and genomics, and etc. The 0-1 Knapsack problem as a combinatorial problem has strong similarity with finding

qualified set of features in feature selection algorithms. The multidimensional version of multi-objective 0-1 knapsack problem (MOMKP) is used in this thesis to study the effect of BU and UC initialization. The MOMKP problem is defined as follows [15]:

Imagine we have  $N$  items ( $i = 1, 2, \dots, n$ ) having  $m$  characteristics  $w_j^i (j = 1, 2, \dots, m)$  such as weight, volume,... and  $p$  profits  $c_k^i (k = 1, 2, \dots, p)$ , some items should be selected in a way to maximize the  $p$  total profits while not exceeding the  $m$  knapsack capacities  $W_j$  regarding the various characteristics. The MOMKP problem is formulated as follows:

$$\begin{aligned} \text{Max} \quad & z_k(x) = \sum_{i=1}^n c_k^i x_i && k = 1, \dots, p \\ \text{subject to} \quad & \sum_{i=1}^n w_j^i x_i \leq W_j && j = 1, \dots, m \\ & x_i \in \{0, 1\} && i = 1, \dots, n \end{aligned} \quad (2.2)$$

In single-objective version of 0-1 knapsack problem, the  $k$  is determined 1 and there is only one objective function to maximize. This means that there is only one specific profit assigned to each items regardless of its characteristics including weight, volume, and etc.

### 2.3.2 Multi-objective 0-1 Knapsack Problem

Multi-objective variant can be defined when  $k > 1$ . It means that each item has more than 1 specific profit and consequently there are multiple objective functions to maximize [15]. Again, same as single-objective version, characteristics of each item can be defined independently from number of objectives (profits) for the 0-1 knapsack problem and these characteristics represent the parameters of inequality constraints. The multi-objective version of 0-1 knapsack problem can be defined as follows:

$$\begin{aligned}
& \text{Max} && z_1(x) = \sum_{i=1}^n c_1^i x_i \\
& \text{Max} && z_2(x) = \sum_{i=1}^n c_2^i x_i \\
& \vdots && \vdots \\
& \text{Max} && z_p(x) = \sum_{i=1}^n c_p^i x_i \\
& \text{Min} && q = \sum_{i=1}^n x_i \\
& \text{subject to} && \sum_{i=1}^n w_i x_i \leq W \\
& && x_i \in \{0, 1\} \quad i = 1, \dots, n
\end{aligned} \tag{2.3}$$

In this thesis both version of 0-1 knapsack problem can be solved using the population-based metaheuristic algorithms. These two problems are great candidates in order to investigate the effect of binary population initialization method. So, in this thesis besides the feature selection problem as a combinatorial optimization problem the single-objective knapsack and its multi-objective variant are selected to evaluate our proposed method in the binary population initialization step.

## 2.4 Summary

In this chapter, we discussed the binary population initialization and explained the concepts of gene-wise and chromosome-wise uniformity in a 0/1 bitstrings or chromosomes in the initial population of EAs as the metaheuristic optimization algorithms. Two different population initialization methods including BU and UC, are explained in detail and disadvantages of the BU method as prevalent binary population initialization is uncovered. At the same time the superiority of the UC method is explained simultaneously as an alternative method for the BU technique. We have also explained the NSGA-II and GA algorithms that are used in this thesis. Evolutionary Feature selection problem and 0-1 knapsack problems are also explained since our experiments are conducted on these problems. In the next chapter we have provided mathematical proofs for our investigations on BU and UC

initialization methods.

# Chapter 3

## Proposed Initialization for Population-based Binary Optimization Algorithms

### 3.1 Introduction

In this section, first, mathematical and Monte-Carlo based investigation of BU, bitstring uniform initialization (BU), and uniform covering initialization (UC) in binary optimization are explained. It has been proved that BU initialization is quite the contrary of providing diversity in the initial population. Also, the powerful ability of uniform covering initialization in covering the total search space is unleashed.

### 3.2 Uniform Covering Initialization

First, for a better understanding we should elaborately determine the definition of the uniformity in a binary population. In binary population initialization or bit-string initialization, uniformity of population can be considered from two perspective:

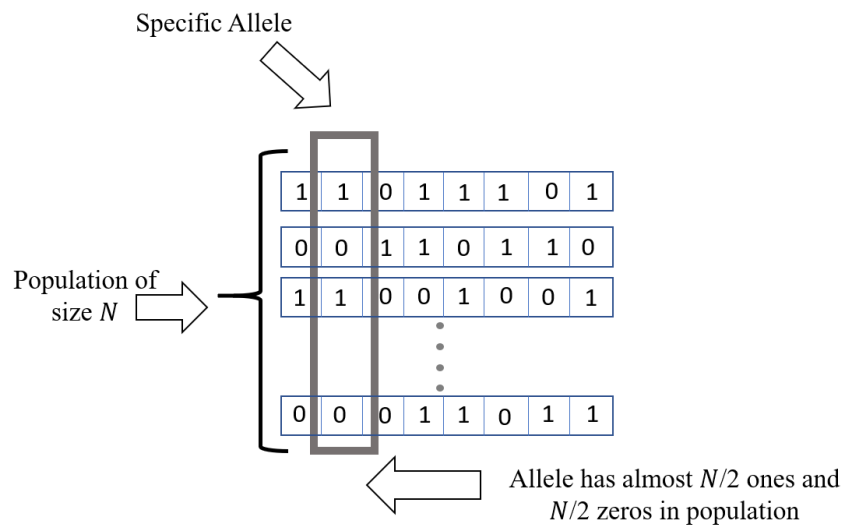


Figure 3.1: If the gene-wise uniformity is fulfilled for an specific allele (gene index in a chromosome), the counts of the number of ones and zeros that have appeared in the population must be almost in equilibrium. For population of size  $N$  which gene-wise uniformity is fulfilled, total number of ones that appeared in each allele must be almost  $N/2$ , obviously the rest of the other  $N/2$  are initialized with zeros.

1. Gene-wise uniformity: making population unbiased to the presence or absence of a specific gene.
2. Chromosome-wise uniformity: making population unbiased to presence of specific numbers of ones inside chromosomes, consequently unbiased to specific regions of the search space and staying diverse in the search space.

Considering the gene-wise uniformity and in doing so, the genes at the same position or index of the chromosomes (alleles), will almost keep same number of zeros and ones in the initialized population, which it means, it makes the population not to be biased to the presence or absence of specific genes. Figure 3.1 shows an specific allele in a population with population size  $N$ , in order to have the gene-wise uniformity total number of ones and zeros in each allele in the population must almost be in equilibrium which means having almost  $N/2$  ones and  $N/2$  zeros.

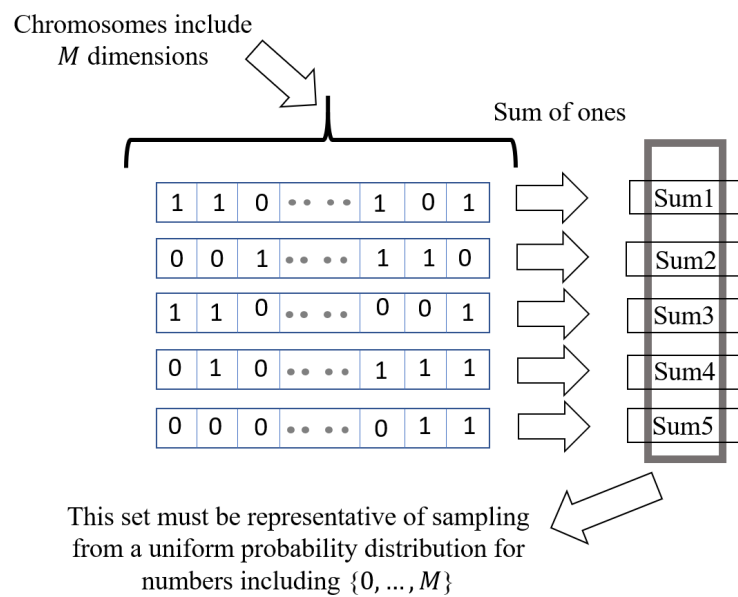


Figure 3.2: If the chromosome-wise uniformity is fulfilled for a population which its chromosomes have  $M$  dimensions, the frequency of seeing each number in the set including sum of ones for each chromosome must be almost the same. For population of size  $M$  which chromosome-wise uniformity is fulfilled, distribution of the numbers of  $\{0, \dots, M\}$  is uniform.



From the other perspective, the chromosome-wise uniformity means that chromosomes in the population, are not biased to have specific number of the genes inside them and there is a uniform probability distribution for total number of the genes inside a chromosome. This fact emphasizes on providing equal chance for each chromosome in the search space to be selected for the initial population. Figure 3.2 illustrates the concept of chromosome-wise uniformity in a population with dimension  $M$ . In order to have chromosome-wise uniformity, the set of sum of total number of ones inside each chromosome which is an integer number between  $[0, M]$ , must have a uniform probability distribution for these numbers from 0 to  $M$ . For example, if each one of the chromosomes has almost equal number of ones and zeros inside them, then  $M/2$  would be the dominant number that can be seen in the set containing sums of ones for chromosomes of population of any size. As previously mentioned, the common method in initialization for the binary space  $\{0, 1\}^n$  of fixed-length- $(n)$  bitstrings is by using “Equation (2.1)”. In what follows, we have introduced a few studies that have targeted the effect of zeros/one bitstring initialization with probability of 0.5, then we have discussed a few methods that are proposed in the evolutionary feature selection algorithms as the alternative methods for this common method of binary population initialization.

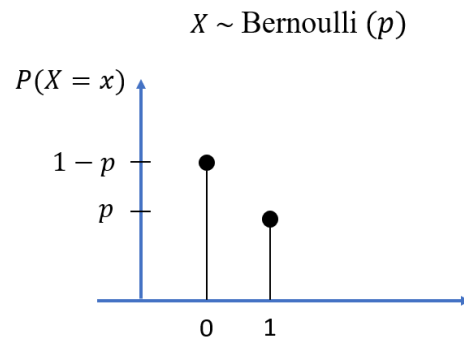


Figure 3.3: For BU initialization method, Bernoulli probability distribution with  $p = 0.5$  is used equally for the bitstrings/chromosomes of the population to fill them with 1/0.

In 1997, leila kallel et al [43], conducted a study on effect of prevalent 0/1 equiprobable

### Binary Optimization Problems

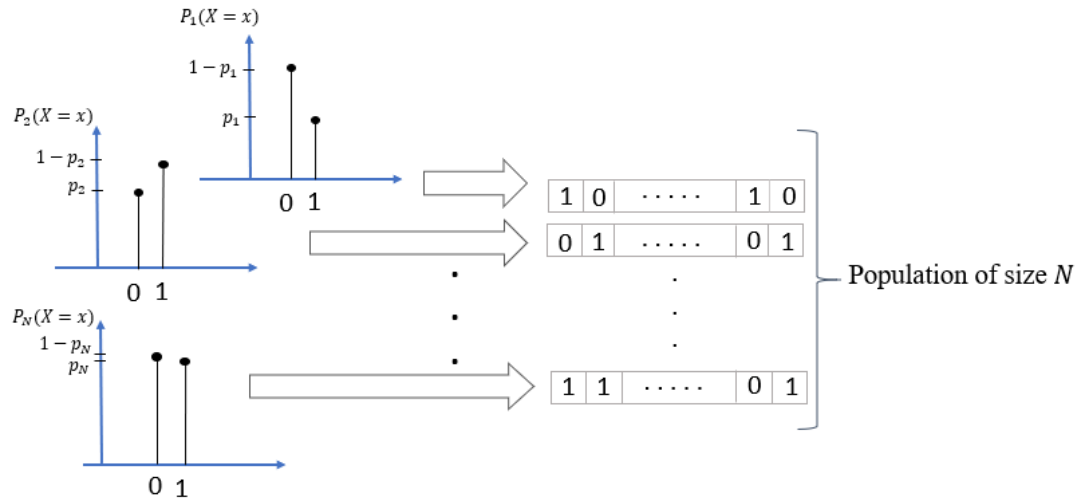


Figure 3.4: In UC initialization method, for initializing each chromosome a separate probability distribution will be chosen to fill the bitstrings. The  $p$  parameter is selected randomly and uniformly in  $[0,1]$ , then each bit will be defined 1 based on probability of  $p$ .

bitstring initialization method and defined it as the bitstring-uniform standard initialization procedure (BU) where assigns 0 or 1 with probability 0.5 to every bit. Figure 3.3 also shows the probability distribution known as Bernoulli distribution with determined  $p = 0.5$  for the case of BU initialization method for filling the bitstrings. The study investigates the initialization particularly in bitstring genetic algorithms and it has been argued that the BU initialization will not guarantee the chromosome-wise uniformity and diversity in population in general binary framework for GA. However, the uniform covering initialization procedure (UC) as a solution for this critical problem has been proposed with focusing on the density of 1's and 0's for chromosomes to be originated from a uniform distribution. Figure 3.4 presents the process of population initialization using UC method.

The procedure for UC method is defined as: for each bitstring or chromosome, select the density of ones represented as  $p$  parameter for Bernoulli distribution from between  $[0,1]$  randomly, and then fill the bitstring with this randomly defined probability distribu-

tion. It is worth mentioning that for BU method the  $p$  parameter for all chromosomes is equally defined as 0.5 but for UC method  $p$  parameter can be take any number between  $[0,1]$  differently for each chromosome.

This concepts have not been investigated for binary optimization problems and its effect on binary optimization algorithms have not been uncovered due to the mapping from binary to real numbers which eliminated the chromosome-wise bias. To the best of our knowledge, BU initialization method with its inability in providing chromosome-wise uniformity is still the dominant method in random binary population initialization. However, looking at the literature, in some specific optimization problems such as feature selection, the community have been attracted to improve the quality of the generated bitstrings which have been generated with BU procedure in the population initialization step, without specifically mentioning the impact of BU initialization on diversity and uniformity in the search space.

### 3.3 Mathematical and Monte-Carlo based Investigation

In this section, the motivation behind this study and the important concepts and mathematics for understanding the effect of prevalent binary initialization method in the literature and its proposed replacement are discussed. First the detailed effect and mechanism of bitsrting formation with commonly used BU method and UC method as its replacement are discussed, then Monte-Carlo simulation of BU initialization is performed for exploring the way it fills the binary search space [44]. Finally, the main reason for underestimating the effect of UC method in the literature have been explained.

For better understanding of BU and UC binary initialization, it has been illustrated in Figure 3.4 the way each chromosome is initialized with zero and one using different probability distribution functions. In BU binary initialization method which is common

method of population initialization in the literature, each chromosome uses the uniform probability distribution on  $\{0, 1\}$  (equally for each individuals in the population), which is exactly the same as filling a bitsring of a chromosome using a fair coin for producing the head as one and tail as zero. Consequently, the probability distribution function is a Bernoulli probability distribution with parameter  $p=0.5$ . However, in UC binary initialization, the scenario is completely different. In this case probability distribution function for filling each chromosome is not equal for the chromosomes (although it might be equal for some chromosomes) in the population, and is generated by assigning a random value for  $p$  which is a selected number(float) with uniform probability from  $[0,1]$ . In this case, the situation is exactly the same as filling a bitsrting of chromosome with an unfair coin or a fair coin (based on random value assigned for  $p$  from  $[0,1]$ ), however each time a new coin with different amount of unfairness is used for generating zero and one bitstring.

As previously mentioned, the concept of uniformity and uniform distribution must be considered from two perspective: Gene-wise uniformity and chromosome-wise uniformity. In Gene-wise uniformity, where the alleles (the genes at the same position or locus) contain almost the equal number of zeros and ones, it is guaranteed that the population is not biased to presence or absence of specific gene(s) inside the population. This means that zero and one would have same probability of 0.5 to fill the alleles in the population and consequently, in a population with  $N$  individuals, for each specific gene or an index which is representative of a gene in a chromosome, almost  $N/2$  of population are filled with ones for this allele.

We have conduct an experiment, to investigate the accomplishment of this gene-wise uniformity conditions in commonly used BU initialization method and UC initialization method in population initialization step. In this experiment, two different population with same number of 1000 individuals are generated with BU initialization and UC initialization method respectively. Each individual also includes 80 genes. Figure 3.5

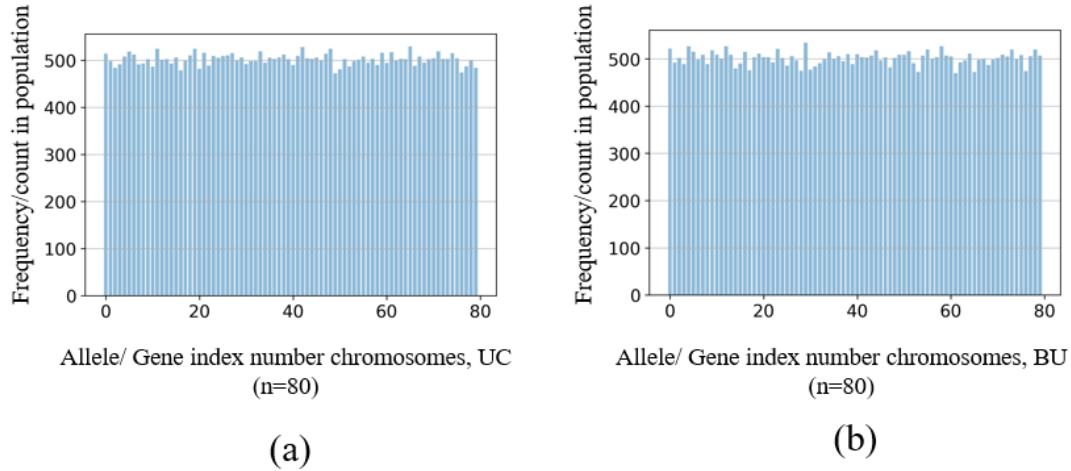


Figure 3.5: Distribution of genes in a population with 1000 individuals and 80 genes initialized with UC method (a) and BU method (b).

shows the histogram (distribution) of the presence of each gene, or index of a gene of the chromosome, in the population for both BU and UC initialization methods. As we can see, both methods comply with this criterion and there is a uniform distribution of different genes in both methods.

Considering the chromosome-wise uniformity, we expect to fill the search space in population initialization step, as uniform as and as diverse as possible, and providing an equal chance for each different permutation of zeros and ones in a bitstring chromosome, to be selected as a potential solution. The fact of providing equal chance for each different permutation, or in other words, generating the candidate solutions from a uniform probability distribution for different permutation is a must for fulfillment of having diversity and uniformity in binary search space. We have conducted another experiment with the same number of population and genes, 1000 and 80 respectively, to investigate if the chromosome-wise condition is accomplished with BU and UC methods. Figure 3.6 shows the histogram of presence of different permutations with equal amount of genes assigned to 1 for BU and UC methods.

As we can see clearly, for BU method, predominant number of individuals include the permutations which have almost half of the genes assigned to one and there are a few

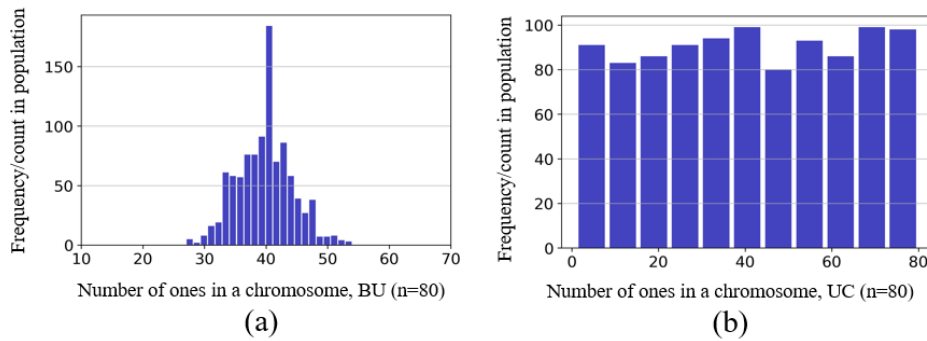


Figure 3.6: Distribution of chromosomes with total number of ones in a population with 1000 individuals and 80 genes initialized with UC method (a) and BU method (b).

other candidates representative of so many other different permutations. The histogram is in this case a ring shape graph which indicated that population have been generated from a Gaussian distribution where its chromosomes which have same amount of zeros and ones, have the highest chance to be generated from this distribution. This type of distribution is quite the contrary of proving equal chance for every possible permutation of different number of zeros and ones, and will indeed drastically degrade the diversity in the population. On the other hand, if we look at the histogram of UC method, permutations with different amount of ones, have almost same frequency of existing in the population which indicated the fact that these candidate solution have been generated from a uniform probability distribution and each and every possible permutation of different number of zeros and ones have exactly the same chance to be in the initial population. So, despite BU method, UC method fulfills the diversity and uniformity in producing the random candidate solution in the search space.

### Mathematical and Monte-Carlo based Investigation:

To prove our investigation and also to see how generation of population using BU method adversely affects its diversity and uniformity in binary search space, consider an  $n$ -dimension binary search space as an  $n$ -dimensional unit hypercube. Imagining vertices of the unit hypercube as potential solutions in the decision variable space, the total

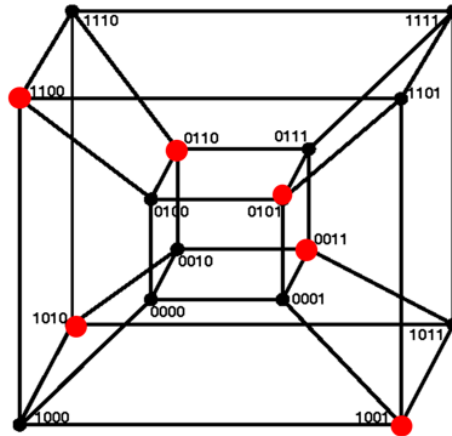


Figure 3.7: A 4-dimensional hypercube as a 4-dimensional binary search space with its total candidate solutions represented as vertices. Red vertices in BU method have higher chance in comparison to other vertices to participate in initial population and the initial population is biased to this specific region of the search space.

number of the solutions is equal to  $2^n$  same as total vertices of an  $n$ -dimensional unit hypercube. It is very important, especially for avoiding the premature convergence and convergence speed of algorithms, to give equal chance to each one of these potential solutions to be able to participate in initializing the population. As, previously mentioned, BU generates candidate solutions randomly from a Gaussian distribution, which is equal to being biased to do sampling from specific region of search space where the number of zeros and ones are almost in balance. For better illustration, Figure 3.7 provides an example of the 4-dimension unit hypercube with its total vertices as the total solutions in 4-dimensional binary search space.

The process of initializing a population of size  $N$ , resembles the process of sampling from a binary search space size  $n$ ,  $N$  times ( $n > N$ ). When sampling each time, a vertex of this  $n$ -dimensional hypercube is introduced as a candidate solution. With BU initialization, however, vertices with equilibrium states (almost same amount of zero and ones) have higher chance in doing the sampling. For example, when initializing with BU

method, vertices which are in red colors will have the maximum probability to be chosen as a potential solution in sampling for population initialization and the other will have lower chance to be selected.

Let us go back to  $n$ -dimensional space and consider the BU initialization method as a sampling process. With BU method, the vertices with almost equal number of zeros and ones (vertices in almost equilibrium states) have the highest chance in sampling for generating the initial population. If we calculate the total number of available permutations that contain almost equal amount of zeros and ones, we can have an estimation of available search space in BU method. Considering the total number of permutations for the situations (vertices) which they have  $\frac{n}{2}$  ones and considering them as quite sufficient representatives of available sampling space in BU, almost the total number of accessible search space can be approximately estimated as:

$$C(n, \frac{n}{2}) = \binom{n}{\frac{n}{2}} = \frac{n!}{(n - \frac{n}{2})! (\frac{n}{2})!} \quad (3.1)$$

$C(n, \frac{n}{2})$  is representative of total different combinations of almost  $\frac{n}{2}$  ones and  $\frac{n}{2}$  zeros inside an  $n$ -dimensional bitsrtring/chromosome. At the same time, we can clearly see that the available search space in UC method is  $2^n$  which means that we have access to all the search space and vertices in sampling for initialization the population. With a simple comparison, we can understand that this condition is equal to shrinking search space from  $2^n$ , to almost the amount of  $C(n, \frac{n}{2})$ , which is a drastic reduction of available points in a search space for sampling, especially in large dimensions. The estimated ratio of accessible search space in BU method to UC method, can be be formulated as:

$$\text{Number of accessible states by BU} = \frac{C(n, \frac{n}{2})}{2^n} = \frac{\frac{n!}{(n - \frac{n}{2})! (\frac{n}{2})!}}{2^n} \quad (3.2)$$

The graph of estimate ratio for variable values for  $n$  also can be seen in Figure 3.8. It is worth mentioning that, we could observe this ratio from other point of view and consider



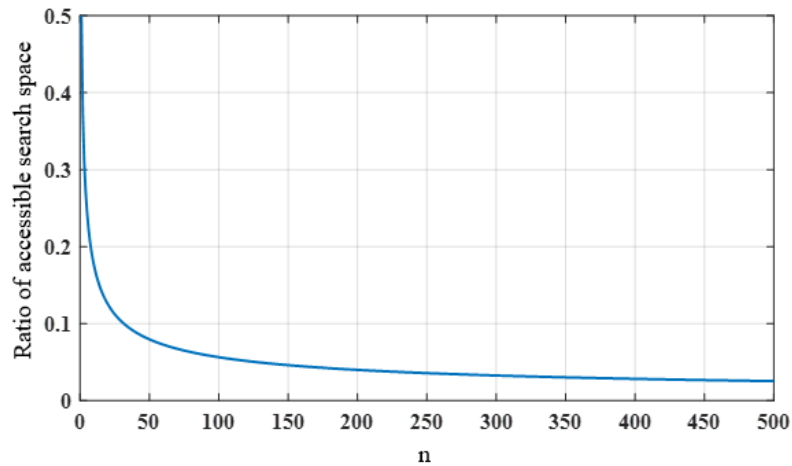


Figure 3.8: Ratio of accessible search space of BU method to UC method. Total available search space in BU method is approximately  $C(n, \frac{n}{2})$ , where the available search space for UC method is  $2^n$ .

it as the percentage of available search space for BU method since the available search space for UC is equal to entire search space. Figure 3.8. shows that the ratio of accessible search spaces from BU to UC decreases drastically in larger values for  $n$ . If we look at graph carefully, the BU initialization method can only sample from 0.025 percentage of the entire search space for almost  $n > 300$ . This fact states a major problem for all type of algorithms that their population initialization is accomplished using BU method, since real-world problems mostly have larger dimensions. Even in smaller dimensions such as  $n=25$ , BU can have access to only 10 percent of the entire search space which is a very small number. This issue becomes more serious in large scale problems when the dimension of problem are  $n > 1000$  and the ratio drops to near zero value. This situation will force the algorithms to start its exploration from a tiny and specific region of the search space, which kills the ability of the algorithms to have access to remaining vast number of potential solutions. It means that whole burden of finding optimal solution has been laid on the optimizer or even worst, the optimizer needs to invest significantly more additional effort to overcome the adverse impact of a poor initialization. This issue

also indicates that, the common belief of increasing number of populations to increase its diversity, has almost zero effect in larger dimensions on the diversification in this case, since the accessible search space has confined already. However, here it comes the question why the effect of such search space shrinkage have not been paid much attention through the years? As previously mentioned, [43] introduced the UC method as a replacement for BU method in population initialization in the GA algorithm. However, based on the results and experiments conducted in the mentioned study, there was no huge difference for the performance of GA algorithm regarding the problem of the huge shrinkage of search space that has been solved by using the UC population initialization. At first glance, one may think that algorithm can overcome the search space shrinkage problem, but quite the contrary, the fact is that UC initialization method illustrates its power and capabilities for binary optimization problems and algorithms. The fact is that because of existing of a encoding step in the algorithm such as binary to real, binary to integer, etc., the issue of shrinkage of space does not exist anymore. Consequently, based on the discovery in this study, for binary optimization problems BU method drastically damages the performance of the algorithm due to the massive search space shrinkage that exist in the population initialization step and it is proven in this thesis that UC method is highly effective when used properly for combinatorial optimization problems with binary initialization rather than BU method.

Another metric for evaluating the quality of the generated binary populations, initialized with BU as prevalent method and UC method, is to calculate the pair-wise Hamming distance between the individuals. The Hamming distance between two binary bitstring or vector is defined as follows: Given two vector  $U = (u_1, u_2, \dots, u_n)$  and  $V = (v_1, v_2, \dots, v_n)$ , the Hamming distance between  $U$  and  $V$  is  $d(U, V)$  that is the number of bit places where  $U$  and  $V$  differ, so can simply be calculated by sum of the positional mismatches of the two bit strings. In other words, Hamming distance is the number of bits that must be changed in order to change one vector to another.

The distribution of measured Hamming distances gives us information about the differences or similarities between pairs of binary strings or sequences, accordingly it can be beneficial for quantifying the amount of existing diversity in a population. We have conducted an experiment to evaluate, measure, and compare the amount of diversity a population can have when generated with BU method and UC method. Two populations with  $N = 100$  as population size, and  $n = 100$  as number of genes are initialized with BU and UC methods for this aim, then the pair-wise Hamming distance between the individuals is calculated. Finally, the distribution of the distances is illustrated by plotting the histogram of distances for both BU and UC method for comparison. In conducted experiment the minimum Hamming distance could be 0 (chromosomes are exactly the same) and maximum Hamming distance could be  $n$  (all bits are completely different in two chromosomes) which is 100 in here. It is worth mentioning that total number of calculated distances for a population of size  $N$  would be  $(N \times \frac{N-1}{2})$ . Figure 3.9 shows the histogram of Hamming distances for BU and UC methods. As we can see, in BU method, the histogram is confined to a specific region and it is narrowed to a specific distance. The narrow histogram of Hamming distance which is confined to a specific distances in this population is indicator of several fact:

- Many strings are nearly identical or share a common pattern.
- The strings are tightly clustered in the search space and belong to the specific subspace of the search space (initialization only targets one specific region in the search space).

Consequently, with BU method the population will not be a representative of the available search space and always confined the search space to a very tiny subspace which they share a similar characteristic which in this case its having almost equal amount of zeros and ones.

In UC method, As we can see the histogram of distances, the distribution of distances

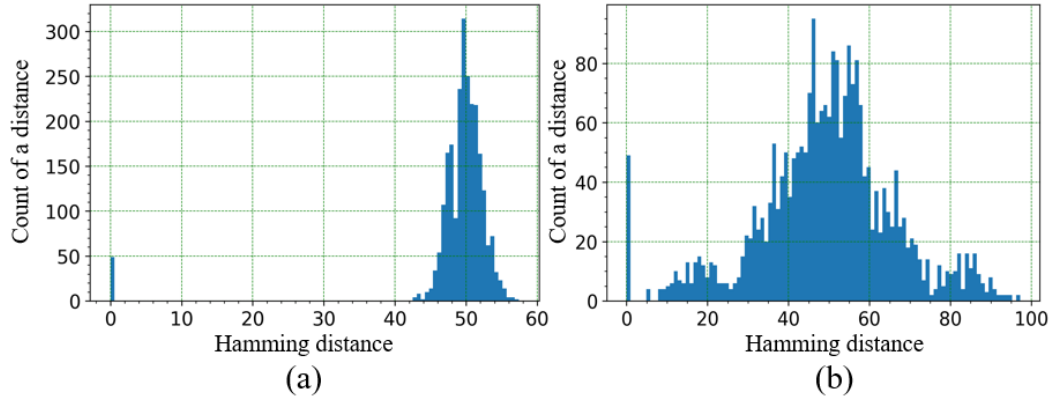


Figure 3.9: Histogram of pair-wise Hamming distances in BU initialization (a) and UC initialization (b) for population size of  $N = 100$  and dimension of  $n = 100$ .

in the population is scattered across various Hamming distance values and includes the smallest (most similar individuals) and largest distance (completely different individuals) that can be defined for a string of 100 bits. The wide-spread histogram indicates that the population has almost quite good samples of all types of available permutations, representative of  $2^n$  total permutations. If we generate all total permutation of  $2^n$ , the histogram for the population would have a Gaussian distribution as a most diverse population that we can ever have, however with UC method, even with very small number of samples ( $N = 100$  sample out of  $2^n$ ), the population still could maintain such diversity in the population. Consequently, this wide-spread distribution indicates that the population contains a more diverse set of Hamming distances, and the data points may exhibit a wider range of binary string patterns and heterogeneity and available search space for sampling is not confined to any specific region.

### 3.4 Summary

In this section, we investigated the gene-wise uniformity and chromosome-wise uniformity for both BU population initialization method and Uc population initialization method.

It has been proved that gene-wise uniformity is fulfilled for both methods, however BU method can not accomplish the chromosome-wise uniformity in a binary population. This investigation is confirmed mathematically and also with Monte-Carlo simulation. Finally, the capabilities of UC initialization method as the alternative method is discussed and investigated. In the next chapter, several experiments including single-objective and multi-objective combinatorial optimization problems are solved using binary optimization algorithms initialized with both BU and UC initialization methods to confirm our investigations.

# Chapter 4

## Experimental Results and Analysis

### 4.1 Introduction

In this chapter, different experiments are conducted to evaluate the effect of BU and UC initialization methods on performance of the binary optimization algorithms including GA and NSGA-II. Both single-objective and multi-objective type of optimization problems have been selected to investigate the effect of BU and UC initialization methods. It is worth mentioning that proper method of population initialization that fulfills the diversity and uniformity is important in single-objective, multi-objective, and many-objective optimization all together, but in this thesis we have focused on single-objective and multi-objective optimization. Single-objective problems are considered for the experiments in order to eliminate the direct effect of the binary initialization that may affect the second objective function in this study which may intensify the effect of search space shrinkage in population initialization step of the optimization algorithm. The selective optimization problems include feature selection problem and 0-1 knapsack problem, that are the representative of combinatorial optimization problems in this thesis. In this study, GA and NSGA-II algorithms are used as the representative of binary optimization algorithms. Each optimization algorithm is initialized with two different population initialization

methods including BU method and UC method, and the performance of each algorithm with two different initialization are compared when initialized differently, to investigate the effect of different initialization. In what follows, first multi-objective version of both feature selection and 0-1 problems are studied and effect of BU and UC initialization are investigated, finally the single-objective feature selection and single-object 0-1 knapsack problems are studied.

## 4.2 Multi-objective Problems

### 4.2.1 Multi-objective Feature Selection

In this section feature selection problem is considered as a multi-objective optimization problem. Feature selection problems have intrinsically the characteristics of multi-objective optimization problems with two conflicting objective functions including:

- Improving the effectiveness of the machine learning model
- Reducing the number of the features

In this thesis, we have considered these two objectives as our objective functions, consequently there is a bi-objective optimization problem.

For implementing the feature selection problem, simple binary coding scheme is used to show presence or absence of a feature in a binary chromosome. Zero indicates the absence of a corresponding feature in a chromosome and one shows its presence. For the first objective function, the K-Nearest Neighbor, KNN classifier is introduced as the most popular machine learning model for feature selection problem [45]. The accuracy score of the KNN classifier is used as a evaluation metric to investigate the effectiveness of the model, however it has been changed to the error, due to the compatibility of the problem with the platform used for conducting the experiment. The NSGA-II algorithm with two different initialization methods including the prevalent BU initialization method and

UC initialization method, are used to discover the best feature subsets with minimum number of the features and with better results for the KNN classifier. The performance of the NSGA-II algorithm with two different initialization methods are compared with each other in order to investigate the effect of BU and UC initialization. For NSGA-II algorithm, the parameter setting is followed based on its original paper [36]. In this thesis, the entire experiments are conducted on open source platform Pymoo [46] including feature selection using NSGA-II algorithm. For both NSGA-II algorithms with different initialization, single-point crossover and bit-flip mutation operators are used to generate the offspring solutions [47, 48]. Control parameters including number of population and value of K in KNN algorithm are set to be 50 and 5 respectively in solving the MOEAs-based feature selection problem. Each optimization algorithm has a limited budget to find the better solutions. In this study, we have defined maximum number of iterations/generations as the stop criterion of the NSGA-II algorithm. A set of different maximum number of iteration are defined to evaluate the performance of the algorithm in its different stages, based on a dataset in general. Average ranking of the Pareto fronts and average number of features for minimum errors in Pareto fronts in feature selection problems respectively have been used for comparison. It is important to mention that number of function evaluations is used in some results of experiments which is defined as population size  $\times$  iteration. Three different datasets are introduced for the feature selection problem and We have run each algorithm on each dataset 31 times in order to avoid the stochasticity in our results.

**Dataset for MOEAs-based feature selection** To show the performance of the NSGA-II algorithm with different initialization on feature selection problems, 2 different datasets are selected from UCI repository and 1 dataset from ASU repository [49]. The datasets include different dimensions (number of features) and different sample sizes. The selected datasets are quite the good example of small and large scale datasets in order to have a comprehensive evaluation on performance of algorithm and its generalization



capability with different initialization. In each experiment also, the ratio of the training dataset to test dataset is set to be 80 to 20. Table 4.1 shows the description of the datasets.

Table 4.1: Detailed information about used datasets in multi-objective feature selection

Dataset	Number of Features	Number of Samples	Number of Classes
Madelon	500	2600	2
TOX-171	5748	171	4
Arcene	10000	900	2

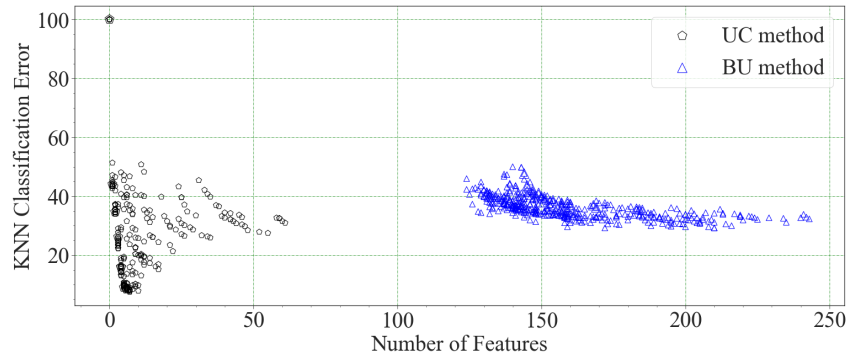
### Experimental Results and Analysis:

In order to assess the performance of the NSGA-II algorithm with different initialization, a series of experiments are conducted on datasets described in Table 4.1. In this section, we analyse the performance of NSGA-II algorithm with BU and UC initialization methods and we have compared them. Besides, vividly, the distribution of the Pareto front solutions in objective spaces is used to compare their performance. The Pareto front solutions of an algorithm comprises of all nondominated solutions for 31 independent runs.

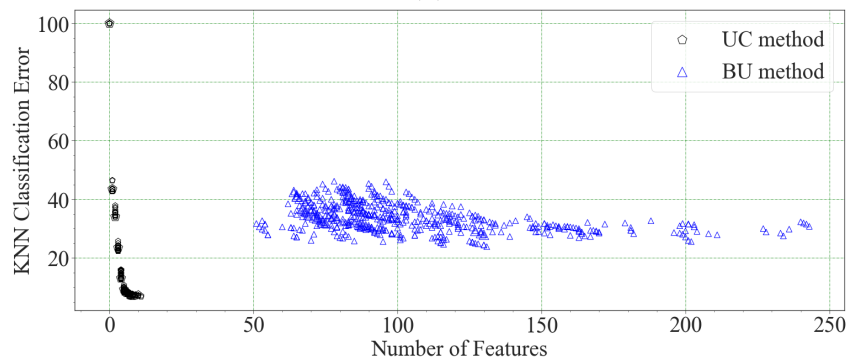
**Results on Madelone dataset:** Figure 4.1 shows the total generated Pareto front solutions for both BU and UC initialization methods with the Madelone dataset after 31 runs. For the Madelone dataset, both algorithms are stopped after 200, 300, 500 iterations in order to capture the evolution of the pareto fronts during the optimization. In Figure 4.1 the total Pareto front solutions are illustrated in (d) and (e) separately for BU and UC method respectively, after 500 iterations. In Figure 4.2, Results on Madelone dataset evidently indicate the superiority of NSGA-II algorithm with UC initialization method since after non-dominated sorting only Pareto front solutions of UC method exist. Pareto fronts distribution graph indicate that NSGA-II algorithm with UC initialization not only start optimization from a better region in the search space, it is also much powerful when it comes to explore the search space for optimal solutions as we can see within 300 iteration NSGA-II with UC initialization is able to reach to least number of features

with higher accuracy (lower error) for KNN model. Meanwhile, after 500 iterations, NSGA-II with BU initialization even can not reach to UC method's discovered Pareto front solutions in iteration number 200. This illustrates that UC method has highly improved the performance of the NSGA-II. Besides, all Pareto fronts points generated with UC method dominate all Pareto fronts points generated using BU method. Table 4.2 presents the average minimum error on Pareto front with 31 runs and its corresponding average number of the features for minimum error on Pareto front. As we can see, UC method achieves the lower error for the classifier with fewer number of the features in comparison to BU method.

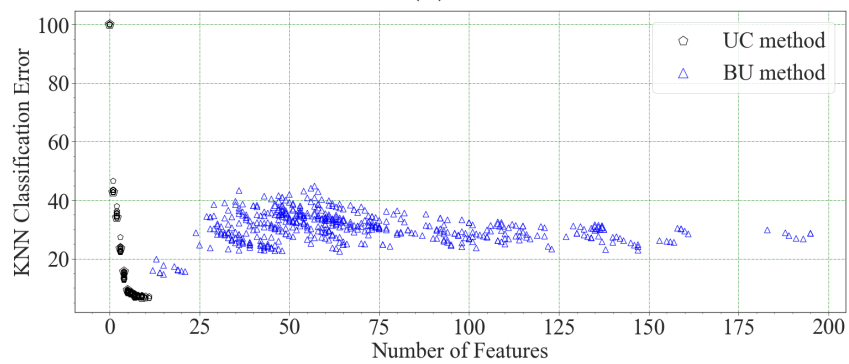
**Results on TOX-171 dataset:** Figure 4.3 presents the total generated Pareto front solutions for both BU and UC initialization same as the Madelon dataset. For the TOX-171 dataset, both algorithms are stopped after 50, 200, 400 iterations in order to capture the evolution of the Pareto fronts during the optimization. In Figure 4.3 the total Pareto front solutions are separately illustrated in (d) and (e) for BU and UC method respectively after 400 iterations. In Figure 4.4, Results on TOX-171 dataset evidently indicate the superiority of NSGA-II algorithm with UC initialization method since after non-dominated sorting only Pareto front solutions of UC method exist and also results of average minimum error and number of features in Table 4.2 confirm the superiority of UC method as well. Pareto fronts distribution graph confirm that NSGA-II algorithm with UC initialization start its optimization from better solutions in the search space due to its effective and wide-spread initialization. Optimizer's empowered exploration and exploitation abilities can be easily noticed from its faster convergence to optimal solution with very fewer features and lower classification error. Within 200 iteration NSGA-II with UC initialization is able to reach to least number of features with higher accuracy (lower error) for KNN model. However, for NSGA-II with BU initialization even after 400 iterations, algorithm is much far away from NSGA-II with UC initialization method's initial search steps and its all Pareto front solutions are dominated by UC Pareto front



(a)



(b)



(c)

Figure 4.1: Pareto fronts (31 runs) of Madelon dataset after (a) 200 iterations, (b) 300 iterations, (c) 500 iterations.

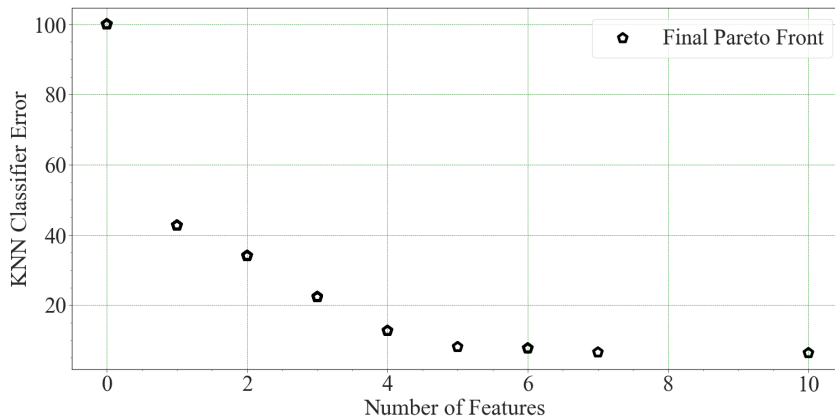
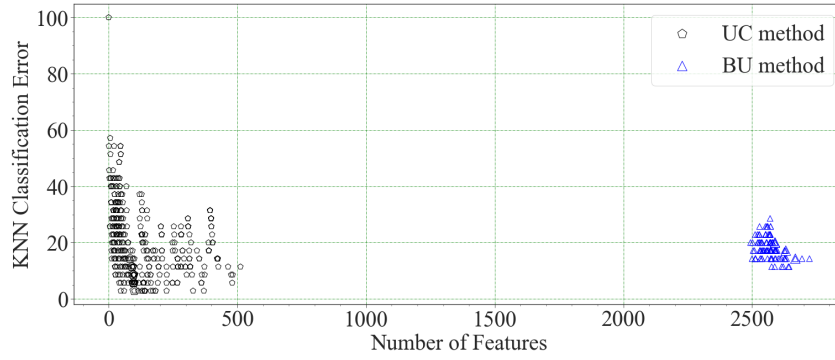


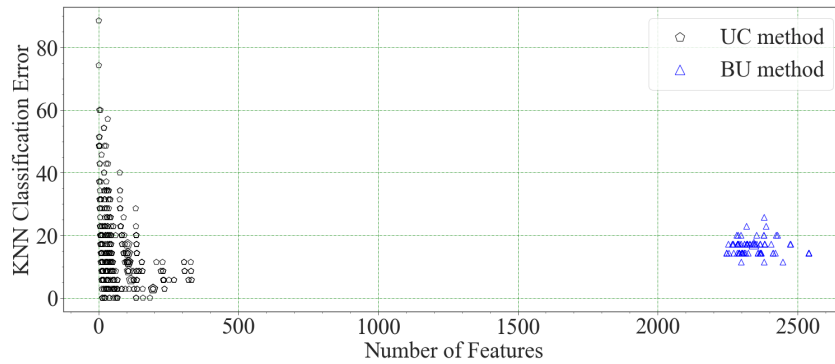
Figure 4.2: Final Pareto front after non-dominated sorting of total points in objective space for Madelon dataset

solutions in all stages. This experiment again illustrates that UC method has highly improved the performance of the NSGA-II.

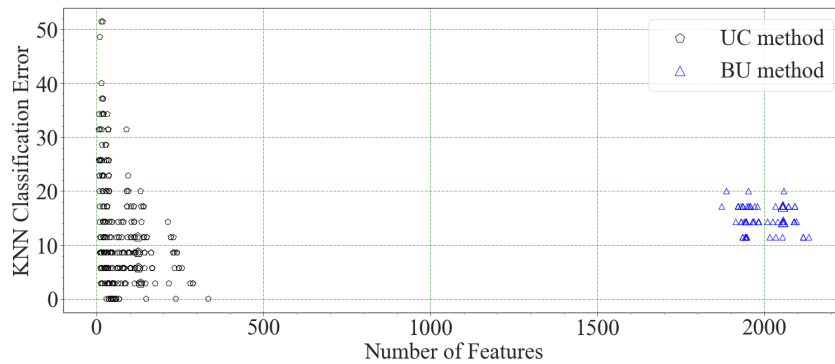
**Results on Arcene dataset:** Figure 4.5 presents the total generated Pareto front solutions for both BU and UC initialization same as the previous datasets. For the Arcene dataset, both algorithms are stopped after 50, 100, 250 iterations in order to capture the evolution of the Pareto fronts during the optimization. In Figure 4.5 the total Pareto front solutions are illustrated in (d) and (e) separately after 250 iterations for BU and UC method respectively. In Figure 4.6, Results on Arcene dataset evidently indicate the superiority of NSGA-II algorithm with UC initialization method since after non-dominated sorting only Pareto front solutions of UC method exist. Both average minimum error and its average number of features in Table 4.2 and Pareto fronts distribution graph confirm that NSGA-II algorithm with UC initialization start its optimization from better candidate solutions in the search space due to its effective initialization. Optimizer’s empowered exploration and exploitation abilities can be easily seen from its rapid convergence to optimal solution with very fewer features and lower classification error again for this experiment. Within 250 iteration NSGA-II with UC initialization is able to reach to least number of features with higher accuracy (lower error) for KNN model, however, for NSGA-II with BU initialization even after 250 iterations, algorithm



(a)



(b)



(c)

Figure 4.3: Pareto fronts (31 runs) of TOX-171 after (a) 50 iterations, (b) 200 iterations, (c) 400 iterations.

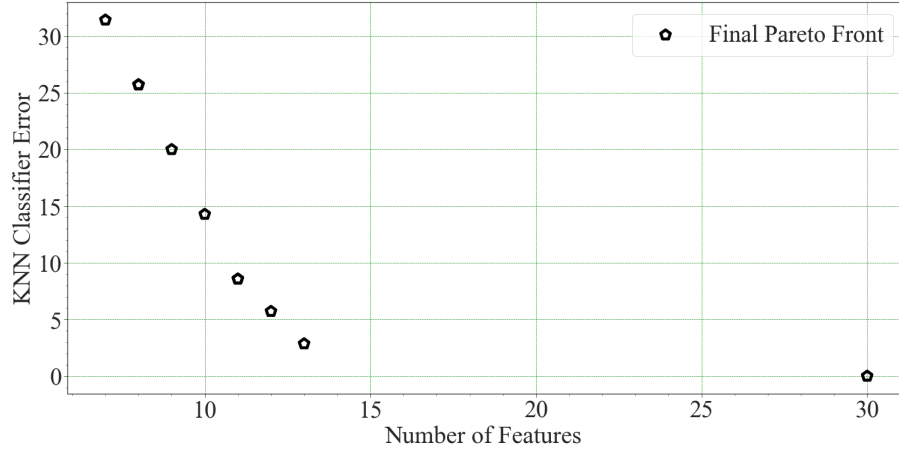
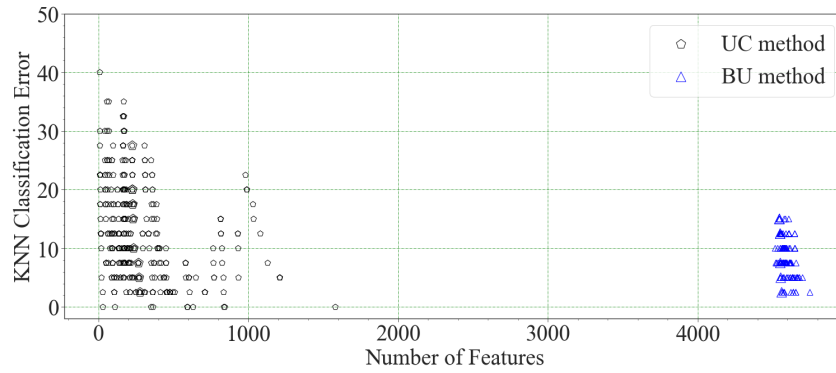


Figure 4.4: Final Pareto front after non-dominated sorting of total points in objective space for TOX-171 dataset

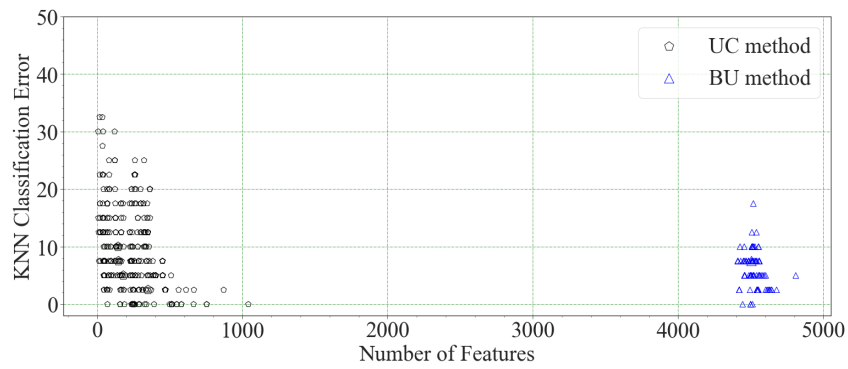
Table 4.2: Mean value of minimum errors for 31 runs and the corresponding average number of features for the solution with minimum error in BU and UC method. Results are compared based on t-test with p-value of 0.05.

Dataset	Initialization Method	Number of Feature	Min Error
Arcene	BU	101	24.84
	UC	<b>99</b>	<b>6.96</b>
Madelon	BU	1989	14.19
	UC	<b>114</b>	<b>2.58</b>
TOX-171	BU	4271	4.68
	UC	<b>313</b>	<b>1.45</b>

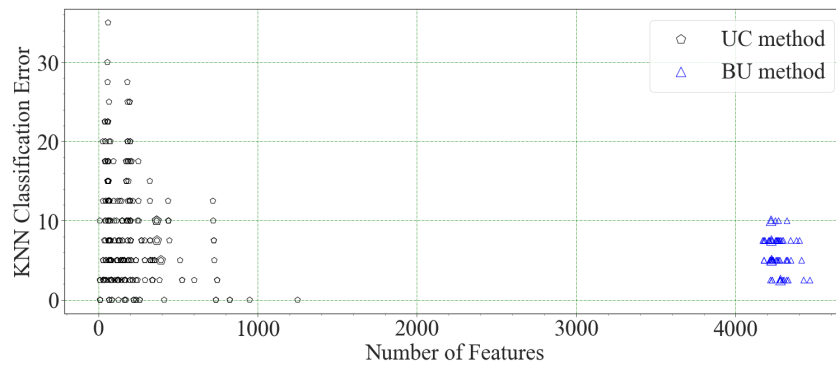
is far away from NSGA-II with UC initialization method's Pareto front solutions in its initial stages. This experiment again illustrates that UC method has highly improved the performance of the NSGA-II since all Pareto front solutions generated with UC method dominate Pareto front solutions generated with BU method. Besides, the algorithm in UC method achieves lower error with fewer number of features in comparison to BU method.



(a)



(b)



(c)

Figure 4.5: Pareto fronts (31 runs) of Arcene dataset after (a) 50 iterations, (b) 100 iterations, (c) 250 iterations.

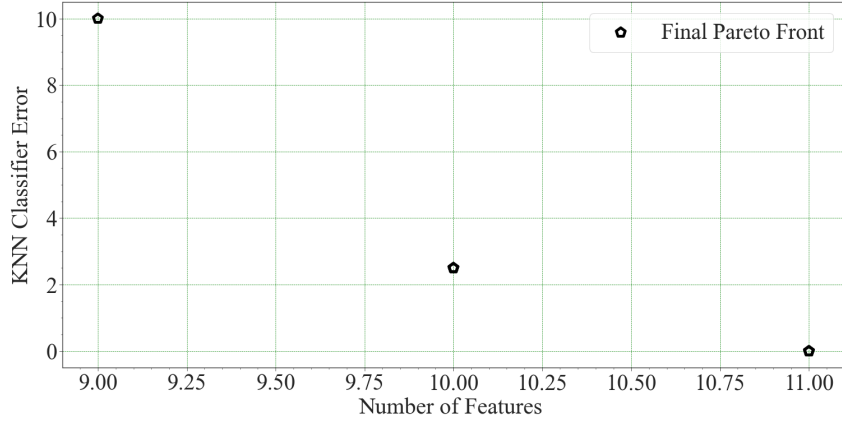


Figure 4.6: Final Pareto front after non-dominated sorting of total points in objective space for Arcene dataset

### 4.2.2 Multi-objective 0-1 Knapsack Problem

The multi-objective knapsack problem as a multi-objective combinatorial optimization problem can be solved using evolutionary multi-objective optimization algorithms, so it is a great candidate to assess the new initialization technique. 0-1 Knapsack problem as a combinatorial problem has strong similarity with finding qualified set of features in feature selection algorithms. Here, the multidimensional version of multi-objective 0-1 knapsack problem (MOMKP) is reviewed from chapter 2 for comprehensive understanding. Imagine having  $N$  items ( $i = 1, 2, \dots, n$ ) with  $m$  characteristics  $w_j^i (j = 1, 2, \dots, m)$  such as weight, volume, ... and  $p$  profits  $c_k^i (k = 1, 2, \dots, p)$ , some items should be selected in a way to maximize the  $p$  total profits while not exceeding the  $m$  knapsack capacities  $W_j$  regarding the various characteristics. The MOMKP problem is formulated as follows:

$$\begin{aligned}
 & \text{Max} \quad z_k(x) = \sum_{i=1}^n c_k^i x_i && k = 1, \dots, p \\
 & \text{Min} \quad q = \sum_{i=1}^n x_i && i = 1, \dots, n \\
 & \text{subject to} \quad \sum_{i=1}^n w_j^i x_i \leq W_j && j = 1, \dots, m \\
 & \quad \quad \quad x_i \in \{0, 1\} \quad i = 1, \dots, n
 \end{aligned} \tag{4.1}$$

In our experiments for knapsack problem,  $k$  and  $j$  have only one value,  $k = 1$  and  $j =$



1, meaning that there is only one constraint inequality, and also just one maximization function regarding the profits of the items inside a knapsack which has been change to the minimization problem due to the compatibility with performing optimization in Pymoo platform. For this aim the maximization objective function has been multiplied to a minus. Accordingly, in order to define a bi-objective knapsack problem in this thesis, we have considered two objective functions defined as: sum of total items in the knapsack which has to be minimized and another one is the objective function regarding the profit of items inside knapsack, consequently the formulation of the first knapsack problem is defined as:

$$\begin{aligned}
\text{Min} \quad & f_1(x) = z(x) = - \sum_{i=1}^n c_i x_i \\
\text{Min} \quad & f_2(x) = \sum_{i=1}^n x_i \\
\text{subject to} \quad & \sum_{i=1}^n w_i x_i \leq W \\
& x_i \in \{0, 1\} \quad i = 1, \dots, n
\end{aligned} \tag{4.2}$$

Different values for  $n$ , which is representative of dimension of the knapsack problem is determined, which include  $n = 50, 1000, 5000$  to compare the performance the algorithms in small and large scales.  $W$  is defined based on [46] and for the  $c_i$  and  $w_i$  as the profit and weight characteristic of the item  $i$  respectively, random integer number have been selected uniformly from  $[1, 100]$  for both coefficients for each item.

$$\begin{aligned}
c_k^i &= \text{uniform randomly selected from } [1,100] \\
C_k &= \{c_k^1, \dots, c_k^i, \dots, c_k^n\} & i = 1, \dots, n \\
w_i &= \text{uniform randomly selected from } [1,100] & i = 1, \dots, n
\end{aligned} \tag{4.3}$$

**Experimental Results and Analysis:** In order to investigate the effect of BU and UC initialization, the knapsack problems with different number of items (dimensions)  $n=50, 1000, 5000$  are solved using the NSGA-II algorithm with different initialization. Same as the multi-objective feature selection problem, different number of iterations are used as stop criteria of the algorithms in order to do the comparison at different stage of

the optimization. The parameter settings and control parameters are defined identical to the multi-objective feature selection optimization problem. The Pareto front solutions distribution for 31 runs and average ranking for Pareto front solutions are used to compare the performance of the NSGA-II algorithm with different initialization. Since in Knapsack problem, the optimal Pareto front solutions comprises both Pareto front solutions from BU and UC method, the ranking score metric is used for comparison of BU and UC method. Here for better understanding of evaluation, calculation of ranking score metric is described. The final results also are compared based on t-test with p-value of 0.05.

**Ranking Score:** In the process of ranking each Pareto front, first the  $f_1$  objective function is considered for solutions, and each solution in objective space will be ranked based on the fitness value for this objective. It means that the Pareto front solution with fitness value is ranked first with this method and the Pareto front solution with worst fitness value is ranked last (number of the total points in the objective space). Again, total Pareto front solutions are ranked based on their fitness value of  $f_2$  objective function. Finally, for calculation of ranking score of a solution, the ranking score is defined the average ranking of  $f_1$  and  $f_2$  in general. The ranking score for a Pareto front is the average ranking score of all its Pareto front points and the Pareto front with superior solutions has lower ranking score accordingly.

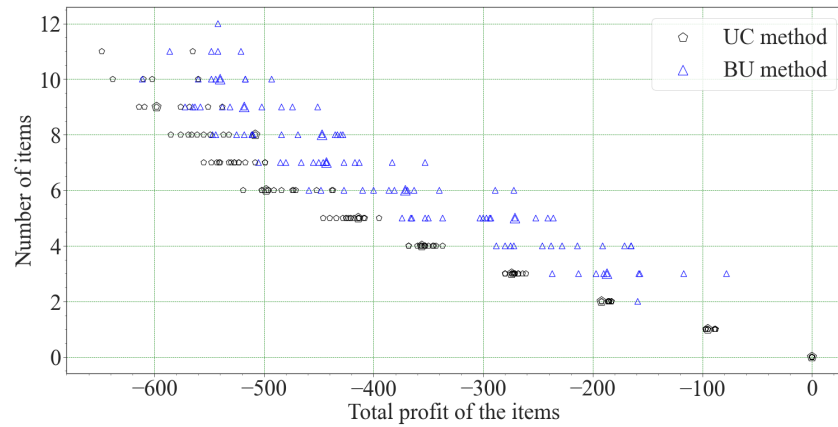
**Results on Knapsack with  $n=50$  items:** Figure 4.7 shows the Pareto front solutions for 31 runs after different iterations including 15, 25, 60 for the knapsack problem with  $n=50$  items. The  $f_1$  and  $f_2$  objective functions represent the total profit of the items in the knapsack and number of the items in knapsack respectively, and both objective functions are minimization problems. Table 4.3 shows the average ranking score for Pareto front solutions regarding first objective function, second objective function and in general, average score of two objective functions. Due to the vast accessible search space provided with UC method, NSGA-II starts the optimization from better candidate solutions and its exploration and exploitation ability is boosted due to the diversity exist

Table 4.3: The average ranking score for Pareto front solutions regarding first objective function ( $f_1$ ), second objective function ( $f_2$ ) and in general, average score of two objective functions defined as Ave-rank score for BU and UC method with different number of items including 50, 1000, and 5000.

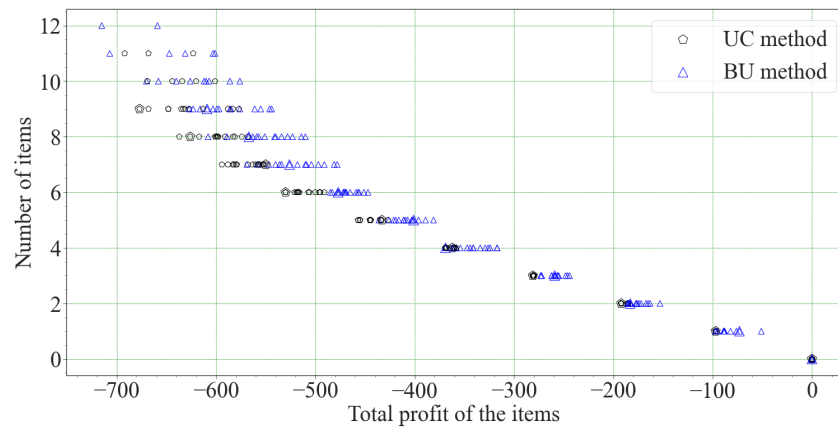
Items	Initialization Method	Avg- $f_1$ rank	Avg- $f_2$ rank	Avg-rank score
50	BU	5.9	11.8	8.85
	UC	5.5	10.8	<b>8.15</b>
1000	BU	63.13	130.2	96.67
	UC	125.4	64.53	<b>95</b>
5000	BU	49.3	155	102.15
	UC	150.5	50.5	<b>100.5</b>

in its population from the initial steps of optimization. Results in Figure 4.7 indicate that NSGA-II with UC method has almost reached to its optimal Pareto solutions with 25 iterations, however for NSGA-II with BU method it takes 60 iteration to reach to the same Pareto solutions. This means that good initialization not only empowers the algorithm to start the optimization from better candidate solutions, it also accelerate the exploration and exploitation capability of the optimizer. The Ave-rank score for BU method is 8.15 in comparison to BU method which is 8.85. Lower ranking score indicates the Pareto front solutions achieved fittest values for  $f_1$  and  $f_2$  objective function in general.

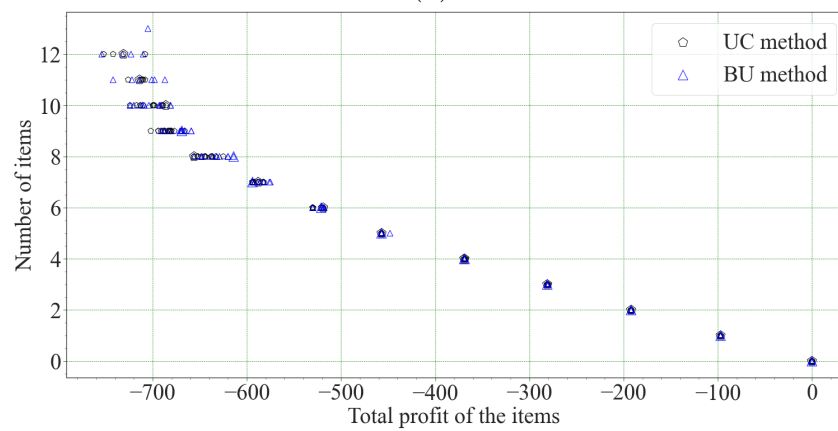
**Results on Knapsack with  $n=1000$  items:** Here in this experiment, the number of the items for the knapsack problem, is set to be 1000 to investigate the effect of BU and UC population initialization in a different dimension. Figure 4.8 presents the results of the experiment of solving knapsack problem with  $n=1000$  using NSGA-II algorithm with BU and UC population initialization methods. Again, both of the Pareto fronts distribution and the average ranking score, indicate the superiority of NSGA-II algorithm with UC initialization method to BU initialization method in solving the knapsack problem with  $n=1000$ . The average ranking score for UC method is 95 which is lower in comparison to BU method with average ranking value of 96.67. As we can see in the results, for  $n=1000$  the effect of UC initialization method is much powerful than lower dimensions



(a)



(b)

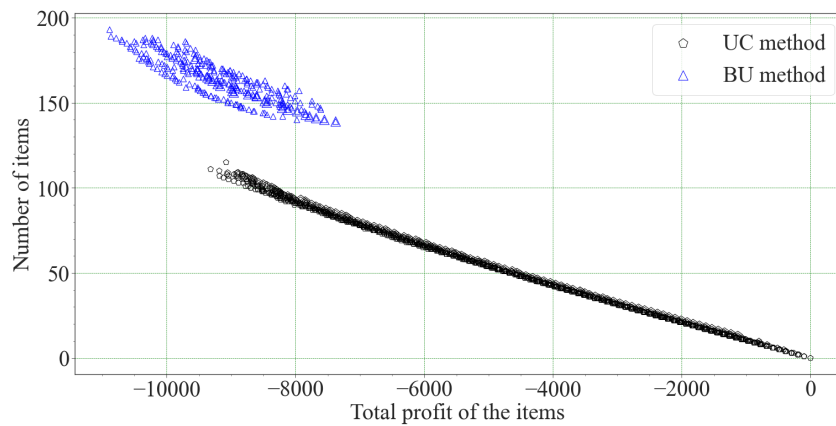


(c)

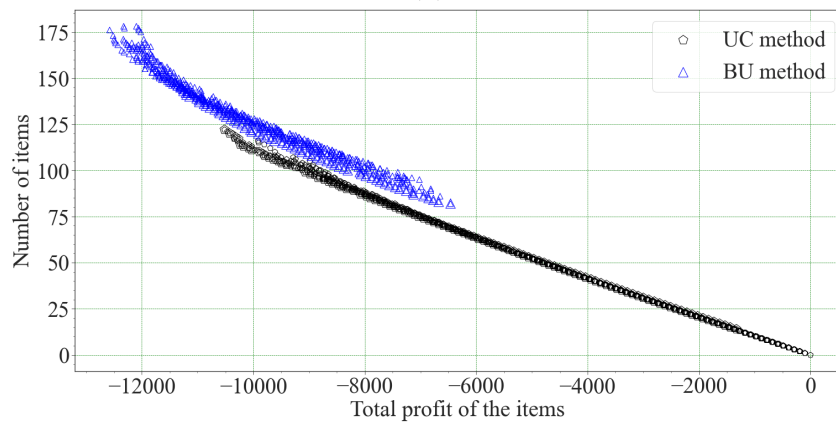
Figure 4.7: Pareto fronts distribution (31 runs) in the objective space including: 1-total profit of the items inside the knapsack and 2-number of the items inside the knapsack for (a) 15 iterations, (b) 25 iterations, (c) 60 iterations with 50 population size, for knapsack problem with  $n=50$ .

such as  $n=50$  and the Pareto fronts distributions is significantly different in BU and UC methods. In NSGA-II algorithm with UC, only after 200 iterations, the optimal Pareto front solutions are become widely distributed in the objective space, however, for NSGA-II with BU initialization method, the Pareto front only covers tiny region in the objective space and its poor population initialization forces the algorithm to use its budget on exploring the search space for the optimal solutions in only small region of the search space. Consequently, the convergence to optimal solution with BU method takes huge time in comparison to UC method as we see in the graphs. For this experiment population size is set to be 100. The main reason for this decision is that BU method is incapable in finding feasible solutions in the beginning generations such as 200. Accordingly, it takes to much time for BU method with lower population size to reach to a feasible solution. One solution witch is slightly helpful with this issue is to increase the population size as it used in this experiment.

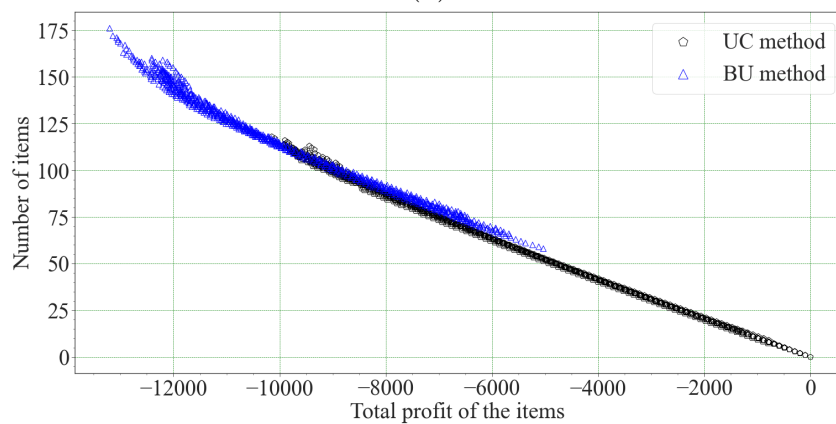
**Results on Knapsack with  $n=5000$  items:** Figure 4.9 shows the Pareto front solutions for 31 runs after different iterations including 1000, 1500, 2000 for the knapsack problem with  $n=5000$  items. The Pareto fronts distribution and the average ranking indicator, present the superiority of NSGA-II algorithm with UC method. The average ranking for BU method is 102.15 which is higher the average ranking for UC method with the value of 100.5. The lower rank for UC method indicates that Pareto front solutions achieve better values for the  $f_1$  and  $f_2$  objective function in average. This experiment also confirms our discovery on the severeness of shrinkage of search space due to the BU initialization method in larger dimensions since the generated Pareto front with BU method is distributed in a very tiny region in the objective space.



(a)

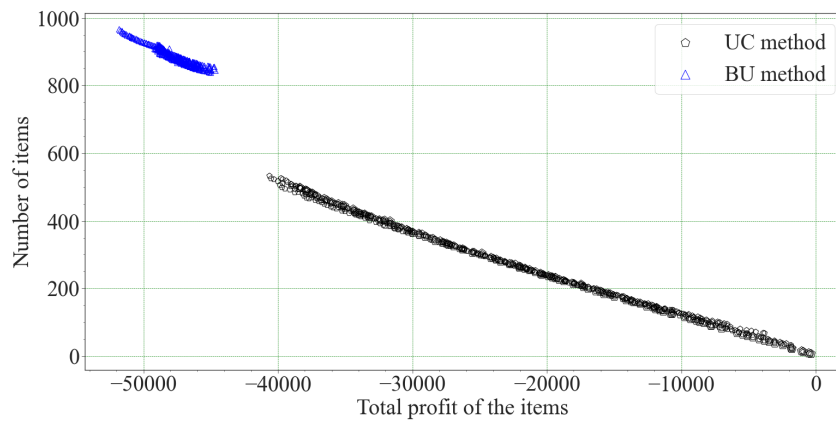


(b)

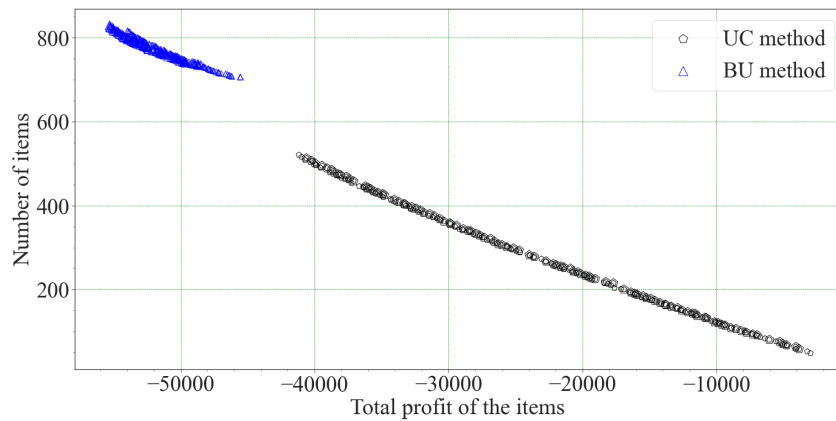


(c)

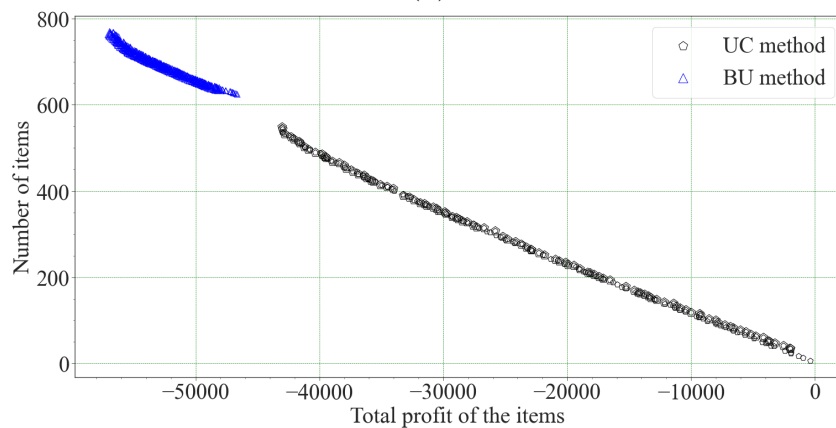
Figure 4.8: Pareto fronts distribution (31 runs) in the objective space including: 1-total profit of the items inside the knapsack and 2-number of the items inside the knapsack for (a) 200 iterations, (b) 400 iterations, (c) 600 iterations with 100 population size, for knapsack problem with  $n=1000$ .



(a)



(b)



(c)

Figure 4.9: Pareto fronts distribution (31 runs) in the objective space including: 1-total profit of the items inside the knapsack and 2-number of the items inside the knapsack for (a) 1000 iterations, (b) 1500 iterations, (c) 2000 iterations with 100 population size, for knapsack problem with  $n=5000$ .

## 4.3 Single-objective Problems

### 4.3.1 Single-objective Feature Selection

In this section, we have conducted several experiments on feature selection problem using the datasets described in Table 4.4. However, the feature selection problem is considered as a single-objective problem where the objective function is defined as the effectiveness of the machine learning model only. The main reason for the experiments on single-objective problems using single-objective optimizers is to eliminate the direct effect of population initialization that may affect the performance of the second objective (sum of number of features or sum of number of items) directly. In this part, the GA is the introduced population based metaheuristic algorithm which is responsible to find the optimal and the most relevant features to achieve the fittest value for the objective function. The KNN with  $k=5$  for classification task is defined as the machine learning model and the evaluation metric for effectiveness of the KNN model is determined the error of the KNN algorithm. The ratio of train to test data is also 80 to 20 similar to previous experiments. GA algorithm with two different population initialization including BU and UC methods, are used to solve the feature selection problem and the performance of the algorithm with different method of population initialization compared to each other. The parameter settings for GA algorithm is originated from the Pymoo platform. Besides, the two point crossover and bit flip mutation are defined as crossover and mutation operators respectively in implemented GA algorithm using the Pymoo platform. Each algorithm performs 31 times on each datasets. Average performance plot representative of best accuracy of the KNN algorithm over the generations/iterations is used to compare the GA algorithm with different initialization methods. The population size for all three problems is 50 and, stop criterion for all the algorithms is defined maximum number of generations equal to 150.



Table 4.4: Detailed information about used datasets in single-objective feature selection

Dataset	Number of Features	Number of Samples	Number of Classes
TOX-171	5748	171	4
Arcene	10000	900	2
GLA-BRA-180	49151	180	4

Table 4.5: Mean and variance of best solution (minimum error for KNN classifier) over 31 runs for the datasets

Dataset	Mean		Variance	
	BU	UC	BU	UC
TOX-171	5	<b>1</b>	0.125	0.05
Arcene	25	<b>22.22</b>	0	0.0172
GLA-BRA-180	25	<b>14</b>	1.68	1.83

**Dataset for single-objective feature selection:** In order to compare the performance of GA algorithm with BU population initialization and UC population initialization methods, 3 datasets are selected to perform GA-based feature selection. Table 4.4 shows the description of the datasets.

### Experimental Results and Analysis:

To evaluate the effect of the proposed UC initialization method in GA algorithm that is representative of a single-objective optimization algorithm, a series of experiments are conducted in this section. Table 4.5 presents the average best error (performance) for both BU and UC initialization for GA over 31 runs for three datasets in Table 4.4. The results are compared based on t-test with p-value of 0.05.

Results on GA-based feature selection with two different initialization indicate the superiority of performance of the feature selection algorithm with UC initialization method to feature selection algorithm with BU initialization method on these three datasets. Table 4.5 presents the mean value of best solutions (minimum error for KNN classifier) over 31 runs and the variance of the best solutions at the same time. Results show that the GA algorithm is more successful in finding the best solution for single-objective feature selection problem with GA initialized using UC method in all three datasets. The main

reason for such behaviour of BU method is that, with poor initialization method, algorithm is confined to search specific region of the search space. This phenomenon will cause two major issues:

- algorithm is trapped into local optimal and can not find the local optima due to the considering only specific region of the search space
- algorithm needs a huge budge in order to be able to expand its search direction

However, in GA with UC initialization method, because the GA algorithm starts its optimization from almost the quite good candidate solutions which are representative of entire search space, the chance of being trapped into local optimal decreases hugely. Beside, since the algorithm has these effective candidate solutions from initial steps of optimization, its convergence to the optimal solution is much faster in comparison to GA algorithm with BU initialization method.

### 4.3.2 Single-objective 0-1 Knapsack Problem

In this section, the single-objective version of the 0-1 knapsack problem has been solved using GA algorithm for evaluating the effect of BU and UC initialization in performance of the binary optimization algorithm. Again, the GA algorithm is introduced as representative of binary optimization algorithm in solving the single-objective 0-1 knapsack problem representative of combinatorial optimization problems. We can formulate the single-objective knapsack problem as follows:

$$\begin{aligned}
 \text{Min} \quad & f_1(x) = z(x) = - \sum_{i=1}^n c_i x_i \\
 \text{subject to} \quad & \sum_{i=1}^n w_i x_i \leq W \\
 & x_i \in \{0, 1\} \quad i = 1, \dots, n
 \end{aligned} \tag{4.4}$$

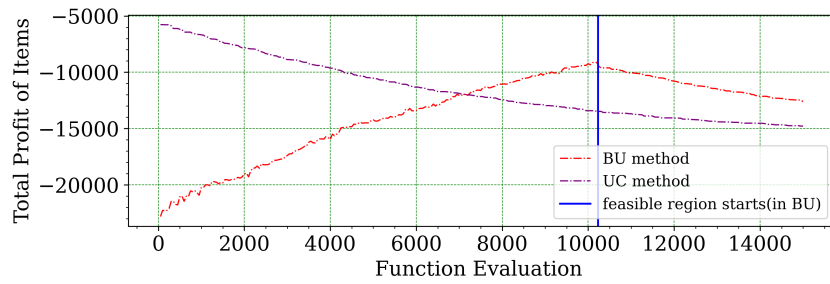
Same as multi-objective 0-1 knapsack problem, different values for  $n$  including 1000, 5000 are defined to compare the performance of the algorithms in small and large scales.  $W$

is defined based on [46] and for the  $c_i$  and  $w_i$  random integer number have been selected uniformly from [1, 100] for both coefficients for each item. The parameter setting for GA, and control parameters are identical to the previous experiments on single-objective feature selection with GA. For knapsack problem with  $n=1000$  the GA stops optimization after 1000 iteration and for  $n=5000$  the GA stops optimization after 3000 iterations. Population size is also 50 for both experiments, and both performance plot and constraint violation graphs have been plotted over function evaluations (iterations  $\times$  population size).

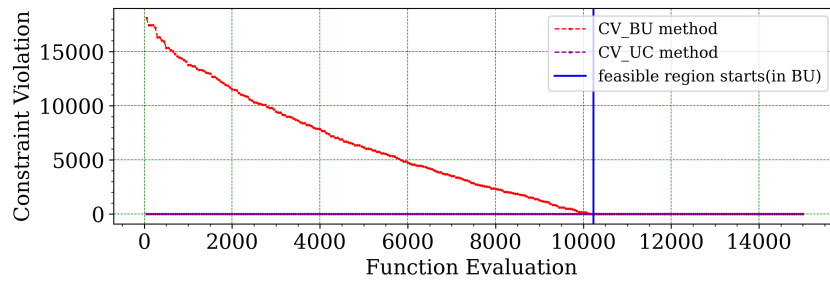
### **Experimental Results and Analysis:**

Figure 4.10 presents the performance plot and the minimum constraint violation of all individuals in the population in each function evaluation (generation  $\times$  population size). To evaluate the effect of the BU and UC initialization, we have conducted two experiments on single-objective 0-1 knapsack problem that is a representative of a combinatorial optimization problem. Same as previous experiments, to avoid the stochasticity in our results, each algorithm runs 31 times on each different dimension of the knapsack problems independently. The GA optimizer tries to find the best items to minimize the objective function (the original objective is the maximization of the total profit of the items inside the knapsack, however due to the compatibility with Pymoo platform it has been changed to the minimization problem by multiplying it with a minus) with one constrain on capacity of the knapsack defined as  $W$  again defined by Pymoo platform based on the dimension of the knapsack problem.

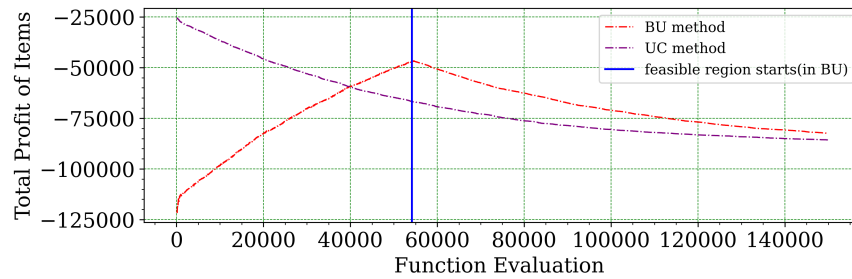
Considering the performance plots of  $n=1000$  and  $n=5000$  in Figure 4.10a and Figure 4.10c, the best solution in UC method is in the feasible region from the first generation. On the other hand, in BU method, the performance plots first show an ascending behaviour until finding the first feasible region which is determined by a vertical blue line in the performance plot figures. Looking at the constraint violation graphs in Figure 4.10b and Figure 4.10d, it can be seen that GA with BU initialization is not able



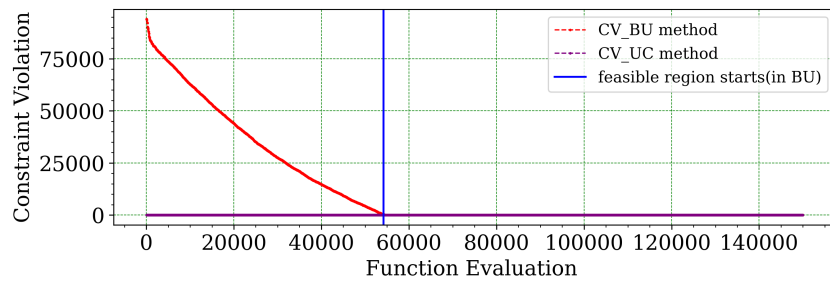
(a)



(b)



(c)



(d)

Figure 4.10: Performance plot of GA algorithm with BU and UC initialization in solving single-objective knapsack problem for  $n=1000$  and  $n=5000$  (a) and (C). In the knapsack problem with  $n=1000$  and  $n=5000$ , the optimization is stopped after 300 and 3000 iterations respectively. (b) and (d) present the minimum constraint violation (average of 31 runs) in the population over function evaluation.

to find the solutions in the feasible region and in first generations the budget is mainly used for reducing the constraint violation of the individuals. In this case, GA algorithm almost starts the process of optimization when non of individuals violate the constraint. This phenomena indicates that the BU method has delay in reaching to the best solution of the problem in comparison to UC method and is not able to overcome the constraints of the problem in first generations.

This phenomenon presents an other advantage of the UC method over BU method, specifically in problems that include constraints and the algorithms are unable to find the feasible solutions. This effect of this phenomenon is much intensive in problems with larger dimensions as the graphs shows. For  $n=5000$ , the GA algorithm needs to spend more budgets only to find the feasible solutions and continue the optimization from that point. Again, the main reason for this issue is poor initialization with BU and its bias to the specific region which may not include the optimal solutions in most of the problems since it only can covers 10% of the available search space. However, for GA with UC initialization method, due to the existence of the candidate solutions which are representative of entire search space, algorithm can quickly have access to region with optimal solution and local optima. This increase the same chance for solutions to be in feasible region as well. It is worth mentioning that, the proposed UC initialization method, regardless of its application in binary population initialization can be employed in any other algorithm which uses a set of bit-strings for testing or sampling. Accordingly, again providing diversity would play a crucial role when performing the sampling in these cases. For example, in the case of a designing a digital circuit with 100 inputs using a software that automatically performs this task, it would take a considerable time and cost for the software to design and test the results based on the full truth-table which has  $2^{100}$  possible states to be selected. Evaluating these total possible solutions are not manageable. If we want to provide some samples for the software to work, for example  $10^6$  samples form total  $2^{100}$  possible states, then the software may use the proposed

UC method as a technique which provides a small number of samples which fulfill the diversity and uniformity and are quite good example of total available solutions. Finally using these samples, the software would design and test the results with less cost and time.

## 4.4 Summary

In this chapter, two combinatorial optimization problems including feature selection problem and knapsack problem have been selected to evaluate the effect of BU and UC initialization on performance of the binary optimizer. Initially, the effect of different initialization first investigated in multi-objective variant of feature selection and knapsack problems, which minimization of number of features and items are the second objective function for feature selection and knapsack problem respectively. In the next step in our study for this thesis, in order to eliminate the direct effect of population initialization on performance of the algorithm, we have considered single-objective variants of the feature selection and knapsack problems. For multi-objective feature selection problem and knapsack problem, NSA-II algorithm with BU and UC initialization methods have been used to perform the feature selection on three datasets and also solve the knapsack problem with different number of the items including small, medium and large range of numbers. The selection of different dimensions (small to large) for all problems and experiments was introduced to evaluate the scalability of the proposed method and comparing the performance of the BU and UC algorithms in various dimensions. Distribution of the Pareto front solutions for all experiments and also the average ranking metrics confirm the significant superiority of UC initialization method to BU method. Convergence speed and quality of the Pareto front solutions are highly improved using UC initialization method. To eliminate the direct effect of the population initialization which may affect the performance of the algorithms due to its direct participation as

second objective function in multi-objective optimization problems, we have defined the single-objective versions of feature selection and knapsack problem again with different datasets and dimensions. These problems are solved using GA algorithm with BU and UC initialization methods. Objective functions for feature selection and knapsack problem in this case are defined the KNN classifier error and minimized version of maximizing total profit of the items inside a knapsack respectively. The performance plots indicate again the superiority of GA algorithm when initialized with UC method. Overall, the conducted experiments show that the suggested UC initialization method improves the performance of the binary optimization algorithms drastically.

# Chapter 5

## Conclusion and Future Direction

### 5.1 Conclusion Remarks

Random binary population initialization with generating bitstrings of zeros and ones originated from Bernoulli distribution with  $p = 0.5$  is the prevalent random binary population initialization due to its simplicity and a general belief of providing uniformity of distribution in the search space. However, the aforementioned binary population initialization namely as Bitstring-Uniform (BU) procedure of population initialization should be deeply studied to investigate its ability in covering the binary search space with its randomly generated bitstrings. The importance of this thesis and study originates from the fact that BU is still the most commonly used random binary population initialization method in the literature.

In this thesis, a comprehensive study has been done on the concept of uniformity in the binary search space. The concept of uniformity has been studied from two points of view including the chromosome-wise uniformity and gene-wise uniformity. First, a thorough research and investigation have been conducted to examine the fulfillment of chromosome-wise uniformity and gene-wise uniformity with BU population initialization method. Our deep investigation on BU initialization uncovered the fact that despite the



gene-wise uniformity, BU method is not able to fulfill the chromosome-wise uniformity. Using the Monte-Carlo simulation and mathematical proofs, the phenomenon of search space shrinkage due to the poor initialization with BU method is investigated. As an alternative method, the Uniform-Covering (UC) population initialization is a great solution for overcoming the adverse effect of search space shrinkage. However, this method has not been valued as it is needed in the literature. The reason is that, in binary population initialization for non-binary optimization problems and algorithms, the effect of the phenomenon of shrinkage of space vanished due to the existing mapping from binary search space to other types of search spaces such as real value, integer, and etc. But the fact is that phenomenon of shrinkage of search space due to BU initialization method drastically degrades the performance of the binary optimization algorithms since there is no mapping between the search spaces. A deep investigation on superiority of the UC method to BU method has been conducted as well in this thesis.

In general we could say that UC method's contributions are on three main directions:

1. Solving lack of diversity in initial population in binary optimization, diversity was very low for BU.
2. Low diversity makes evolutionary operations (i.e., cross-over and mutation) unable to create better candidate solutions, because of lack of diversity of initial population.
3. Proposed initialization creates a higher chance for initial population to be feasible because of their high diversity, but traditional method has a lower chance because of the lower diversity, starting with feasible candidate solutions is an important factor in final performance of the algorithm.

Several experiments are conducted to confirm the superiority of the UC population initialization to the BU population initialization method. First, as the case studies, the multi-objective feature selection and multi-objective 0-1 knapsack problems are selected

to evaluate the performance of an binary optimization algorithm when initialized differently with BU and UC methods. The NSGA-II is also selected as representative of multi-objective binary optimization algorithm. Three different datasets (Arcene, TOX-171, GLA-BRA-180) are used for multi-objective feature selection problem and three different dimensions  $n=50, 1000, 5000$  are introduced for knapsack problem. In both case studies, distribution of Pareto fronts in the objective space and also average ranking metric confirm the phenomenon of shrinkage of search space due to the severe deterioration of the performance of the algorithm when initialized using BU method instead of UC method. On the other hand, UC initialization, not only increases the convergence speed but it also assist the algorithm to not become trapped in local optima. For investigating the performance of the binary optimization algorithms in single-objective optimization problems, single-objective versions of knapsack problem and feature selection problems are studied. These experiments are conducted mainly to eliminate the direct effect of population initialization that is present in second objective of aforementioned multi-objective case studies. During these experiments, we investigated the ability of the UC initialization method in finding feasible solutions due to its wide-spread solution in entire search space from the initial step of the optimization. Overall, UC population initialization empowers the performance of the binary optimization algorithms drastically when employed as its procedure of population initialization method. In addition due to its powerful ability to generate a quite great and representative samples from  $n$  dimensional binary space with total  $2^n$  possible solution, UC method also can be used in sampling-based binary optimization methods as well.

## 5.2 Future Direction

Although this thesis proposed the UC population initialization method as an alternative for BU initialization method, there are still a lot of room for development. In this sub-

section, potential improvements and some of recommendations are:

- Investigation of effect of BU and UC methods on other families of binary optimization problems.
- Investigation on higher number of objectives (many-objective optimization).

# Bibliography

- [1] T. Bäck and H.-P. Schwefel, “An overview of evolutionary algorithms for parameter optimization,” *Evolutionary computation*, vol. 1, no. 1, pp. 1–23, 1993.
- [2] D. Dasgupta and Z. Michalewicz, *Evolutionary algorithms in engineering applications*. Springer Science & Business Media, 2013.
- [3] K. Deb, “Multi-objective optimisation using evolutionary algorithms: an introduction,” in *Multi-objective evolutionary optimisation for product design and manufacturing*, pp. 3–34, Springer, 2011.
- [4] B. Kazimipour, X. Li, and A. K. Qin, “A review of population initialization techniques for evolutionary algorithms,” in *2014 IEEE congress on evolutionary computation (CEC)*, pp. 2585–2592, IEEE, 2014.
- [5] P. L’Ecuyer, “Uniform random number generators: A review,” in *Proceedings of the 29th conference on Winter simulation*, pp. 127–134, 1997.
- [6] M. R. Hassanzadeh and F. Keynia, “An overview of the concepts, classifications, and methods of population initialization in metaheuristic algorithms,” *Journal of Advances in Computer Engineering and Technology*, vol. 7, no. 1, pp. 35–54, 2021.
- [7] J. O. Agushaka and A. E. Ezugwu, “Initialisation approaches for population-based metaheuristic algorithms: a comprehensive review,” *Applied Sciences*, vol. 12, no. 2, p. 896, 2022.

- [8] J.-S. Pan, P. Hu, V. Snášel, and S.-C. Chu, “A survey on binary metaheuristic algorithms and their engineering applications,” *Artificial Intelligence Review*, vol. 56, no. 7, pp. 6101–6167, 2023.
- [9] D. Simon, *Evolutionary optimization algorithms*. John Wiley & Sons, 2013.
- [10] B. Kazimipour, X. Li, and A. K. Qin, “Initialization methods for large scale global optimization,” in *2013 IEEE congress on evolutionary computation*, pp. 2750–2757, IEEE, 2013.
- [11] M. Sarhani, S. Voß, and R. Jovanovic, “Initialization of metaheuristics: comprehensive review, critical analysis, and research directions,” *International Transactions in Operational Research*, vol. 30, no. 6, pp. 3361–3397, 2023.
- [12] A. Tharwat and W. Schenck, “Population initialization techniques for evolutionary algorithms for single-objective constrained optimization problems: Deterministic vs. stochastic techniques,” *Swarm and Evolutionary Computation*, vol. 67, p. 100952, 2021.
- [13] M. Pant, T. Radha, and V. P. Singh, “Particle swarm optimization: Experimenting the distributions of random numbers.,” in *IICAI*, pp. 412–420, 2007.
- [14] Z. Ma and G. A. Vandenbosch, “Impact of random number generators on the performance of particle swarm optimization in antenna design,” in *2012 6th European conference on antennas and propagation (EUCAP)*, pp. 925–929, IEEE, 2012.
- [15] C. Wilbaut, R. Todosijevic, S. Hanafi, and A. Fréville, “Heuristic and exact reduction procedures to solve the discounted 0–1 knapsack problem,” *European Journal of Operational Research*, vol. 304, no. 3, pp. 901–911, 2023.

- [16] V. Cacchiani, M. Iori, A. Locatelli, and S. Martello, “Knapsack problems—an overview of recent advances. part ii: Multiple, multidimensional, and quadratic knapsack problems,” *Computers & Operations Research*, vol. 143, p. 105693, 2022.
- [17] G. Chandrashekar and F. Sahin, “A survey on feature selection methods,” *Computers & Electrical Engineering*, vol. 40, no. 1, pp. 16–28, 2014.
- [18] B. Xue, M. Zhang, W. N. Browne, and X. Yao, “A survey on evolutionary computation approaches to feature selection,” *IEEE Transactions on evolutionary computation*, vol. 20, no. 4, pp. 606–626, 2015.
- [19] S. Nersisyan, V. Novosad, A. Galatenko, A. Sokolov, G. Bokov, A. Konovalov, D. Alekseev, and A. Tonevitsky, “Exhausts: exhaustive search-based feature selection for classification and survival regression,” *PeerJ*, vol. 10, p. e13200, 2022.
- [20] S. Mirjalili and S. Mirjalili, “Genetic algorithm,” *Evolutionary Algorithms and Neural Networks: Theory and Applications*, pp. 43–55, 2019.
- [21] M. Dorigo and T. Stützle, *Ant colony optimization: overview and recent advances*. Springer, 2019.
- [22] K. V. Price, “Differential evolution,” in *Handbook of optimization: From classical to modern approach*, pp. 187–214, Springer, 2013.
- [23] D. Karaboga and B. Basturk, “A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm,” *Journal of global optimization*, vol. 39, pp. 459–471, 2007.
- [24] R. Eberhart and J. Kennedy, “Particle swarm optimization,” in *Proceedings of the IEEE international conference on neural networks*, vol. 4, pp. 1942–1948, Citeseer, 1995.

- [25] O. H. Babatunde, L. Armstrong, J. Leng, and D. Diepeveen, “A genetic algorithm-based feature selection,” 2014.
- [26] O. H. Babatunde, L. Armstrong, J. Leng, and D. Diepeveen, “Zernike moments and genetic algorithm: Tutorial and application,” 2014.
- [27] R. Leardi, R. Boggia, and M. Terrile, “Genetic algorithms as a strategy for feature selection,” *Journal of chemometrics*, vol. 6, no. 5, pp. 267–281, 1992.
- [28] B. Xue, M. Zhang, and W. N. Browne, “Particle swarm optimisation for feature selection in classification: Novel initialisation and updating mechanisms,” *Applied soft computing*, vol. 18, pp. 261–276, 2014.
- [29] A.-D. Li, B. Xue, and M. Zhang, “Improved binary particle swarm optimization for feature selection with new initialization and search space reduction strategies,” *Applied Soft Computing*, vol. 106, p. 107302, 2021.
- [30] M. Prasad, S. Tripathi, and K. Dahal, “Unsupervised feature selection and cluster center initialization based arbitrary shaped clusters for intrusion detection,” *Computers & Security*, vol. 99, p. 102062, 2020.
- [31] H. Xu, B. Xue, and M. Zhang, “Segmented initialization and offspring modification in evolutionary algorithms for bi-objective feature selection,” in *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, pp. 444–452, 2020.
- [32] N. L. S. Albashah and H. M. Rais, “Population initialization factor in binary multi-objective grey wolf optimization for features selection,” *IEEE Access*, vol. 10, pp. 114942–114958, 2022.
- [33] E. Emary, H. M. Zawbaa, and A. E. Hassanien, “Binary ant lion approaches for feature selection,” *Neurocomputing*, vol. 213, pp. 54–65, 2016.

- [34] H. Lim and D.-W. Kim, “Mfc: Initialization method for multi-label feature selection based on conditional mutual information,” *Neurocomputing*, vol. 382, pp. 40–51, 2020.
- [35] Y. Xue, X. Cai, and F. Neri, “A multi-objective evolutionary algorithm with interval based initialization and self-adaptive crossover operator for large-scale feature selection in classification,” *Applied Soft Computing*, vol. 127, p. 109420, 2022.
- [36] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, “A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii,” in *Parallel Problem Solving from Nature PPSN VI: 6th International Conference Paris, France, September 18–20, 2000 Proceedings 6*, pp. 849–858, Springer, 2000.
- [37] S. Mostaghim and J. Teich, “Strategies for finding good local guides in multi-objective particle swarm optimization (mopso),” in *Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS’03 (Cat. No. 03EX706)*, pp. 26–33, IEEE, 2003.
- [38] E. Zitzler, M. Laumanns, and L. Thiele, “Spea2: Improving the strength pareto evolutionary algorithm,” *TIK report*, vol. 103, 2001.
- [39] T. Murata, H. Ishibuchi, *et al.*, “Moga: multi-objective genetic algorithms,” in *IEEE international conference on evolutionary computation*, vol. 1, pp. 289–294, IEEE Piscataway, 1995.
- [40] B. Babu and M. M. L. Jehan, “Differential evolution for multi-objective optimization,” in *The 2003 Congress on Evolutionary Computation, 2003. CEC’03.*, vol. 4, pp. 2696–2703, IEEE, 2003.
- [41] Y. Gao, H. Guan, Z. Qi, Y. Hou, and L. Liu, “A multi-objective ant colony system algorithm for virtual machine placement in cloud computing,” *Journal of computer and system sciences*, vol. 79, no. 8, pp. 1230–1242, 2013.



- [42] R. Zakaria, M. Dib, and L. Moalic, “Multiobjective car relocation problem in one-way carsharing system,” *Journal of Modern Transportation*, vol. 26, pp. 297–314, 2018.
- [43] L. Kallel and M. Schoenauer, “Alternative random initialization in genetic algorithms,” in *ICGA*, pp. 268–275, Citeseer, 1997.
- [44] C. Z. Mooney, *Monte carlo simulation*. No. 116, Sage, 1997.
- [45] P. Agrawal, H. F. Abutarboush, T. Ganesh, and A. W. Mohamed, “Metaheuristic algorithms on feature selection: A survey of one decade of research (2009-2019),” *Ieee Access*, vol. 9, pp. 26766–26791, 2021.
- [46] J. Blank and K. Deb, “pymoo: Multi-objective optimization in python,” *IEEE Access*, vol. 8, pp. 89497–89509, 2020.
- [47] F. A. Zainuddin, M. F. Abd Samad, and D. Tunggal, “A review of crossover methods and problem representation of genetic algorithm in recent engineering applications,” *International Journal of Advanced Science and Technology*, vol. 29, no. 6s, pp. 759–769, 2020.
- [48] A. Hassanat, K. Almohammadi, E. Alkafaween, E. Abunawas, A. Hammouri, and V. S. Prasath, “Choosing mutation and crossover ratios for genetic algorithms—a review with a new dynamic approach,” *Information*, vol. 10, no. 12, p. 390, 2019.
- [49] Z. Zhao, F. Morstatter, S. Sharma, S. Alelyani, A. Anand, and H. Liu, “Advancing feature selection research,” *ASU feature selection repository*, pp. 1–28, 2010.
- [50] J. Elhachmi and Z. Guennoun, “Cognitive radio spectrum allocation using genetic algorithm,” *EURASIP Journal on Wireless Communications and Networking*, vol. 2016, pp. 1–11, 2016.

# Appendices

# Appendix A

## Optimization Algorithms

## A.1 Genetic Algorithm (GA)

---

**Algorithm 1:** Genetic Algorithm-Pseudo Code [50]

---

**Output:**  $x^*$  : Best candidate solution

**Given:**

$N$ : Number of chromosomes. /\* the size of population \*/

$MaxGeneration$  (Iteration): termination criterion for while loop.

**begin:**

$k=1$ ;

Generate initial population  $X^k = \{x_1^k, x_2^k, \dots, x_N^k\}$ ;

Calculate Fitness ( $x_i^k$ ), for  $i = 1, \dots, N$ ;

**do**

$k=k+1$ ;

**for**( $iter=1$ ;  $iter \leq N/2$  ;  $iter=iter+1$ )

/\* produce new generation  $X_{new}$  \*/

Randomly pick two chromosomes  $x_p$  and  $x_q$  from  $X^{k-1}$ ;

Produce two new chromosomes  $x_p^{new}$  and  $x_q^{new}$  by crossovering  $x_p$  and  $x_q$  with crossover probability  $p_c$ ;

Perform mutation operation on  $x_p^{new}$  and  $x_q^{new}$  with mutation probability  $p_m$ ;

**Calculate Fitness**( $x_p^{new}$ ) **and Fitness**( $x_q^{new}$ );

Insert  $x_p^{new}$  and  $x_q^{new}$  into  $X^{new}$  ;

**end for**

Pick  $N$  best chromosome from  $X^{k-1}$  and  $X^{new}$  to form  $X^k$ ;

$x^* :=$  the best chromosome in  $X^k$ ;

**while**( $k \leq MaxGeneration$ )

**return**  $x^*$ ;

**end**

---

## A.2 Non-dominated Sorting Genetic Algorithm II (NSGA-II)

---

**Algorithm 2:** NSGA-II Algorithm-Pseudo Code [42]

---

**Output:**  $X^*$  : Pareto Optimal Solutions

**Given:**

$N$ : Number of chromosomes. /\* the size of population \*/

$MaxGeneration$  (Iteration): termination criterion for while loop.

**begin:**

```

/* initialize the population  $P$  with  $N$  random individuals */
 $P \leftarrow init(N)$ ;
/* initialize an empty population for children */
 $Q \leftarrow \emptyset$ ;
 $eval(P)$  /* eval means evaluation of each individual in  $P$  */
for  $i=1$  to  $N$  do
   $P \leftarrow P \cup Q$ ;
   $assignRank(P)$  /* based on Pareto dominance */
  for each front  $f \in P$  do;
     $setCrowdingDist(f)$ ;
  end for
   $sort(P)$  /* by rank and in each rank by crowding distance */
   $P \leftarrow P[0:N]$ ;
   $Q \leftarrow buildNextGeneration(P)$  /* binary Tournament selection,
  Crossover, and mutation */
end for

```

---