BIRZEIT UNIVERSITY

THESIS REPORT

Next Release Optimization Problem Using Multi-Objective Harris Hawks Optimization Algorithm

Author:
Fadi Khalil (1175206)

Supervisor:
Dr. Majdi Mafarja

A thesis submitted in fulfillment of the requirements for the degree of Master of Software Engineering

Birzeit University, Palestine

August 4, 2021

جامعة بيرزيت
BIRZEIT UNIVERSITY

Approved by the thesis committee:

Dr. Majdi Mafarja, Birzeit University

[illegible]

Dr. Ahmed Abusnaina, Birzeit University

[illegible]

Dr. Sobhi Ahmed, Birzeit University

[illegible]

Date approved: 7/7/2021

Declaration of Authorship

I, Fadi Khalil , declare that this thesis titled, "Next Release Optimization Problem Using Multi-Objective Harris Hawks Optimization Algorithm" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a master degree at Birzeit University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: Fadi Khalil

Date: 25/6/2021

Acknowledgements

First of all I would thank Allah for getting this work done and my thesis advisor Dr. Majdi Mafarja from the Department of Computer Science / faculty of Engineering and Technology at Birzeit University. Dr. Majdi was always there when I needed him, whether I ran into a problem or needed a question Dr. Majdi was ready with the curing answer and advice.

This work couldn't be achieved without the support of my beloved wife and her encouragement throughout my years of study, nights and nights she was able to absorb all the nervousness I went through.

I dedicate this success to my children and whoever is coming down the road, to my parents, brothers, sister and family, to all of my friends. This accomplishment would not have been possible without them. Thank you.

Author

Fadi Khalil

Abstract

Companies that maintain large and complex software systems are usually facing Next Release Problem (NRP) in determining what requirements and features should be implemented in the next release [4].

Usually, there is a budget for software development, and there is a lot of requirements to be implemented. In this dilemma, the companies must satisfy customers' requests without exceeding the development budget limit.

In such a problem, requirement selection depends on the value, integrity, and dependencies between requirements. In this thesis, a new approach is proposed to tackle NRP. One of the recent Swarm Intelligent (SI) Meta Heuristics algorithms is used to tackle NRP, which is "Harris Hawks Optimization (HHO)" [24]. This algorithm is converted from a Single-Objective to Multi-Objective using two fitness functions, then used to find the best set of requirements that achieve customers' satisfaction within the development budget.

Different datasets from related NRP literature are used to assess the proposed approach's performance. It contains classic and real instances. Classic datasets were generated in labs to be used for experiments. The realistic datasets were derived from a real open source project. Both types include requirements cost, requirements requested by each customer, and the profit will be gained for implementing customer requirements.

In this work, three for the best wrapper meta-heuristics algorithms (NSGA-II, MOCell and MOCHC) are compared with the proposed MO-HHO algorithm using different datasets, each experiment is repeated multiple times to avoid getting good results by chance. The result showed that MOHHO outperforms NSGA-II, MOCell and MOCHC based on comparing Hypervolume values.

الملخص

عادة ما تواجه الشركات التي تقوم بتطوير أنظمة برمجيات كبيرة ومعقدة مشكلة تحديد محتويات الإصدار التالي في تحديد المتطلبات والميزات التي يجب تنفيذها في الإصدار التالي. في معظم الحالات يكون هناك ميزانية لتطوير البرامج ، وهناك الكثير من المتطلبات الواجب تنفيذها.

في مثل هذه المعضلة يجب على الشركات تلبية طلبات العملاء دون تجاوز حد ميزانية التطوير. وفي هذه الحالة يعتمد اختيار المتطلبات على القيمة الربحية والتكلفة التشغيلية والارتباطات بين المتطلبات. في هذه الأطروحة ، تم اقتراح نهج جديد لمعالجة هذه المشكلة. تُستخدم استراتيجية ماطردة الصقور للفريسة وهي إحدى خوارزميات ذكاء القطيع في حل المشكلة. يتم تحويل هذه الخوارزمية من هدف واحد إلى متعددة الاهداف باستخدام معادلتين لحساب تكلفة التطوير والفائدة العائدة من تطوير كل واحد من المتطلبات المطلوب اضافتها من الاصدار التالي من البرنامج، ثم يتم استخدامها للعثور على أفضل مجموعة من المتطلبات التي تحقق رضا العملاء ضمن ميزانية التطوير.

للمقارنة بين الطريقة الجديدة والدراسات السابقة ذات الصلة يتم استخدام مجموعة بيانات مختلفة من دراسات سابقة ذات صلة لتقييم أداء النهج المقترح. تحتوي هذه البيانات على بيانات غير حقيقية وأخرى حقيقية. تم إنشاء مجموعة البيانات الغير حقيقية في المختبرات لاستخدامها في التجارب. تم اشتقاق مجموعة البيانات الواقعية من مشروع حقيقي مفتوح المصدر. يتضمن كلا النوعين تكلفة المتطلبات والمتطلبات التي يطلبها كل عميل ، والعائد الربحي لتنفيذ متطلبات العميل .

في هذا العمل تمت مقارنة ثلاث خوارزميات من أفضل خوارزميات الاستدلال الفوقي NSGA-II و MOCell و MOCHC مع خوارزمية MOHHO المقترحة باستخدام مجموعة بيانات مختلفة ، يتم تكرار كل تجربة عدة مرات لتجنب الحصول على نتائج جيدة بالصدفة. أظهرت النتيجة أن الخوارزمية الجديدة MOHHO تتفوق في الأداء على NSGA-II و MOCell و MOCHC بناءً على عدة مؤشرات لمراقبة أداء الخوارزميات.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Software engineering is defined as a process of producing software applications that start by analyzing the user requirements, then build the required design, and the last step is testing the implemented software to make sure that it satisfies the requirements [25].

In engineering, a requirement is a document that describes what the system should be or do. It also describes the attribute, capability, characteristic, and quality of the system in order to produce a value and utility to the user [37].

Nowadays, the software market became very large; also the customers have many requests, this increase the need for an adaptive methodology that help in meeting the need of customers and receive feedback on delivered products [32]. This model of development is called an incremental model. This model makes use of delivering a small set of requirements; each deliverable is called a release. This development model becomes more popular because of Agile manifest and methodologies [43].

One of the common methods is used to manage the software engineering process is Agile methodology. It comes to boost the development speed while increasing the quality of the development process [48].

The software development companies usually have many feature requests from the customers, and almost there is a limited capacity to implement these requests. Most of the time, they do the enhancement in phases, so they should pick some of the requests to be implemented in the next release. On the other hand, they have to satisfy customers as much as possible without exceeding the development budget's maximum limit. This dilemma is called in literature as Next Release Problem (NRP)[1].

In NRP, the challenge is to pick a set of requirements that can be delivered within the budget while meeting customers' satisfaction. Making an incorrect decision may lead to a serious mistake. Good customers can be lost if their requirements didn't implement; the software development company may exceed the budget or even not deliver the release on time[4].

Software systems have become more popular use in our daily life. Also, the extensions of software systems have been increased and become more complex. Moreover, software companies have to develop and release product releases in a limited time. So, software development companies have to efficiently deal with features suggested by the customers because it is impossible to implement all requests. Thus, the problem here is to minimize the effort (time and money) and meet the customer's satisfaction. Selecting an optimal subset from these requests which achieve client satisfaction in the shortest time and lowest cost is the main challenge of software development companies. Sometimes, there are constraints in choosing the requirements. For example, two requirements may can't be implemented with the same release. However, we may have two requirements that must be developed in the same release, or one of them depends on the other. These constraints increase the complexity of dealing with this problem.

This challenge can be solved manually by finding all the solutions and calculate the cost and profit for each solution. The best solution is the solution that gets the highest profit and the lowest cost. This method is applicable to small scale projects. But when there is a large scale project; the number of possible solutions will grow exponentially; suppose we have two values for including the requirement, either 0 or 1, since 1 means will include the requirement, and 0 means will not include the requirement we will have 2^K solution space for K requirements. Thus, when we deal with large scale projects, the classical methods of NRP become impractical [45].

This problem becomes more complicated when we deal with multi objectives, and since meeting customers satisfaction and minimizing the efforts are contradictory objectives, this problem has classified in the literature as a Multi-Objective Optimization Problem (MOOP)[49].

Referring to the difficulty of dealing with NRP; In literature, it has been classified as a Knapsack problem, and Knapsack is under the category of NP-hard [38]. As a consequence, this problem can't be solved for a large number of requirements using exact optimization techniques where the number of possible solutions is very high. In this case, the approximation techniques (e.g., metaheuristics) can be used to tackle this problem [21]. Despite the fact metaheuristics algorithms cannot guarantee to find the best solution, they can find near-optimal solutions in a reasonable time [16].

This solution will be a single point solution in case of Single-Objective optimization algorithms. But in case of MOOP algorithms, it will be a list of feasible solutions which is called the Pareto optimal set. When this list is plotted in the objective space, it will be called as Pareto Front [49].

However, the solutions obtained in the Pareto should be close as much as can to the "optimal pareto front" of the problem. In NRP, the solutions obtained should produce the highest possible satisfaction of the customers while decreasing the development cost. Moreover, the Pareto front should provide a well-distributed solution that covers the maximum number of different solutions[16].

Despite that NRP can be obtained as MOOP, there is a lot of research that has considered this problem as a Single-Objective optimization problem. Moreover, some researchers have converted the objectives to a single objective using aggregations function[9].

Recently, many Swarm Intelligence (SI) metaheuristics algorithms have been proposed. It has been used in different optimization fields with much success[8]. One of these algorithms is Harris Hawks Optimization (HHO) [26].

HHO optimization algorithm is recently produced by Heidari et al. in 2019 [24]. This algorithm mimics the way of how Harris' hawks chasing the prey. In this tactic, several hawks attack the prey from different directions and pounce the prey.

HHO optimization method has already started helping in many different areas with a good reputation. It provides different chasing techniques based on the dynamic nature escaping patterns of the preys.

A Multi-Objective version of HHO is implemented in jMetal 5.6 Framework [34]. This framework is an open-source framework, it is designed to help in comparing algorithms' performance using different quality indicators. It has many single multi-objective algorithms that are already implemented. Also, it has many predefined problems to be used for testing the performance of the algorithms.

1.1 Motivation

Software development companies' trend is to use the methods that aim to help in getting feedback from the customers for the delivered releases, meet customer needs by increasing the speed of the development process. Since software development companies are expanding and getting more customers; this leads to having more requirements, which means the number of suitable solutions increases exponentially [45].

Having a large set of requirements leading to have a huge number of feasible solutions, which increase the complexity of finding the best solution. This motivated many researchers to tackle NRP using different optimization techniques. However, in optimization field there is a theorem called No-Free-Lunch (NFL), this theorem proves that there is no superior algorithm and applicable to solve all optimization problems[30]. This means almost there is room for enhancement on the current optimization methods. This motivated our attention to using HHO to tackle NRP since it has provided a promising search behavior in many fields[26].

1.2 Problem Statement

NRP is known to be a Multi-Objective optimization problem since it has two contradictory objectives (cost and profit). Therefore, a Multi-Objective version of the recently proposed HHO algorithm is adopted in this thesis.

In this work, firstly, a MO version of HHO is proposed. After that, two fitness functions are proposed to calculate cost and profit. Then, a dominance comparator function is proposed to compare the generated solutions and differentiate between them.

This work proposes a novel approach for tackling NRP using the HHO. It provides a promising search behavior in many fields. HHO is used to find the Pareto front for NRP, since it uses a new search technique that provides a diversity of search criteria with a fast searching way.

1.3 Research Objectives

This research aim is to propose a Multi-Objective version of HHO to help software companies in the software requirements selection problem. The implementation is done using JMetal 5.6 framework. HHO algorithm is enhanced to avoid stucking in local optimal solution in order to improve a better performance for the first time in literature. To achieve this goal, three research objectives were formulated as follows:

- To propose a Multi-Objective version of the HHO algorithm.
- Implementing the Multi-Objective version of the HHO algorithm in JMetal framework.
- To tackle NRP using Multi-Objective HHO.

1.4 Proposal Organization

The following chapters of this thesis report is organized as follows: Chapter 2 contains a literature review of the most related works in the fields of NRP, swarm-based algorithms, requirements prioritization, requirements interactions, and requirements selections. Chapter 3 provides an overview of the NRP, metaheuristics algorithms, multi-objective algorithms, and Harris Hawks optimization algorithm (HHO).

Chapter 4 describes the proposed approach's details and how HHO is converted from Single-Objective optimization algorithm to Multi-Objective optimization algorithm. It also shows how NRP is presented to be tackled using the Multi-Objective HHO algorithm.

Chapter 5 shows the results that are achieved using the proposed approach. Finally, Chapter 6 present the conclusion and future works.

Chapter 2

Background

This chapter introduces the NRP as well as the optimization algorithm used in tackling this problem. Moreover, this chapter will cover the method used to convert the HHO from Single-Objective to Multi-Objective.

2.1 The Next Release Problem

The NRP takes its role in selecting the requirement that should be developed in the software's next release. The selection of requirements process aims to find the set of requirements that achieve the maximum profit with minimum cost. This challenge can be classified as a Multi-Objective problem since it handles contradictory goals.

Bagnall et al. [4] defined the NRP as the challenge of selecting the requirement for the next release that meet the (important) customers within the software requirement company budget.

NRP was formulated as bi-objective at the first time in literature by Zhang et al in [49]. In their formulation, the first objective is the cost is lifted, while the constraint is presented as a second objective. Then, a set of efficient solutions in the Pareto sense will be presented for the decision-maker.

In other words, let R be a set of requirements, and this set is requested for implementation , and $r = (r_1, ..., r_n) \in \{0, 1\}^n$ is a binary vector of all requirements where the item r_i value will be 1 if and only if the i th requirement will be implemented in the next release. Let S be a set of m customers, and $s = (s_1, ..., s_m) \in \{0, 1\}^m$ is a binary customers vector, where the k th component has a value of 1 if and only if the requirements of customer k will be implemented in the next release. Let $c = (c_1, ..., c_n)$ be the cost vector that related to the requirements and $w = (w_1, ..., w_m)$ the weight vector that related to the customers, this vector represents the customer importance for the software development company. To define the constraints between the requirements, let P be the set of pairs (i, j) where requirement i is needed to be implemented before the requirement j and let Q be the set of pairs (i, k) where requirement i is needed for the customer k [14].

2.2 Metaheuristics algorithms

Metaheuristic algorithms are simple and very easy to implement algorithms designed to be competitive and alternative for solving many problems. The most advantage of these methods is that they do not depend on the specific information of the objective or its mathematical analysis. However, there is a drawback of these methods, they are often very sensitive to any change in the parameters defined by the user. Also, this type of algorithms may stuck in local optimal solution and not converge to the global optimal solution [24].

In general. There are two types of metaheuristic algorithms. First one us the single based solution (i.g. Simulated Annealing (SA)). And the other is population based (i.g. Genetic Algorithm (GA))[24]. There is a single solution proposed after the search complete in the first type, but in the second type, a set of solutions is found.

In the single solution based, only one solution can be proceed[24]. While in the Population Based a set of solutions can proceed, and each one of them can be the solution of the optimization problem[24].

The solutions added to the set (population) can be created iteratively, and the adding stopped when the maximum number of iterations reached. The four main groups of P-metaheuristics are described in Figure 2.1.

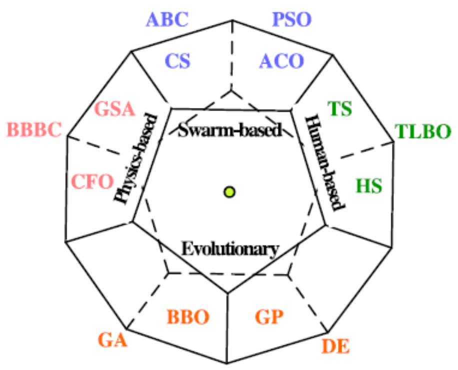


FIGURE 2.1: Metaheuristic Techniques [24]

Evolutionary Algorithms (EAs) inspired by the Biological Evolutionary Behaviors such as selection and mutation. The most popular example of this type is the Darwinian theory of evolution [24].

Physics-Based algorithms build based on physical laws. Central Force Optimization (CFO), Gravitational Search Algorithm (GSA), and Bang Big-Crunch (BBBC) are examples of these algorithms [24].

As the name indicates, Human-Based algorithms mimic Human behaviors. Examples of this type are Socio Evolution and Learning Optimization (SELO), Tabu Search (TS), and Teaching Learning Based Optimization (TLBO) [24].

The last category of these algorithms is Swarm Intelligence (SI) algorithms, which mimic the flocks, swarms, or herds which (e.g. self organized systems, decentralized) [24].

These categories searching process fails in two phases, first phase is exploration and then the second phase which is exploitation [42]. In the first phase, the algorithm starts searching randomly. It should encourage its operators to discover various regions of the search space. In the next phase, the algorithm starts exploitation by focusing on the neighborhood of best quality solutions found in the exploration. A well-organized algorithm should balance the exploration and exploitation phases to avoid stuck in the local optimal solution [24].

2.3 Multi Objective Algorithms

In Multi-Objective optimization (MOO) problem, there is no unique optimal solution, but multiple solutions is provided as the best solutions this set of solutions is called a Pareto front of solutions [11], this set should satisfy the constraints and optimize the objectives. In other words, the Pareto front contains Pareto solutions that are non-dominated by other solutions [9].

For example, we can say solution $s = [s_1, s_2, ..., s_n]$ is dominated solution $m = [m_1, m_2, ..., m_n]$ if there is no objective in m better than corresponding objective in s for any objective i = 1, 2, . . ., n, and there is one or more objective(s) s_i, in s better than corresponding objective(s) m_i in m [9].

On the other hand, two solutions are non-dominated if there is no solution that can dominate the other. Fig 2.2 show the difference between dominated and non-dominated solution. In this figure 2.2, F1 and F2 are two objectives that need to be minimized, and solution A is dominant solution D since F1(A) < F1(D) and F2(A) < F2(D) [9].

In case of MOO problem, the optimal solution is not a single solution. It is a set of solutions called a Pareto optimal set. This set has to satisfy two constraints; first one, there is no solution in the set dominating any other solution in the same set. The second one, any other solution founded in the search space is dominated by one or more solution(s) in the Pareto optimal set. When this Pareto optimal set is presented in the

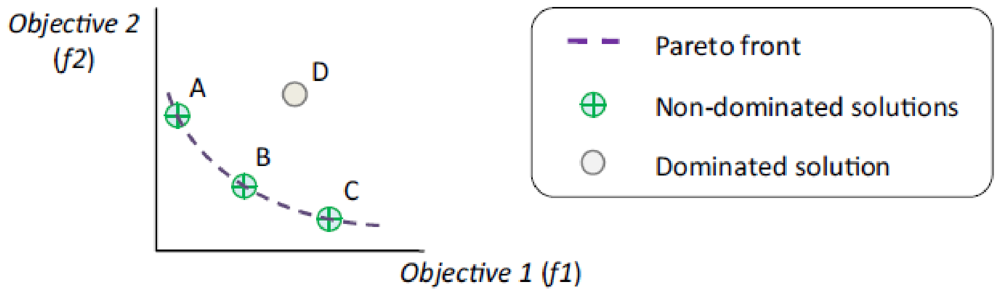


FIGURE 2.2: dominated and non-dominated solutions with the Pareto front [9]

objective space it is called as Pareto front [9].

Figure 2.3 explain how a solution can dominates other solutions.

For example, Solution number 3 dominates solution number 2, but Solution number 3 does not dominates solution number 5.

2.4 Harris Hawks Optimization

Harris Hawks Optimization (HHO) algorithm is one of the recently proposed algorithms, it is one of metaheuristic mechanisms proposed by Heidari et al. in 2019 [24].

This technique mimics the foraging behavior of Harris hawks in nature. The main goal of this algorithm is to find near-optimal solution(s) for a given issue by utilizing the population of search agents.

The exploration and exploitation phases of the proposed approach are starts by exploring prey and then surprise pounce. Different Harris hawks attacking strategies can be used. Figure 2.4 explain the phases of HHO.

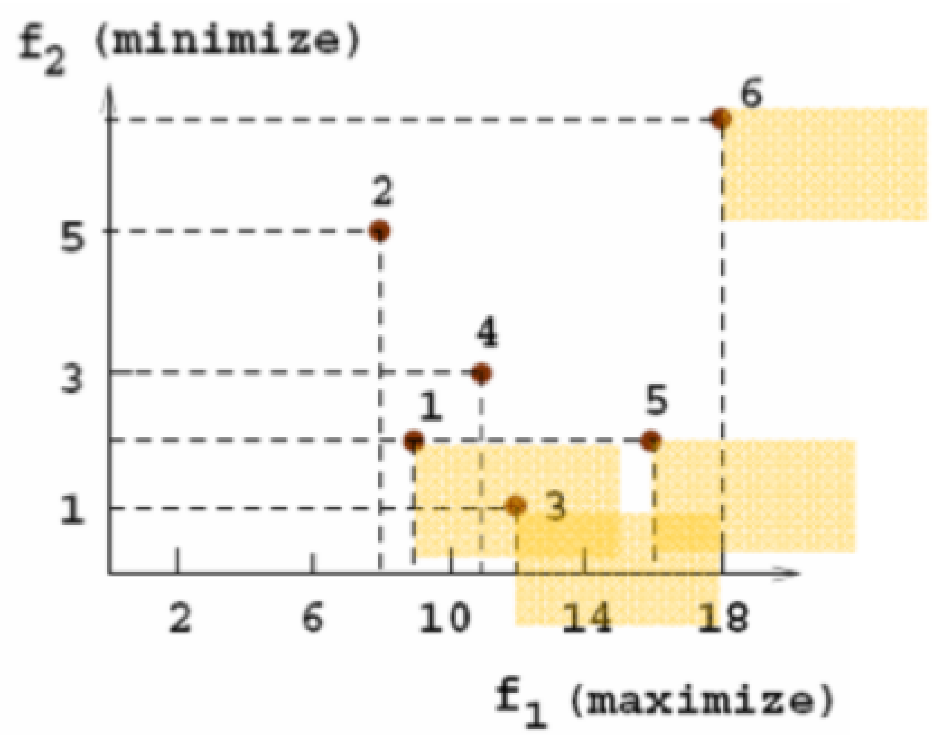
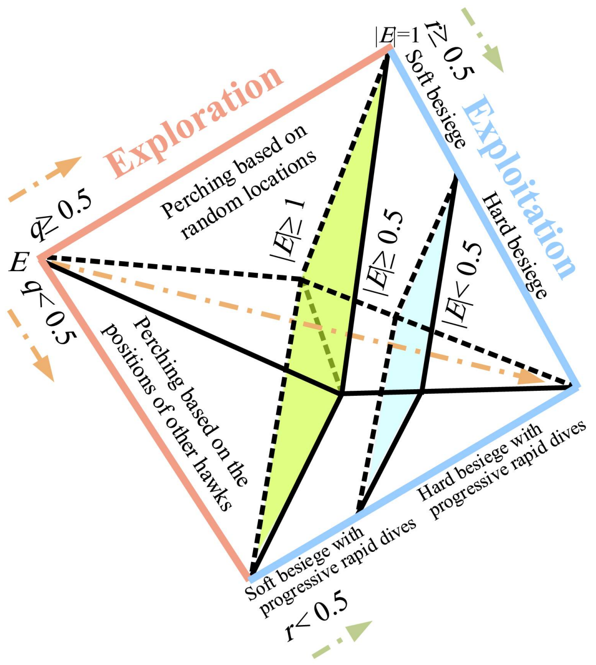


FIGURE 2.3: Optimal Non-dominated Solutions front [9]



Exploration
Exploitation
$|E|=1$
$r \geq 0.5$
Soft besiege
Hard besiege
$q \geq 0.5$
E
$q < 0.5$
Perching based on random locations
Perching based on the positions of other hawks
$|E| \geq 1$
$|E| \geq 0.5$
$|E| < 0.5$
Soft besiege with progressive rapid dives
Hard besiege with progressive rapid dives
$r < 0.5$

FIGURE 2.4: HHO phases [24]

2.4.1 Exploration phase

Each hawk in HHO is a candidate solution, and the closest one from the prey is the best solution. In HHO, the Harris' hawks randomly choose a location to perch and wait until they detect a prey, two strategies are used in detecting a prey. Let consider q as an equal chance for the perching strategy. They may perch based on other family members' location or may perch on a tree that chosen randomly. The two strategies are presented in Eq. (2.1).

$$X(t+1) = \begin{cases} X_{rand}(t) - r_1 \left| X_{rand}(t) - 2r_2 X(t) \right| & q \geq 0.5 \\ (X_{rabbit}(t) - X_m(t)) - r_3(LB + r_4(UB - LB)) & q < 0.5 \end{cases} \tag{2.1}$$

where $X(t+1)$ represents position of hawks in the next iteration t, $X_{rabbit}(t)$ is the rabbits' position, $X(t)$ is the hawks current position vector, r_1, r_2, r_3, r_4, and q are random numbers between 0 and 1, and these number should be updated for each iteration, LB is the lower bound and UB is the upper of variables, $X_{rand}(t)$ is a hawk location that randomly selected, and X_m is the average location of the hawks.

Harris hawks' locations inside the family group can be generated using range (LB, UB). In Eq. (2.1), the first rule generates the hawks location randomly. On the other hand, the second rule calculates the location based on the difference of location and the average position of the group plus a random number scale. The variable r_3 is a scaling variable, which increase the random nature of the rule when r_4 becomes close values to 1 and similar distribution patterns may occur. The hawks average position is presented in Eq. (2.2):

$$X_m(t) = \frac{1}{N}\sum_{i=1}^{N} X_i(t) \tag{2.2}$$

where $X_i(t)$ represent the hawk location in the iteration t and N is the the total number of the hawks.

2.4.2 Transition from exploration to exploitation

The HHO algorithm transfer from exploration phase to exploitation phase using different exploitation strategies, it depends on the escaping energy of the prey. The energy of the prey decreases during the escaping. The energy of prey is modeled in the following equation:

$$E = 2E_0(1 - \frac{t}{T}) \tag{2.3}$$

where E indicates the prey escaping energy, T is the maximum iterations number, and E_0 is the initial energy of the prey.

In HHO, E_0 is randomly changed between 0 and 1 in each iteration. When E_0 decreases from 0 to -1, the rabbit is physically flagging; however, when E_0 is increased, this indicates that the rabbit is strengthening.

When escaping energy is $|E|$ $\geq$1, the Harris hawks keep in the exploration phase and search in a different region for a rabbit location. When $|E|$ <1, the algorithm transition from exploration phase to exploitation phase and start searching in the neighborhoods. The time-dependent behavior for the rabbit energy E is demonstrated in Fig. 2.5.

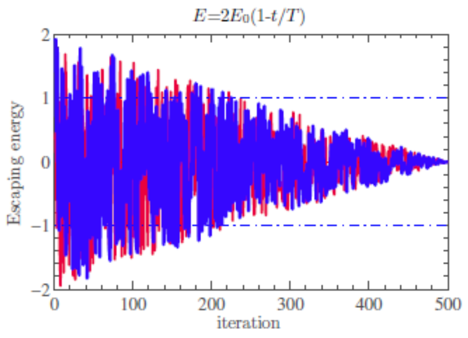


FIGURE 2.5: Behavior of rabbit escaping energy during two runs and 500 iterations [24]

2.4.3 Exploitation phase

After several attempts in the exploration phase, the rabbit escaping energy will be decreased, and the hawks will start another attacking strategy, the hawks start the attack by surprise pounce on the prey detected in the exploration phase. However, the rabbit may attempt to escape. The hawks will do different chasing styles according to escaping behavior. There are four possible strategies to attack the escaping prey.

When the attack starts, the prey will always have a high escaping energy, and it tries to escape. Let r be the chance when the prey escaping successfully (r <0.5), and (r ≥0.5) when the prey not escaping successfully before the surprise pounce. Based on the pray behaviour, the hawks will start hard and soft besiege to catch the prey. The hawks will start the attack from different directions and become closer and closer from the prey, also the prey energy will decrease after several of minutes, the prey will start losing its energy; then, the hawks will start intensive besiege to catch the prey.

In this phase, when $|E|$ $\geq$0.5, the hawks perform soft besiege, and when $|E|$ $<$0.5, the hawks perform hard besiege.

Soft besiege

When $r \geq 0.5$ and $|E| \geq 0.5$, the energy of the rabbit is still enough for escaping, in this case, the Harris hawks encircle the rabbit and try to make the rabbit loos his energy. After that, they perform the surprise pounce. This behaviour is model as the eq 2.4:

$$X(t+1) = \Delta X(t) - E\left|JX_{rabbit}(t) - X(t)\right| \tag{2.4}$$

$$\Delta X(t) = X_{rabbit}(t) - X(t) \tag{2.5}$$

where $\Delta X(t)$ is the represent the difference between the rabbit position and the current location in the iteration t, r_5 is a random number between 0 and 1, and $J = 2(1 - r_5)$ is the rabbit random jump strength throughout the escaping. J is a random value that changes in each iteration to simulate the rabbit motions in nature.

Hard besiege

When r $\geq$0.5 and $|E|$ $<$0.5, the prey becomes exhausted and his energy low. In this case, the rabbit will be encircled hardly bu the hawks to perform the surprise pounce. The current hawks positions in this situation are updated using eq 2.6:

(2.6):

$$X(t+1) = X_{rabbit}(t) - E\,|\Delta X(t)| \tag{2.6}$$

This step is also explained in fig 2.6

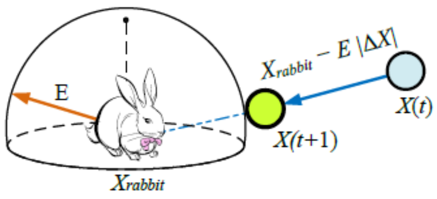


FIGURE 2.6: Hard besiege example [24]

Soft besiege with progressive rapid dives

In case of $|E|$ $\geq$0.5 and r <0.5, the rabbit energy is still enough for successfully escape, and hawks need to do a soft besiege before the surprise pounce.

In order to formulate the prey escaping patterns in a mathematical model, the Levy Flight (LF) concept is used in HHO [6]. LF concept is used to mimic the zigzag motion of the preys during the escaping [24].

The following equation performs the hawks movement in this phase:

$$Y = X_{rabbit}(t) - E\,|JX_{rabbit}(t) - X(t)| \tag{2.7}$$

After that, the hawks compare the last movement result to the previous movement to decide if it is a good dive or not. After that, hawks will start diving based on LF patterns that presented in the following equation:

$$Z = Y + S \times LF(D) \tag{2.8}$$

where D is the problem dimension and S is a random vector of size $1 \times D$ and LF is the function of levy flight. LF patterns is represented in Eq. (2.9):

$$LF(x) = 0.01 \times \frac{u \times \sigma}{|v|^{\frac{1}{\beta}}}, \sigma = \left(\frac{\Gamma(1+\beta) \times sin(\frac{\pi\beta}{2})}{\Gamma(\frac{1+\beta}{2}) \times \beta \times 2^{(\frac{\beta-1}{2})})} \right)^{\frac{1}{\beta}} \tag{2.9}$$

where u, v are random values between 0 and 1, β is a constant with a default value set to 1.5.

The final stage in the soft besiege phase for updating the hawks positions in presented in Eq. (2.10):

$$X(t+1) = \begin{cases} Y & ifF(Y) < F(X(t)) \\ Z & ifF(Z) < F(X(t)) \end{cases} \tag{2.10}$$

where Y and Z are calculated using Eqs.(2.7) and (2.8).

A demonstrated of this step is in Fig. 2.7.

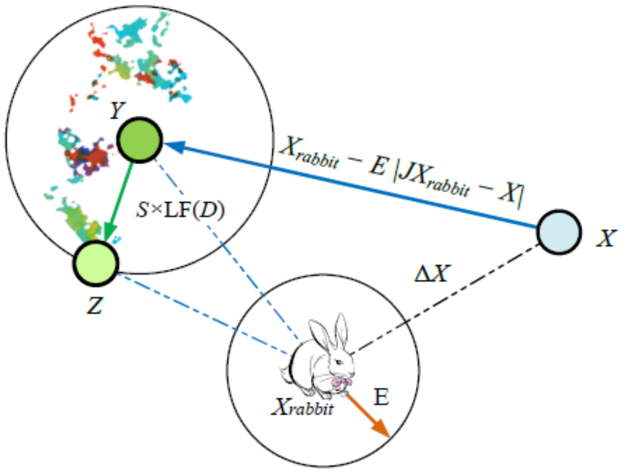


FIGURE 2.7: Soft besiege with progressive rapid dives[24]

Hard besiege with progressive rapid dives

In case of $|E|$ <0.5 but r <0.5, the rabbit energy is exhausted , and hard besiege should be done before the surprise pounce. The following equation performs the hard besiege in this phase:

$$X(t+1) = \begin{cases} Y & if F(Y) < F(X(t)) \\ Z & if F(Z) < F(X(t)) \end{cases} \tag{2.11}$$

where Y and Z are calculated using Eqs.(2.12) and (2.13).

$$Y = X_{rabbit}(t) - E\,|JX_{rabbit}(t) - X_m(t)| \tag{2.12}$$

$$Z = Y + S \times LF(D) \tag{2.13}$$

where $X_m(t)$ is calculated using Eq. (2.2). Fig. 2.10 represents an example of this step.

[b]0.65

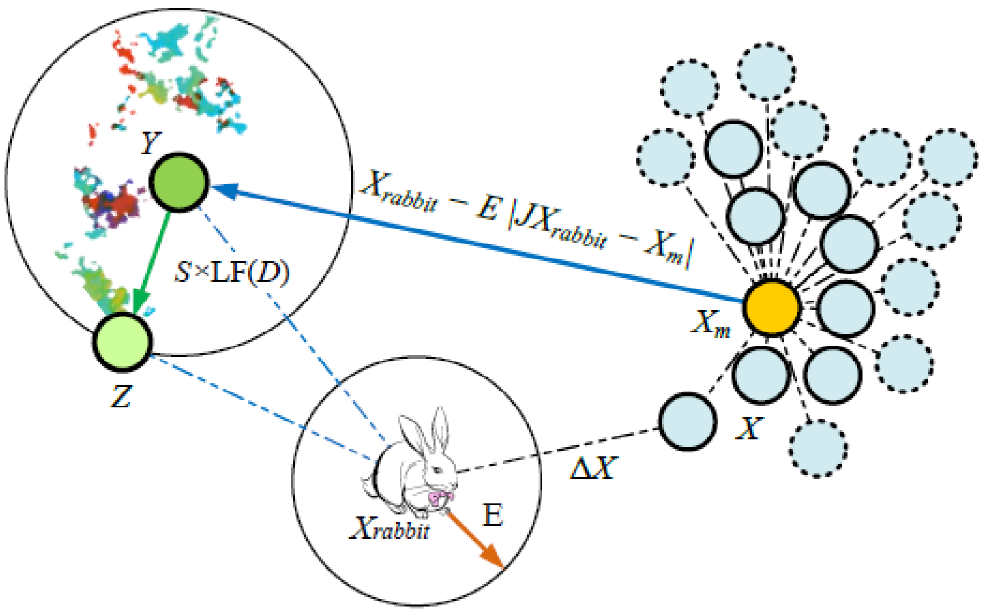

Y
$X_{rabbit} - E\,|JX_{rabbit} - X_m|$
$S{\times}\mathrm{LF}(D)$
X_m
Z
X
ΔX
X_{rabbit}
E

FIGURE 2.8: The process in 2D space [24]
[b]0.65

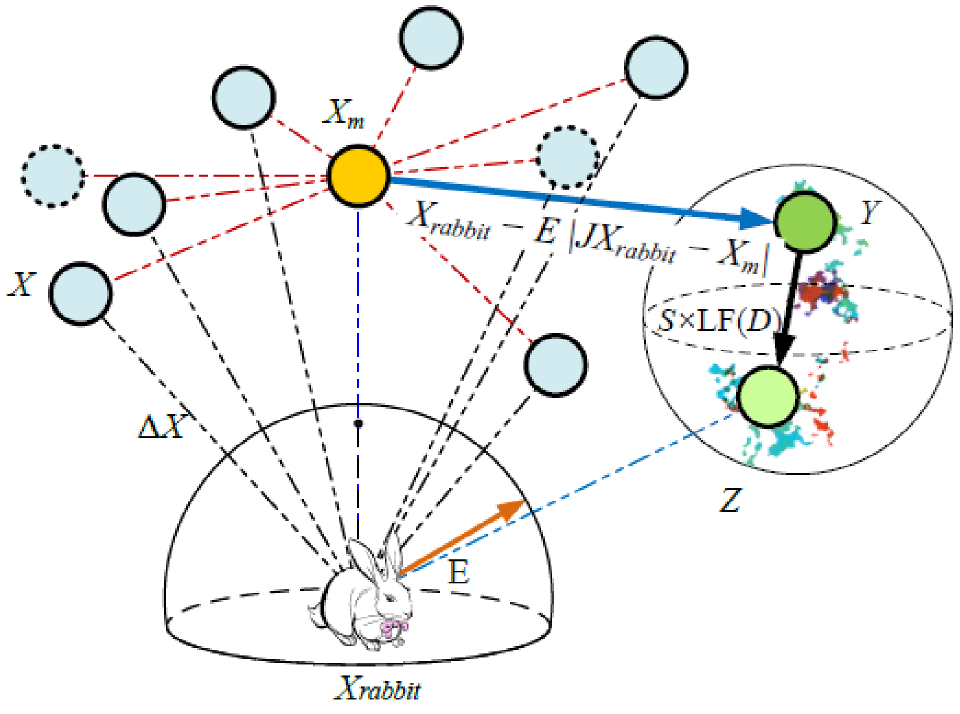

X_m
X
$X_{rabbit} - E\,|JX_{rabbit} - X_m|$
Y
$S{\times}\mathrm{LF}(D)$
ΔX
Z
E
X_{rabbit}

FIGURE 2.9: The process in 3D space [24]

FIGURE 2.10: Hard besiege with progressive rapid dives example in 2D and 3D

2.4.4 Pseudocode of HHO

The HHO algorithm pseudocode is reported in Algorithm 2.11.

Algorithm 1 HHO algorithm pseudocode [22]

```
**Inputs**: The size of population $N$ and maximum iterations number $T$
**Outputs**: The rabbit location and its fitness value
create an initial population values $X_i (i = 1, 2, \ldots, N)$
while (stopping condition is not met) do
    Calculate the hawks positions fitness values
    Set $X_{rabbit}$ as the rabbit location (best location)
    for (each hawk ($X_i$)) do
        Update the initial energy $E_0$ and jump strength $J$    ▷ E0=2rand()-1, J=2(1-rand())
        Update the $E$ using Eq. (2.3)
        if ($|E| \geq 1$) then    ▷ Exploration phase
            Update the location vector using Eq. (2.1)
        if ($|E| < 1$) then    ▷ Exploitation phase
            if ($r \geq 0.5$ and $|E| \geq 0.5$ ) then    ▷ Soft besiege
                Update the location vector using Eq. (2.4)
            else if ($r \geq 0.5$ and $|E| < 0.5$ ) then    ▷ Hard besiege
                Update the location vector using Eq. (2.6)
            else if ($r < 0.5$ and $|E| \geq 0.5$ ) then    ▷ Soft besiege with progressive rapid dives
                Update the location vector using Eq. (2.10)
            else if ($r < 0.5$ and $|E| < 0.5$ ) then    ▷ Hard besiege with progressive rapid dives
                Update the location vector using Eq. (2.11)
Return $X_{rabbit}$
```

FIGURE 2.11: HHO algorithm pseudocode [24]

Chapter 3

Related Works

For a large project, determining the requirements that should be implemented in the next software release becomes a complex problem. The complexity of this problem is because of dealing with two contradictory objectives (cost and profit). Having an incorrect decision may lead the company to run out of budget or lost essential customers. This problem motivated the attention of many researchers to find a solution to this problem.

Different methods are used to tackle the NRP. Most of the used methods fails in two categories [40]. The first one contains the Software Engineering Decision Support (SEDS) methods. Which is a research field that uses decision-making algorithms to tackle software engineering field problems. This category includes algorithms like Quality Function Deployment (QFD), Analytical Hierarchy Process (AHP) and fuzzy logic. It is usually focusing on ranking and prioritizing the requirements.

The other category is the Search Based Software Engineering (SBSE) methods. which is a research field that uses search based optimization algorithms to tackle software engineering problems [40]. linear programming, heuristic, and meta-heuristic algorithms are examples of the algorithms are belong to this category.

This chapter review the past related works to the problem we have, and how the researchers tackled it.

NRP is classified as an NP-hard problem according to Papadimitriou and Steiglitz [38], since it evaluates two conflicting objectives, it tries to find the best set of requirements that should be developed in the next software development iteration. The selection of this set of requirements should minimize the development cost and maximize the clients' satisfaction.

Karlsson in [28] suggested two methods to tackle NRP, which are Quality Function Deployment (QFD) and Analytical Hierarchy Process (AHP). In AHP, classification of requirements is a pair of [cost, value], but in QFD, requirements are prioritized on an ordinal scale. But, both methods don't handle requirement interactions, and they are not convenient in large projects duo to their long-running time.

Bagnall et al in [4] was formulated NRP for the first time in literature as a Single-Objective problem. They used three algorithms to tackle NRP. They compared Greedy Randomized Adaptive Search Procedure (GRASP), hill climber, and Simulated Annealing (SA) algorithms in finding the best combination of software requirements for different data sizes. They found that the SA achieved better performance than the GRASP and the hill climber, especially with large scale NRP instances [4].

The original problem proposed by Bagnall has been tackled by several researchers using different metaheuristics algorithms. However, most of these algorithms that were published before 2007 were based on Single-Objective evolutionary algorithms. Some of these works combined the objectives using aggregation function and tackled it as a Single-Objective such as [5, 22]. In [22], the authors used a genetic algorithm to introduce a new method to determine the optimal set of requirements. However, they didn't consider the interactions between these requirements. On the other hand, in [5], the authors tried to solve NRP using Greedy and SA algorithms, but they also didn't consider the interactions between the requirements.

In 2007, Zhang et al [49] proposed a Multi-Objective version of the NRP for the first time in literature. However, in this approach, each objective was tackled separately without considering any other objectives or constraints like interactions between the requirements or the cost limitations.

A trade-offs comparison between multiple clients using Multi-Objective optimization is used in [18, 19]. This research considered the potentially conflicting requirements priorities, but the intersection between requirements are not handled. Moreover, in [29, 16, 27], different Multi-Objective optimization algorithms were proposed without considering dependencies among requirements.

The authors in [27] combined the Hill Climbing (HC) algorithm with the Ant Colony Optimisation algorithm (ACO) to select the optimal subset of requirements. However, in [29], a quantum inspired evolutionary algorithm is proposed to tackle NRP.

In [15], the Pareto Archived Evolution Strategy (PAES), NSGA-II, and MOCell were employed in finding the optimal set of requirements.

The Ant Colony System (ACS) algorithm is employed by authors in [13] for solving NRP. The main point of this paper is that it employed five different ways to consider the interactions between the requirements that should be implemented in the next release. The main types that were considered in this paper are listed below:

- Precedence. $r_i \Rightarrow r_j$. A requirement r_i cannot be selected if a requirement r_j is not implemented.
- Combination. $r_i \otimes r_j$. A requirement r_i cannot be included without including a requirement r_j .
- Exclusion. $r_i \oplus r_j$. A requirement r_i can not be included with a requirement r_j in the same release.
- Revenue-based. The implementation of a requirement r_i will affect other requirements values.
- Cost-based. The development of a requirement r_i will affect other requirements implementation cost.

The Teaching-Learning-Based Optimisation (TLBO) algorithm was used in [10] to tackle the NRP. In that paper, they formulated NRP as a MOOP with two objectives: the total software requirement cost and overall customer satisfaction. Also, they considered three types of interactions between requirements.

In [9], authors developed Differential Evolution with the Pareto Tournament (DEPT) algorithm to tackle NRP. They used DEPT to search for high-quality sets of solutions in a predefined development effort and prioritize software requirements while considering requirements interactions. They made an enhancement on Differential Evolution (DE) algorithm that initially proposed in [44] to add the capability to deal with MOOP. However, they took the ideas of dominance and non-dominated sorting from NSGA-II . These ideas are used to sort the solutions and qualify thim within the population in a Multi-Objective environment. Moreover, they took the idea of a non-dominated solution archive (NDS-archive) from PAES. This NDS-archive is used to update the algorithm's best solutions after applying the solution to an acceptance function.

An interactive model for the NRP using the ACO algorithm was developed in [33]. In this model, the user can define which requirements should be included or excluded in the next release. So they used human expertise during the search to achieve better performance results than mathematical models.

In [3], the authors introduced an architecture based on genetic algorithm and machine learning to tackle NRP.

The Grey Wolf Optimisation (GWO) algorithm was used to select the best-proposed requirements in [31]. The authors compared their work with AHP algorithm, and they found that GWO performs better than the AHP mechanism by approximately (30%).

Glauber et al. [7] investigated solving the large scale NRP using two metaheuristic algorithms, ACO and the Particle Swarm Optimization (PSO). The researchers aimed to identify which metaheuristic algorithm is more suitable for handling the large scale NRP. They proved that the PSO algorithm achieved much better performance than ACO, and the PSO's best results were obtained with a higher number of particles. Moreover, their experiments showed that ACO could achieve the same performance as PSO for small sizes dataset.

In [7], the authors tackled NRP using two metaheuristic algorithms, the ACO and the PSO. In this work, researchers tried to find which metaheuristic algorithm achieves better performance in dealing with NRP. They proved that PSO achieved better performance than ACO, also the foundation that best PSO's result was obtained with a higher number of particles. On the other hand, they found that ACO can achieve the same performance as PSO for a small size of datasets.

In [1], the authors used NSGA-II and PSO. In this work, the qualification of requirements is determined by the three factors; dependencies between requirements, product integrity, and product value. They encoded the requirements model in a binary string. after that they tried to find possible solutions by using binary version of NSGA-II and PSO algorithms.

In [45], the authors proposed a mathematical formulation of the NRP.

This work handled non-additive customer valuations across requirements. They conduct experiments to test the performance of Multi-Objective evolutionary algorithms (MOEAs) in tackling NRP with non-additive valuations and implication constraints on requirements.

In [2], they proposed an algorithm based in fuzzy inference system to help in choosing the requirements for the next release. They compared the result of the experiment with a genetic algorithm. The generated solution by the proposed algorithm provided a better solution than the genetic algorithm.

In [8], A Fuzzy Multi-Objective Particle Swarm Optimization (FMOPSO) algorithm has been proposed. The authors conduct an experiment to compare the proposed algorithm with other state-of-the-art algorithms. They found that FMOPSO is adequate for finding very detailed Pareto Fronts comparing to different algorithms.

In [14], the authors proposed five methods to find a list solutions that is spread well at any time during the search, and the decision-maker has the ability stop the searching when there is an acceptable solution.

In [23], the authors proposed an Improved Binary Particle Swarm Optimization (IBPSO) algorithm to address the Multi-Objective constrained NRP. Also, they enhanced the approach by using a greedy methodology to seed the swarm with good solutions. They found that the enhanced approach achieved better performance than the original binary PSO.

Finally, the Binary Artificial Algae algorithm is proposed and employed for finding the optimal subset of requirements in [40], they also handled the case if customers' priorities is changed during the product development period.

From reviewing the related works, we found that different optimization methods are used in tackling NRP. Most of them didn't handle the requirement interactions. The other leak we found in the related works that most of the works tackled NRP as a Single-Objective problem. Moreover, some of the methods are not convenient in large projects due to their long-running time.

All the previous works are focused on finding the optimal requirements subset in minimal time, but they still didn't find an optimal way to fix NRP. Also, they tried to find a large variety of solutions to be provided for the decision-makers.

Chapter 4

Research Methodology

This research will follow the controlled experiment methodology in software engineering, which contains three main stages[39]. It started by deciding the problem by defining the objectives. After that, the design of how to achieve the defined objectives by defining the set of questions. Also, defining a set of tests to be conducted through the experiment as well as defining the quality indicators in order to evaluate the results[39].

After the design is ready, the preparation for the experiment started by collecting the data sets to be used for evaluating the proposed approach. After that, the software development phase started by converting HHO to Multi-Objective HHO and implement it in JMetal 5.6 Framework. Also, NRP is implemented as a Multi-Objective problem in JMetal 5.6.

The next step is comparing the new proposed approach with three of the best optimization algorithms. A set of commonly used quality indicators is used to evaluate the proposed approach and compare it with other algorithms.

4.1 Used Data Sets

The datasets used in this thesis are from related NRP literature [46]. It is available online for research on http://cstar.whu.edu.cn/p/nrp/ . It contains real and classic instances.

The classic data sets are generated in labs for experimental purposes. It includes the cost per each requirement and the dependency between the requirements. Moreover, the profit that will be gain from delivering customer requests of requirements. It contains 5 groups. Each group has different levels of requirements dependencies. For example, all the requirements in nrp-1 dataset are classified into 3 levels. Requirements in the 2nd level may depend on some requirements in the 1st level, while requirements in the 3rd level may depend on requirements in the 1st and 2nd levels. Figure 4.1 illustrates a sample of dependencies between the dataset's requirements.

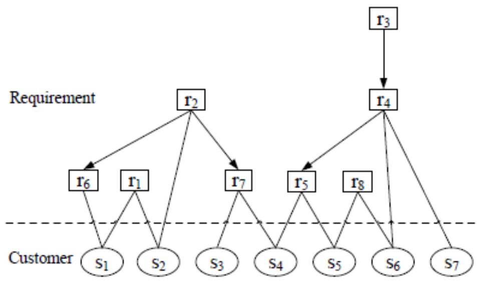


FIGURE 4.1: Requirements Dependency Sample [46]

Table 4.1 list the classic data set instances characteristics.

TABLE 4.1: Classic Data Set Instances Characteristics

Instance	# of Customers	# of Requirements	# of Dependencies	Total Profit
nrp1	100	140	97	875
nrp2	500	620	556	5048
nrp3	500	1500	1486	8870
nrp4	750	3250	4961	22161
nrp5	1000	1500	2036	3992

The second type of the data sets that used is the realistic data sets, it was derived from a real open source projects by the original authors, in this type there is no dependencies between the requirements [46], the data set instances characteristics appears in table 4.2.

According to dataset size, datasets categorization fails in five categories: the small, big, large, x-large, and xx-large size dataset instances as appears in table 4.3.

Two datasets are chosen to be used from classic datasets, and four datasets are chosen to be used from realistic datasets as appears in table 4.4.

TABLE 4.2: Realistic Data Set Instances Characteristics

Instance	# of Customers	# of Requirements	Total Profit
nrp-e1	536	3502	13150
nrp-e2	491	4254	15928
nrp-e3	456	2844	10399
nrp-e4	399	3186	11699
nrp-g1	445	2690	13277
nrp-g2	315	2650	12626
nrp-g3	423	2512	12258
nrp-g4	294	2246	10700
nrp-m1	768	4060	15741
nrp-m2	617	4368	16997
nrp-m3	765	3566	13800
nrp-m4	568	3643	14194

TABLE 4.3: Dataset Instances Size Categories

Category	Datasets	Requirement Size Range
Small	nrp1, nrp2	[100, 1000]
Medium	nrp3, nrp5	[1001, 2000]
Large	nrp-e3, nrp-g1, nrp-g2, nrp-g3, nrp-g4	[2001, 3000]
X-Large	nrp4, nrp-e1, nrp-e4, nrp-m3, nrp-m4	[3001, 4000]
XX-Large	nrp-e2, nrp-m1, nrp-m2	[4001, 5000]

TABLE 4.4: Chose Data Set Instances

Instance	# of Requirements	# of Dependencies	# of Customers	Total Profit
nrp-e1	3502	0	536	13150
nrp-e2	4254	0	491	15928
nrp-g1	2690	0	445	13277
nrp-g2	2650	0	315	12626
nrp-m1	4060	0	768	15741
nrp-m2	4368	0	617	16997
nrp1	140	97	100	875
nrp4	3250	4961	750	22161

4.2 The Proposed Approach

One of the most recently proposed SI algorithms is used in this work. A Multi-Objective version of the HHO algorithm is proposed and used to tackle NRP. A Single-Objective version of HHO using Java language is available online.

In the first version of the proposed approach; the HHO algorithm was converted from a Single-Objective to Multi-Objective using Fuzzy Logic methodology, and used to find the best set of requirements that achieved customers satisfaction within the limited development budget. In such a problem, requirement selection depends on the value, integrity, and dependencies between requirements. After several attempts of enhancements, we decided to use another way to convert the HHO algorithm from Single-Objective to Multi-Objective.

In the new version, HHO is converted from Single-Objective to Multi-Objective by creating two fitness functions; profit fitness function and cost fitness function.

As NRP is a Multi-Objective problem, two fitness functions are proposed. The first one is to calculate the profit for the generated solution. The other one is to calculate the cost of implementing the same solution. In each iteration, the two fitness functions should be evaluated for each member of the population list.

4.2.1 Convert HHO from Continuous to Discrete

The original HHO algorithm is implemented to deal with problems that have continued solution variables, but NRP has only two variables 0 and 1. Thus, the HHO algorithm in this thesis is enhanced to tackle such a problem. HHO is modified to tackle the binary problem by setting the maximum number for each bit as 1 and the minimum number as 0. After each step in executing the algorithm, a check is executed to check if the number is more than or equal 0.5; it will be rounded to 1, else it will be rounded to 0. As mentioned before, in the NRP problem, each solution consists of bits. If the bit is 1, the requirement will be included, and if the bit is 0, then the requirement will be excluded.

4.2.2 Cost Fitness Function

Each requirement in the data sets has an implementation cost. The total cost of the solution is the summation of cost for the selected requirements, the pseudo-code 1 describes the formula that used to calculate the cost for each solution.

Algorithm 1 Pseudo-code of cost fitness function

```
**Inputs**: The binary set of solution $X_i$
**Outputs**: The cost of implementing the solution
Initialize the cost of solution $total = 0$
for (each bit in the solution ($X_i$)) do
    if ($X_i > 0$) then
        $total = total + cost[i]$
Return $total$                                  ▷ Return the cost of solution
```

4.2.3 Profit Fitness Function

Software development companies have many customers. Each customer will pay some money if his requirements are implemented. Thus, the total profit that will be gain for the solution is the summation of profit that will be gain from a customer if the solution contains all the requirements requested by that customer. And since we have two contradictory objectives. One is minimized, and the other is maximized. The maximization objective will be multiplied by minus to simplify the comparison and make the two objectives minimized. The pseud-code 2 describes the fitness function to calculate the solution profit.

4.2.4 Comparing and Contrasting Solutions

Since each solution has two contradictory objectives (cost and profit) and since the better solution who is return highest profit and lowest cost, a

Algorithm 2 Pseudo-code of profit fitness function

```
Inputs: The binary set of solution $X_i$
Outputs: The profit of implementing the solution
Initialize the profit of solution $total = 0$
for (each bit in the solution ($X_i$)) do
    if ($X_i > 0$) then
        $total = total + profit[i]$
Return -1 * $total$                         ▷ Return the profit of solution
```

dominance comparator function is proposed to compare a solution with each others as appear in Algorithm 3.

Algorithm 3 Pseudo-code of dominance comparator function

```
Inputs: The binary set of solution $Solution1$ and $Solution2$
Outputs: -1, or 0, or 1 if $solution1$ dominates $solution2$, both are non-
dominated, or $solution1$ is dominated by $solution2$, respectively

Initialize the variable value $bestIsOne = 0$
Initialize the variable value $bestIsTwo = 0$
Initialize the variable value $result$
for (each $objective_i$ in the $solution1$) do
    if ($solution1\_objective_i$ != $solution2\_objective_i$) then
        if ($solution1\_objective_i$ < $solution2\_objective_i$) then
            $bestIsOne$ = 1
        else if ($solution2\_objective_i$ < $solution1\_objective_i$) then
            $bestIsTwo$ = 1
if ($bestIsOne$ > $bestIsTwo$) then
    Return -1                    ▷ Solution1 is better than Solution2
else if ($bestIsOne$ < $bestIsTwo$) then
    Return 1                     ▷ Solution2 is better than Solution1
else if ($bestIsOne$ == $bestIsTwo$) then
    Return 0                            ▷ both are non-dominated
```

4.2.5 MOHHO Implementation

The optimization process of MOHHO for NRP starts by generating a set of random solutions to form the population. After that, the algorithm starts the searching iterations. Each iteration begins with the exploration phase; in this phase, searching for the best solution is random. The best solution found in the exploration phase will be used in the next phase, which is the exploitation phase. In the exploitation phase, the algorithm will search in the neighborhood of the best solution found in the exploration phase. The Pseudocode 4.2 describes the proposed algorithm to tackle NRP.

Algorithm 5 Pseudo-code of MOHHO algorithm

```
Inputs: The population size $N$ and maximum number of iterations $T$
Outputs: The non-dominated solutions for NRP
Initialize the random population $X_i (i = 1, 2, \ldots, N)$
Calculate the fitness values of hawks using code 2 and 3
while (stopping condition is not met) do
    Calculate the fitness values of hawks using code 2 and 3
    for (each hawk ($X_i$)) do
        if $X_i$ dominate $X_{rabbit}$ then    ▷ compare using the pseudo code 4
            Set $X_{rabbit}$ as the location of $X_i$ (best solution)

        for (each hawk ($X_i$)) do
            Update the initial energy $E_0$ and jump strength $J$    ▷ E0=2rand()-1, J=2(1-rand())
            Update the $E$ using Eq. (2.3)
            if ($|E| \geq 1$) then    ▷ Exploration phase
                Update the location vector using Eq. (2.1)
            if ($|E| < 1$) then    ▷ Exploitation phase
                if ($r \geq 0.5$ and $|E| \geq 0.5$) then    ▷ Soft besiege
                    Update the location vector using Eq. (2.4)
                else if ($r \geq 0.5$ and $|E| < 0.5$) then    ▷ Hard besiege
                    Update the location vector using Eq. (2.6)
                else if ($r < 0.5$ and $|E| \geq 0.5$) then    ▷ Soft besiege with progressive rapid dives
                    Update the location vector using Eq. (2.10)
                else if ($r < 0.5$ and $|E| < 0.5$) then    ▷ Hard besiege with progressive rapid dives
                    Update the location vector using Eq. (2.11)

        Find the non-dominated solution
        Return non-dominated solution
```

FIGURE 4.2: Pseudo-code of MOHHO algorithm

For more clarification, the flowchart of MOHHO is added in Fig 4.3.

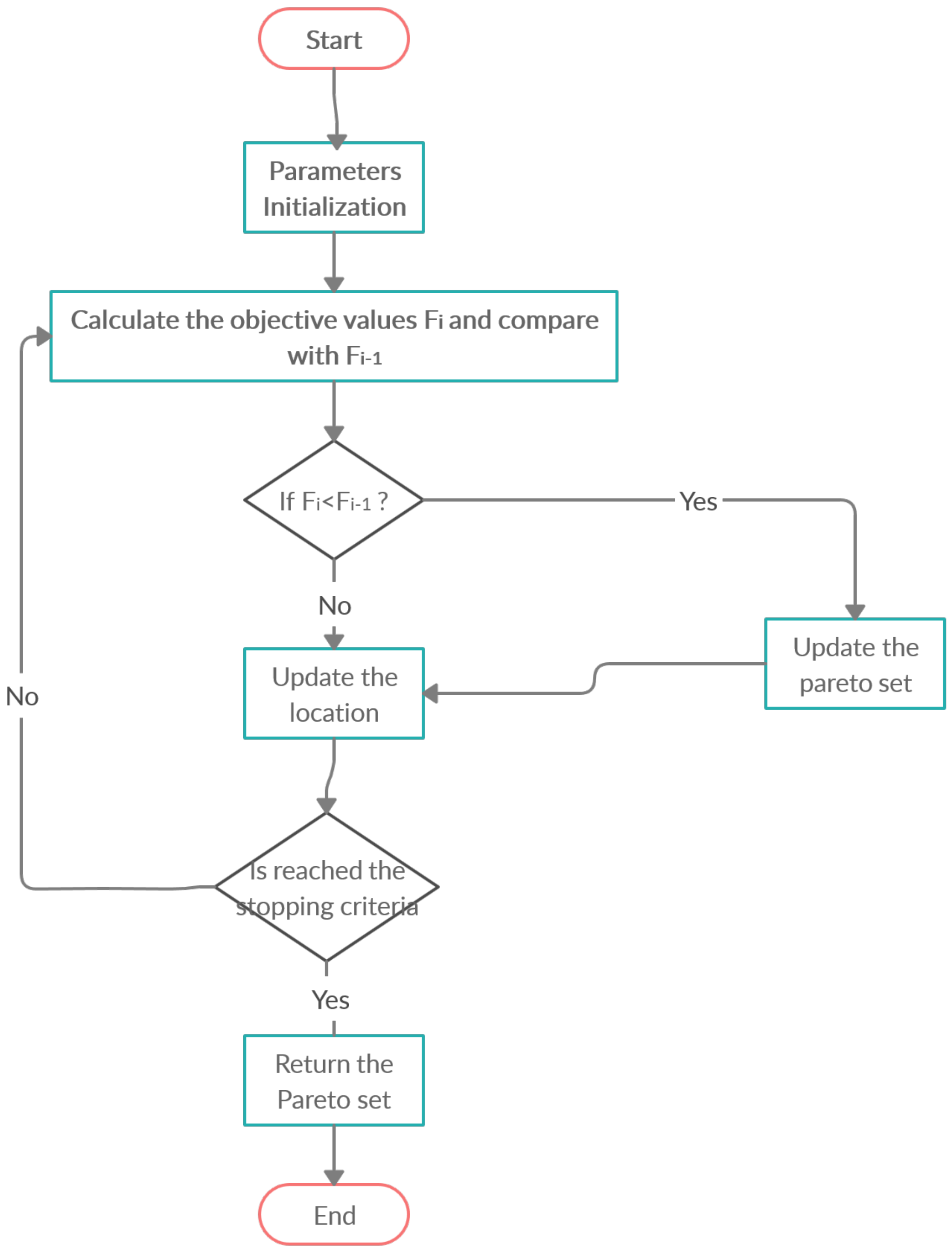

Start
Parameters Initialization
Calculate the objective values Fi and compare with Fi-1
If Fi<Fi-1 ?
Yes
No
Update the pareto set
Update the location
No
Is reached the stopping criteria
Yes
Return the Pareto set
End

FIGURE 4.3: MOHHO

4.3 Used Algorithms

Several optimization algorithms have tackled NRP. Some of these algorithms are Genetic Algorithms (GA), and some of them are Swarm Intelligent Algorithms (SIA). In this experiment, the proposed approach has compared with the most commonly used algorithms in multi-objective optimization problems. Also, the chosen algorithms have the best results from the multi-objective algorithms implemented in JMetal 5.4 framework in solving multi-objective problems.

4.3.1 NSGA-II

NSGA-II (Non-dominated Sorting Genetic Algorithm II) is a well-known genetic multi-objective algorithm [47]. It is used in different optimization problems and achieved a good reputation. NSGA-II has three special characteristics. It has a fast non-dominated sorting approach, fast crowding distance estimation procedure, and a simple crowding comparison operator. The Pseudocode of NSGA-II algorithm is added in Fig 4.4

Pseudocode of NSGA-IIr.

```
Input: $n$ // the population size
P ← Random_Population() // P = population
Q ← ∅                        // Q = auxiliar population
while not Termination_Condition() do
    for i ← 1 to ($n$) do
        randValue←rand();
        if (randValue $\leq$ 1/3) then
            parent←Selection1(P); // only one parent is selected
            offspring←PolynomialMutation(parent);
        else
            if (randValue $\leq$ 2/3) then
                parents←Selection2(P); // two parents are selected
                offspring←SBX(parents);
            else
                parents←Selection3(P); // three parents are selected
                offspring←DE(population[i], parents);
            end if
        end if
        Evaluate_Fitness(offspring);
        Insert(offspring,Q);
    end for
    R ← P ∪ Q
    Ranking_And_Crowding(R);
    P ← Select_Best_Individuals(R)
end while
Return P;
```

FIGURE 4.4: NSGA-II Code

4.3.2 MOCell

MOCell (Multi-Objective Cellular Genetic Algorithm), it is proposed by Nebro et al in [35], which is a cellular genetic algorithm (cGA).

In this algorithm, each cell can cooperate only with its nearby neighbors in the breeding loop. This can be shown in Figure 4.5.

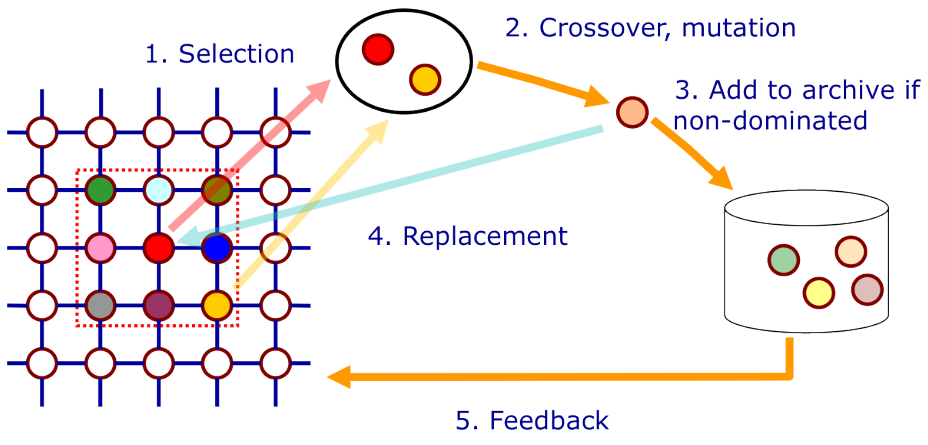
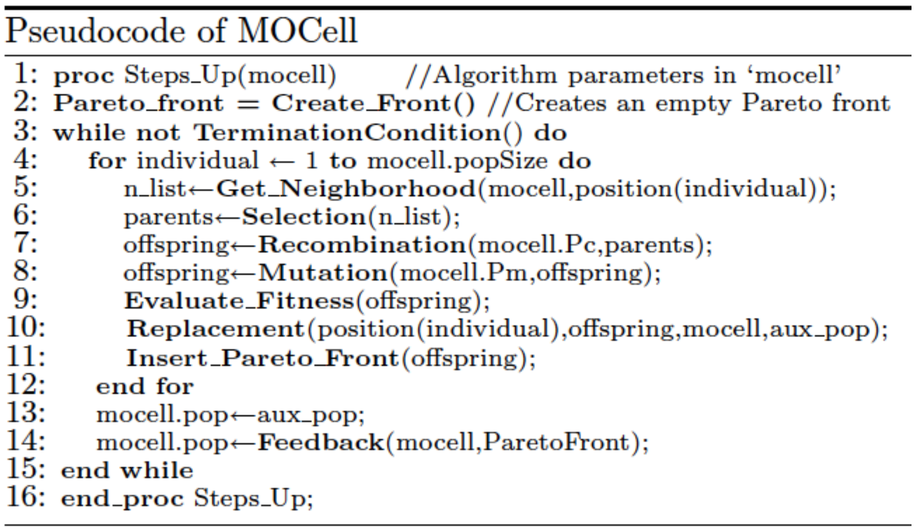


FIGURE 4.5: MOCell [35]

The exploration phase in cGAs depends mainly on the overlapped small neighborhoods, which help in exploring the search space (diversification). The exploitation (intensification) phase happens for each neighborhood by genetic operations. MOCell also tries to find the non-dominated solutions and store them in an external archive. This archive which has a limited size uses the crowding distance that used in NSGA-II to keep diversity in the Pareto Front.



Pseudocode of MOCell

```
proc Steps_Up(mocell)       //Algorithm parameters in 'mocell'
Pareto_front = Create_Front() //Creates an empty Pareto front
while not TerminationCondition() do
    for individual ← 1 to mocell.popSize do
        n_list←Get_Neighborhood(mocell,position(individual));
        parents←Selection(n_list);
        offspring←Recombination(mocell.Pc,parents);
        offspring←Mutation(mocell.Pm,offspring);
        Evaluate_Fitness(offspring);
        Replacement(position(individual),offspring,mocell,aux_pop);
        Insert_Pareto_Front(offspring);
    end for
    mocell.pop←aux_pop;
    mocell.pop←Feedback(mocell,ParetoFront);
end while
end_proc Steps_Up;
```

FIGURE 4.6: MOCEllCode

4.3.3 MOCHC

MOCHC is a multi-objective version of CHC, which is an evolutionary algorithm proposed by Eshelman in [17].

This algorithm aims to tackle binary-coded problems. The main feature in this algorithm is the use of HUX crossover mechanism.

A multi-objective version of CHC (MOCHC) is proposed by Nebro et al. in [36] to tackle Radio network design (RND) problem. They compared the proposed approach with NSGA-II in order to assess the performance. MOCHC has proven to be more efficient than NSGA-II.

```
Pseudocode of CHC
$t \leftarrow 0$
Initialize($Pa$, $convergence\_count$, k) // $Pa$: population
while not ending_condition($t$, $Pa$) do
    $Parents \leftarrow$ Selection_parents($Pa$, $convergence\_count$)
    $Offspring \leftarrow$ HUX($Parents$)
    Evaluate($Offspring$)
    $Pn \leftarrow$ Elitist_selection($Offspring$, $Pa$) // $Pn$: new pop.
    if not modified($Pa$,$Pn$) then
        $convergence\_count \leftarrow convergence\_count - 1$
        if $convergence\_count \leq -k$ then
            $Pn \leftarrow$ Restart($Pa$)
            Initialize($convergence\_count$)
        end if
    end if
    $t \leftarrow t + 1$
    $Pa \leftarrow Pn$
end while
```

FIGURE 4.7: CHCCode

4.4 Quality of the Solutions Obtained

In this experiment, Hypervolume quality indicator is used to evaluate the solutions obtained from running the experiment. Also Friedman and Wilcoxon are used as statistical analysis.

4.4.1 Hypervolume

Hypervolume (HV) quality indicator is categorized as a single unary value that measure the spread of the obtained solution along the Pareto front, also the closeness of the solution to the Pareto-optimal front. In other words, it is an algorithm performance measure, it measures the convergence of the algorithm throw the experiment execution, and the diversity of the obtained solutions [41]. Given that a list Q contains the vectors of solutions, the algorithm calculates the hypercube value where the intersection occurs, also add this value to a partial sum value as appears in formula 4.8.

$$HV = volume\left(\bigcup_{i=1}^{|Q|} v_i\right)$$

FIGURE 4.8: Hypervolume

The total sum value will be the Hypervolume union, which represents the intersections between each vector in the solution and a reference point [41]. The algorithms with higher HV value is better than the algorithms with lower HV value.

4.4.2 Statistical Analysis

In each experiment some of statistical analysis should be measured to make sure that the obtained results cannot have occurred by chance.

Friedman Statistical Analysis

The Friedman test is a non-parametric statistical test developed by Milton Friedman [20]. Friedman statistical test is used multiple test attempts to detect differences in treatments.

This statistical test can be used to approved the null hypothesis that different of group measures have same variant to a certain level of significance. Otherwise,it means these group measures have different variance values.

jMetal has the capability to calculate the Friedman statistical test.

Wilcoxon Statistical Analysis

The Wilcoxon test is a non-parametric statistical test. It is used to compare two related samples and test if they are statistically significant or not [12].

This test can be used to tell if the results between the optimization algorithms are statistically significance and did not occur randomly or by chance.

Two symbols are used (▽ ▲) to indicate that the results are statistically significant when $p-value$ < 5% between two optimization algorithms for each data set.

The black triangle (▲) means that there is a relationship between X & Y; also, X performance is better than Y. However, down triangle (▽) means that Y performance is better than X. Moreover, a dash symbol (–) indicates that results for a given test case are not statistically significant, and there is no difference between using X & Y.

Chapter 5

Results

In this chapter, the experiment parameters used in the experiment were presented. Also the used datasets are mentioned, followed by description for the quality indicator used to test and compare used algorithm's performance. Finally, discussion and analysis of the experiment results.

5.1 Experimental Setup

In this thesis, all experiments were run on a machine with windows 10 64 bit, Core(TM) i7-3520M CPU @ 2.90GHz (4 CPUs) and 8GB RAM, and MOHHO algorithm is implemented using jMetal 5.6 above Java.

A jMetal is designed to run an experiment with multiple datasets.It is also has the capability to run different algorithms and setup the configuration for each algorithm independently. Moreover, it can calculate lots of quality indicators and generate meaningful statistics for the experiment like Wilcoxon and Friedman statistics. Also, generate graphs like box plots to compare the algorithms, and provide the needed data to draw the convergence graph.

The used parameters for the experiment are explained in table 5.1

TABLE 5.1: MOHHO Parameters

Parameters	Values
Number of iterations	50
Population number	100
Crossover rate	0.9
Mutation rate	0.35
Number of independent runs	25

The execution has been repeated twenty-five times for each algorithm on each dataset to reduce the probability that an algorithm started with a bad solution and optimized a bad solution or started by luck in a good solution. In SI algorithms, the initial solution is generated randomly. So the initial state is important. Generally, SI algorithms try to solve the problem of getting stuck in local optima. However, the initial solution that the algorithms start with may affect the algorithm accuracy.

TABLE 5.2: HV. Median and Interquartile Range

	MOHHO	NSGAII	MOCell	MOCHC
nrp-e1	$3.41e-01_{1.1e-01}$	$2.45e-01_{1.4e-02}$	$2.44e-01_{1.9e-02}$	$2.39e-01_{1.8e-02}$
nrp-e2	$2.48e-01_{1.3e-01}$	$8.24e-02_{1.3e-02}$	$8.84e-02_{1.9e-02}$	$8.41e-02_{1.6e-02}$
nrp-g1	$3.13e-01_{1.9e-01}$	$2.38e-01_{1.2e-02}$	$2.40e-01_{1.1e-02}$	$2.33e-01_{1.4e-02}$
nrp-g2	$2.41e-01_{2.9e-02}$	$1.93e-01_{1.7e-02}$	$1.98e-01_{1.3e-02}$	$1.92e-01_{1.7e-02}$
nrp-m1	$3.07e-01_{1.3e-01}$	$7.82e-02_{1.7e-02}$	$8.04e-02_{2.9e-02}$	$7.29e-02_{2.2e-02}$
nrp-m2	$2.38e-01_{1.3e-01}$	$0.00e+00_{1.7e-02}$	$5.87e-03_{1.4e-02}$	$0.00e+00_{1.3e-02}$
nrp1	$4.48e-01_{2.6e-01}$	$1.98e-01_{2.8e-02}$	$2.05e-01_{2.5e-02}$	$1.95e-01_{1.7e-02}$
nrp4	$2.57e-01_{1.3e-01}$	$1.71e-01_{1.4e-02}$	$1.79e-01_{1.9e-02}$	$1.69e-01_{1.6e-02}$

5.2 Experimental Results

In this section a deep analysis of the obtained results is presented.

5.2.1 HV Quality Indicator Median

The obtained results are compared using HV quality indicator. Four different algorithms are used with eight different dataset sizes. Each experiment is repeated 25 times to get accurate results, and the median of the 25 runs is presented for each data set as appears in table 5.2.

Table 5.2 shows HV median values for small, medium, large datasets. It represents the median result of 25 independent runs. Inspecting HV Median values, the following observations can be discussed:

1. MOHHO outperformed NSGA-II, MOCell and MOCHC: Grey cells in the tables 5.2 points to the algorithm with higher HV per dataset. Out of the 8 datasets of different sizes; 8 times MOHHO HV values were better than NSGA-II, MOCell and MOCH. MOHHO outperformed other algorithms perfectly for the NRP in different dataset sizes.

2. MOCell outperformed NSGA-II, and MOCHC. Out of the 8 datasets of different sizes; 7 times MOCell HV values were better than NSGA-II and MOCH.

3. NSGA-II outperformed MOCell and MOCHC for nrp-e1 dataset, which is x-larg data set. So NSGA-II can achieve good reputation for large scale data sets.

From the previous observations, we can conclude that the MOHHO outperformed the other algorithms in all cases. It has the highest HV value for different dataset sizes (number of requirements and customers) and different dataset types (with dependencies and without dependencies between requirements).

5.2.2 Friedman Statistical

JMetal 5.4 framework has the capability to calculate the Friedman statistical test. Which is a non-parametric statistical test developed by Milton Friedman. In this experiment, Friedman test is calculated in order to compare the performance of the new proposed algorithm and the other algorithms. The best algorithm is which has the highest average ranking value. Table 5.3 shows that MOHHO performed better than the other algorithms.

Friedman statistic considering reduction performance (distributed according to chi-square with 3 degrees of freedom: 20.7).

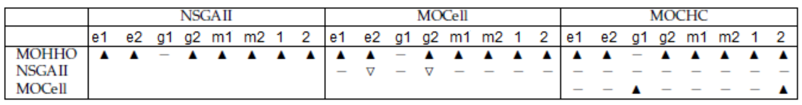
TABLE 5.3: Average ranking of the algorithms

Algorithm	Ranking
MOHHO	3.875
NSGAII	1.625
MOCell	3.125
MOCHC	1.375

5.2.3 Wilcoxon Statistical

The Wilcoxon test is a non-parametric statistical test. It is used to compare two related samples and test if they are statistically significant or not [12].

The results of statistical tests using Wilcoxon were collected and aggregated so that we can tell if the results between the optimization algorithms are statistically significance and did not occur randomly or by chance.



	NSGAII								MOCell								MOCHC							
	e1	e2	g1	g2	m1	m2	1	2	e1	e2	g1	g2	m1	m2	1	2	e1	e2	g1	g2	m1	m2	1	2
MOHHO	▲	▲	–	▲	▲	▲	▲	▲	▲	▲	–	▲	▲	▲	▲	▲	▲	▲	–	▲	▲	▲	▲	▲
NSGAII									–	▽	–	▽	–	–	–	–	–	–	–	–	–	–	–	–
MOCell																	–	–	▲	–	–	–	–	▲

FIGURE 5.1: Wilcoxon Statistical Test

From table 5.1, the following observations can be discussed:

1. MOHHO is statistically significant with NSGAII, MOCell and MOCHC for 7 out of 8 datasets, also it performed better than NSGAII, MOCell and MOCHC for the 7 datasets.

2. NSGAII is statistically significant with MOCell for 2 out of 8 datasets, also it performed worse than MOCell for the 2 datasets.
3. There is no statistically significant between NSGAII and MOCHC.
4. MOCell is statistically significant with MOCHC for 2 out of 8 datasets, also it performed better than MOCHC for the 2 datasets.

From the previous observations, we can conclude that the MOHHO algorithm outperforms the other algorithms using Wilcoxon statistical analysis.

5.2.4 Box Plots

In order the compare the obtained result that presented in table 5.2, box plots charts are used. Box plots is generated in order to have a clear comparison between the algorithms.

Moreover; box plots were used to help us in understand the distribution characteristics of HV values for each dataset per each one of the algorithms. Figures 5.2, 5.3 illustrates box plots for a representative datasets.

The following information from the box plots can be extracted:

1. HHO outperformed the other algorithms in terms of the maximum median HV value can be obtained for 8 datasets for 25 runs as appears in table 5.2, maximum global values obtained by each algorithm for all dataset categories. Cells highlighted in dark grey means that this value is the maximum value for the data set.

2. NSGAII and MOCell compete in terms of the maximum median HV value can be obtained. NSGAII was performed better than MOCell for nrp-e1 dataset. On the other hand, MOcel performed better on the other 7 data sets.

3. MOCHC algorithm achieved the minimum HV value for all datasets.

4. Data centering for MOHHO is higher than other algorithms as appear in figures 5.2, 5.3; this means that MOHHO can allocate promising solutions faster than other algorithms.

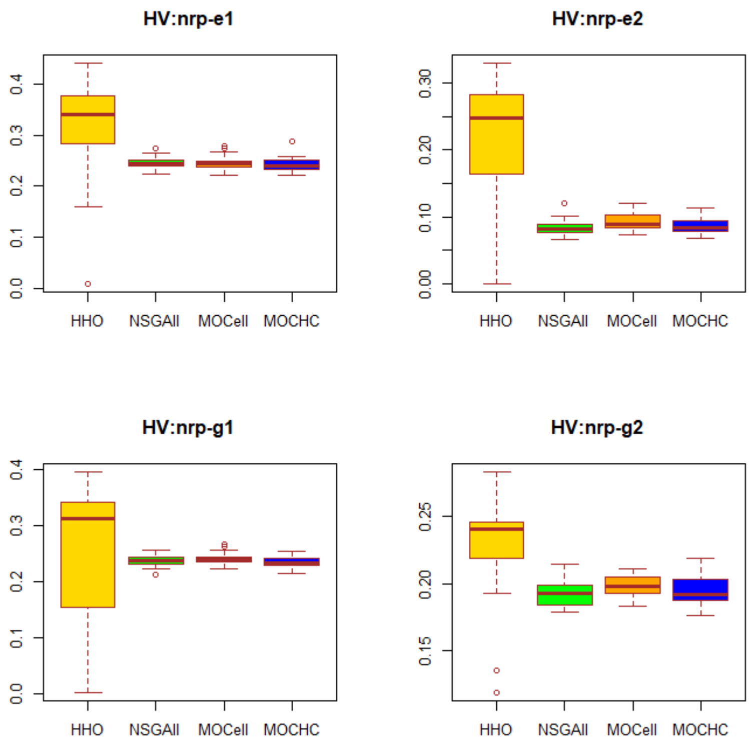


FIGURE 5.2: Box Plots group1

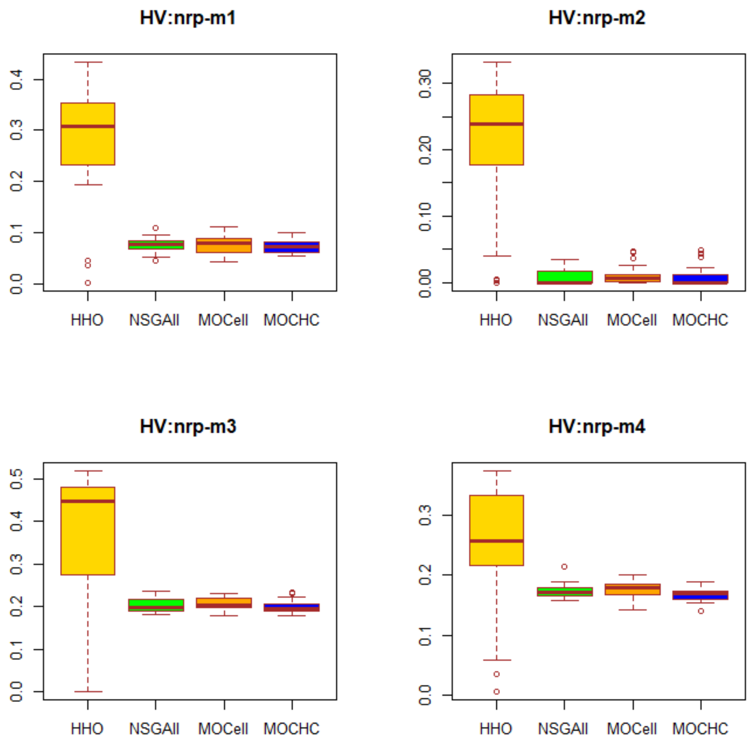


FIGURE 5.3: Box Plots group2

5.2.5 Convergence Curve

The convergence curve figure is used to compare the convergence speed of the algorithms. It is also used to find the algorithm that can allocate better solutions in fewer iterations. As appears in figures 5.4, 5.5, 5.6, 5.7, 5.8, 5.9, 5.10 and 5.11, MOHHO can allocate perfect solutions better and faster than other algorithms. It is started with imperfect solutions, but after a few iterations, it can reach promising solutions. This happens because the MOHHO algorithm populations learn from each other.

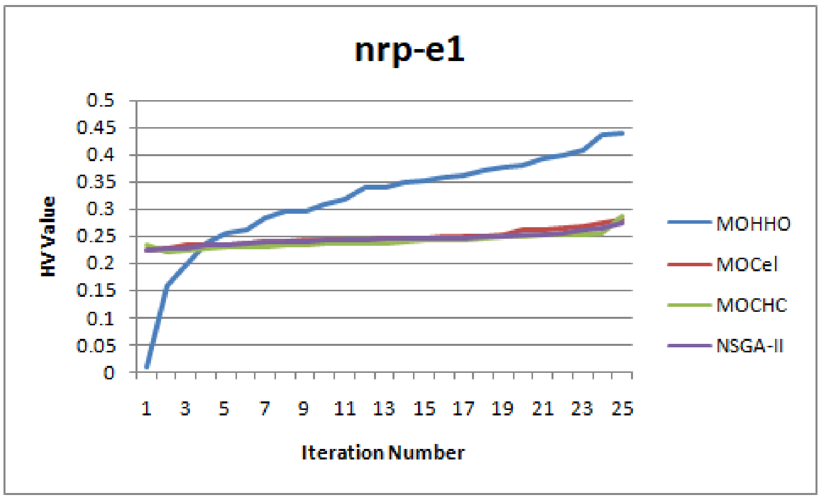


FIGURE 5.4: Convergence Curve nrp-e1

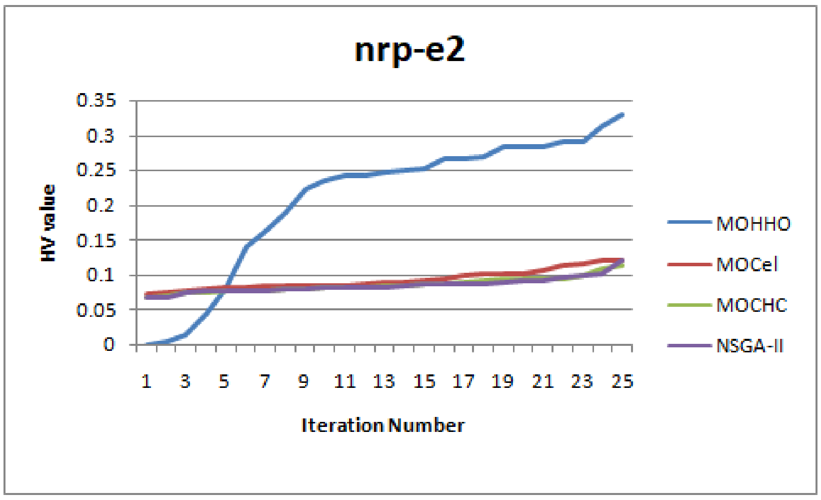


FIGURE 5.5: Convergence Curve nrp-e2

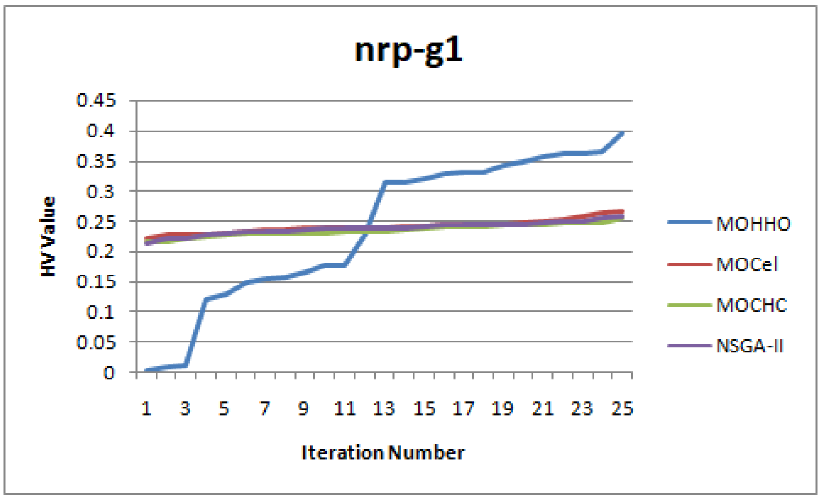


FIGURE 5.6: Convergence Curve nrp-g1

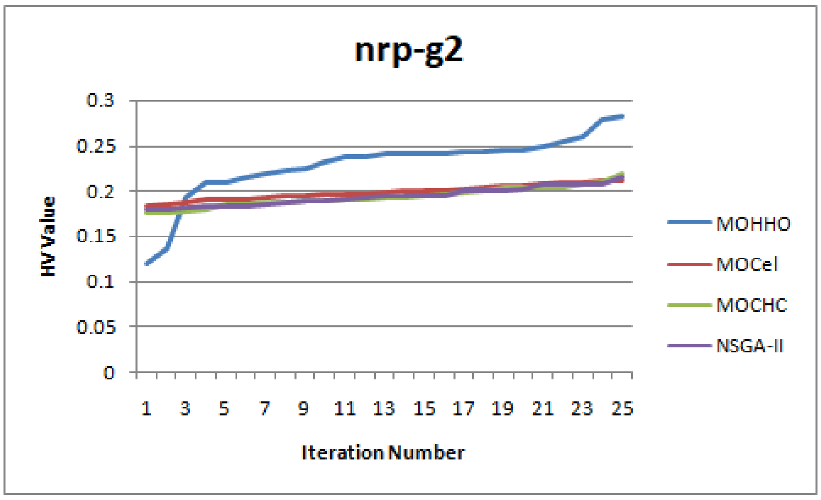


FIGURE 5.7: Convergence Curve nrp-g2

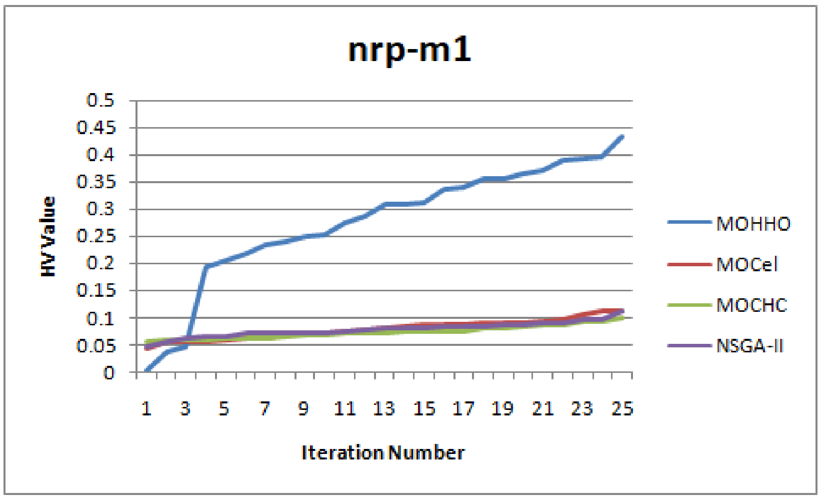


FIGURE 5.8: Convergence Curve nrp-m1

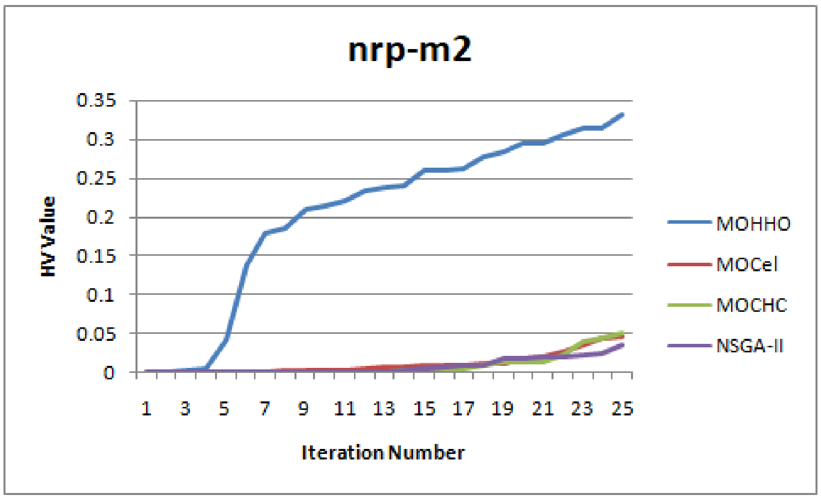


FIGURE 5.9: Convergence Curve nrp-m2

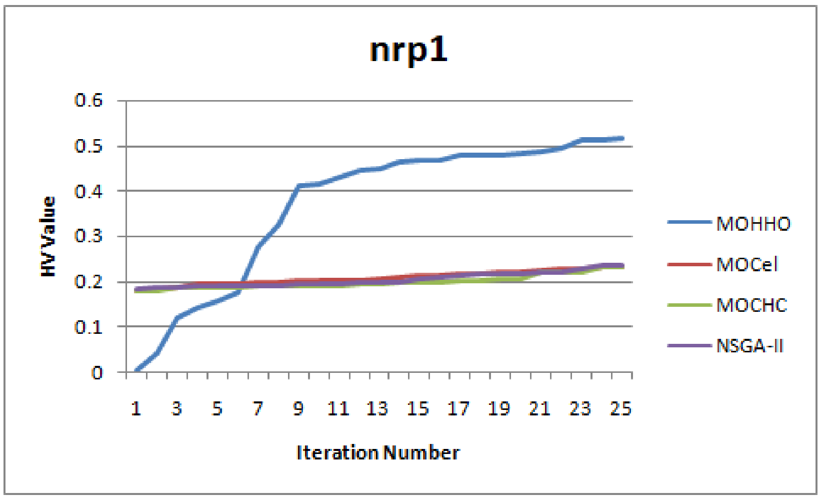


FIGURE 5.10: Convergence Curve nrp-1

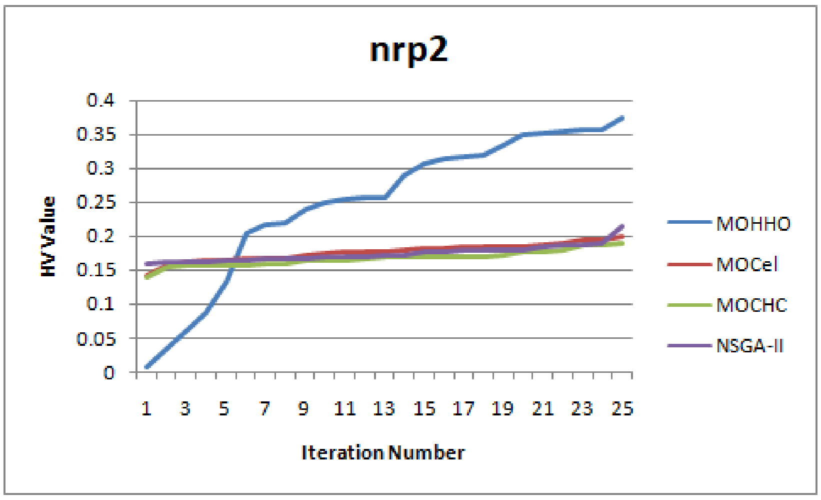


FIGURE 5.11: Convergence Curve nrp-2

From the previous performance measuring, we can notice that MOHHO is performed better than other solutions in many factors. It can allocate perfect solutions faster than other algorithms, also outperform the other algorithms in the diversity of the obtained solutions.

Chapter 6

Conclusion and Future Work

In this work, a set of Meta-Heuristics algorithms have been used to tackle NRP. Different quality indicators are used in comparing the results. This chapter presents a conclusion of the obtained results and highlight the future work.

6.1 Conclusion

Software development companies usually need to implement a lot of requirements in a limited time; since they can't implement all requirements in the next release, they need to choose a subset of requirements to be implemented in the next release [4].

In this problem, the selection of requirements depends on requirement value, requirement integrity, and dependencies between requirements.

In this thesis, A Multi-Objective version of HHO is proposed to tackle NRP and find the best set of requirements that achieved customer satisfaction and gain a high profit within the limited development budget.

Different datasets from related NRP literature are used to assess the proposed approach's performance. It contains classic and real instances. On the first hand, classic datasets were generated in labs for experimental purposes. On the other hand, the realistic datasets were derived from a real open source project. Both types include requirements cost, requirements requested by each customer, and the profit will be gained for implementing customer requirements.

Three of the best meta-heuristics algorithms (NSGA-II, MOCell, and MOCHC) are used in addition to the proposed algorithm MOHHO to tackle NRP using different datasets. Each experiment is repeated several times to avoid randomness in the founded results.

The obtained results showed that MOHHO outperforms NSGA-II, MOCell, and MOCHC based on comparing Hypervolume quality indicator values. Also, MOHHO has a better learning curve and can reach promising solutions in a few steps comparing to the other algorithms.

6.2 Future Work

SI algorithms are performed a good reputation in many optimization problems. It is also can allocate good results in a low number of iterations. MOHHO can perform better if it has some enhancements to avoid stucking in a local-optimal solution. Since most of the algorithms in JMetal belong to the Evolutionary algorithms, we suggest applying more SI algorithms and make a deep comparison between the performance of these two families of Meta Heuristic algorithms. Another room of enhancement can be applied by tuning the MOHHO algorithm parameters. Also, it worth to add more objectives in NRP, execution time can be considered as a third objective.

Bibliography

[1] Abdullah Al Mamun, Fahim Djatmiko, and Mridul Kanti Das. "Binary multi-objective PSO and GA for adding new features into an existing product line". In: *2016 19th International Conference on Computer and Information Technology (ICCIT)*. IEEE. 2016, pp. 581–585.

[2] Hamidreza Alrezaamiri, Ali Ebrahimnejad, and Homayun Motameni. "Solving the next release problem by means of the fuzzy logic inference system with respect to the competitive market". In: *Journal of Experimental & Theoretical Artificial Intelligence* (2019), pp. 1–18.

[3] Allysson Allex Araújo et al. "An architecture based on interactive optimization and machine learning applied to the next release problem". In: *Automated Software Engineering* 24.3 (2017), pp. 623–671.

[4] Anthony J. Bagnall, Victor J. Rayward-Smith, and Ian M Whittley. "The next release problem". In: *Information and software technology* 43.14 (2001), pp. 883–890.

[5] Paul Baker et al. "Search based approaches to component selection and prioritization for the next release problem". In: *2006 22nd*

IEEE International Conference on Software Maintenance. IEEE. 2006, pp. 176–185.

[6] James C Bednarz. "Cooperative hunting in Harris' hawks (Parabuteo unicinctus)". In: *Science* 239.4847 (1988), p. 1525.

[7] Glauber Botelho et al. "Investigating Bioinspired Strategies to Solve Large Scale Next Release Problem." In: *CIbSE*. 2015, p. 248.

[8] Carlos Casanova et al. "Fuzzy Bi-Objective Particle Swarm Optimization for Next Release Problem". In: *International Conference on Software Engineering and Knowledge Engineering*. 2019, pp. 509–512.

[9] José M Chaves-González and Miguel A Pérez-Toledano. "Differential evolution with Pareto tournament for the multi-objective next release problem". In: *Applied Mathematics and Computation* 252 (2015), pp. 1–13.

[10] José M Chaves-González, Miguel A Pérez-Toledano, and Amparo Navasa. "Teaching learning based optimization with Pareto tournament for the multiobjective software requirements selection". In: *Engineering Applications of Artificial Intelligence* 43 (2015), pp. 89–101.

[11] Carlos A Coello Coello, Gary B Lamont, David A Van Veldhuizen, et al. *Evolutionary algorithms for solving multi-objective problems*. Vol. 5. Springer, 2007.

[12] Jack Cuzick. "A Wilcoxon-type test for trend". In: *Statistics in medicine* 4.1 (1985), pp. 87–90.

[13] José Del Sagrado, Isabel M Del Águila, and Francisco J Orellana. "Multi-objective ant colony optimization for requirements selection". In: *Empirical Software Engineering* 20.3 (2015), pp. 577–610.

[14] Miguel Ángel Domınguez-Rıos et al. "Efficient anytime algorithms to solve the bi-objective Next Release Problem". In: *Journal of Systems and Software* 156 (2019), pp. 217–231.

[15] Juan J Durillo et al. "A study of the bi-objective next release problem". In: *Empirical Software Engineering* 16.1 (2011), pp. 29–60.

[16] Juan J Durillo et al. "A study of the multi-objective next release problem". In: *2009 1st International Symposium on Search Based Software Engineering*. IEEE. 2009, pp. 49–58.

[17] Larry J Eshelman. "The CHC adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination". In: *Foundations of genetic algorithms*. Vol. 1. Elsevier, 1991, pp. 265–283.

[18] Anthony Finkelstein et al. ""Fairness analysis" in requirements assignments". In: *2008 16th IEEE International Requirements Engineering Conference*. IEEE. 2008, pp. 115–124.

[19] Anthony Finkelstein et al. "A search based approach to fairness analysis in requirement assignments to aid negotiation, mediation and decision making". In: *Requirements engineering* 14.4 (2009), pp. 231–245.

[20] Milton Friedman. “The use of ranks to avoid the assumption of normality implicit in the analysis of variance”. In: *Journal of the american statistical association* 32.200 (1937), pp. 675–701.

[21] FW Glover and GA Kochenberger. *editros. Handbook of metaheuristics*. 2003.

[22] Des Greer and Guenther Ruhe. “Software release planning: an evolutionary and iterative approach”. In: *Information and software technology* 46.4 (2004), pp. 243–253.

[23] A Hamdy and AA Mohamed. “Greedy Binary Particle Swarm Optimization for multi-Objective Constrained Next Release Problem”. In: *International Journal of Machine Learning and Computing* 9.5 (2019).

[24] Ali Asghar Heidari et al. “Harris hawks optimization: Algorithm and applications”. In: *Future generation computer systems* 97 (2019), pp. 849–872.

[25] Watts S Humphrey. *A discipline for software engineering*. Addison-Wesley Longman Publishing Co., Inc., 1995.

[26] Kashif Hussain, William Zhu, and Mohd Najib Mohd Salleh. “Long-term memory Harris’ hawk optimization for high dimensional and optimal power flow problems”. In: *IEEE Access* 7 (2019), pp. 147596–147616.

[27] He Jiang et al. “A hybrid ACO algorithm for the next release problem”. In: *The 2nd International Conference on Software Engineering and Data Mining*. IEEE. 2010, pp. 166–171.

[28] Joachim Karlsson. “Software requirements prioritizing”. In: *Proceedings of the Second International Conference on Requirements Engineering*. IEEE. 1996, pp. 110–116.

[29] A Charan Kumari, K Srinivas, and MP Gupta. “Software requirements optimization using multi-objective quantum-inspired hybrid differential evolution”. In: *Evolve-a bridge between probability, set oriented numerics, and evolutionary computation ii*. Springer, 2013, pp. 107–120.

[30] Majdi M Mafarja and Seyedali Mirjalili. “Hybrid whale optimization algorithm with simulated annealing for feature selection”. In: *Neurocomputing* 260 (2017), pp. 302–312.

[31] Raja Masadeh et al. “Grey Wolf algorithm for requirements prioritization”. In: *Modern Applied Science* 12.2 (2018), p. 54.

[32] Victor JAT de Melo França et al. “Mixed Integer Programming helping Requirements Allocation for the NRP in SCRUM Teams”. In: *Proceedings of the 17th Brazilian Symposium on Software Quality*. 2018, pp. 279–286.

[33] Thiago do Nascimento Ferreira et al. “Incorporating user preferences in ant colony optimization for the next release problem”. In: *Applied Soft Computing* 49 (2016), pp. 1283–1296.

[34] Antonio J Nebro, Juan J Durillo, and Matthieu Vergne. “Redesigning the jMetal multi-objective optimization framework”. In: *Proceedings of the companion publication of the 2015 annual conference on genetic and evolutionary computation*. 2015, pp. 1093–1100.

[35] Antonio J Nebro et al. “MOCell: A cellular genetic algorithm for multiobjective optimization”. In: *International Journal of Intelligent Systems* 24.7 (2009), pp. 726–746.

[36] Antonio J Nebro et al. “Optimal antenna placement using a new multi-objective CHC algorithm”. In: *Proceedings of the 9th annual conference on Genetic and evolutionary computation*. 2007, pp. 876–883.

[37] Frauke Paetsch, Armin Eberlein, and Frank Maurer. “Requirements engineering and agile software development”. In: *WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003.* IEEE. 2003, pp. 308–313.

[38] Christos H Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998.

[39] Shari Lawrence Pfleeger. “Experimental design and analysis in software engineering”. In: *Annals of Software Engineering* 1.1 (1995), pp. 219–253.

[40] Poria Pirozmand et al. “A novel approach for the next software release using a binary artificial algae algorithm”. In: *Journal of Intelligent & Fuzzy Systems* Preprint (), pp. 1–15.

[41] Aurora Ramirez, Jose Raul Romero, and Christopher L Simons. "A systematic review of interaction in search-based software engineering". In: *IEEE Transactions on Software Engineering* 45.8 (2018), pp. 760–781.

[42] S Salcedo-Sanz. "Modern meta-heuristics based on nonlinear physics processes: A review of models and design procedures". In: *Physics Reports* 655 (2016), pp. 1–70.

[43] Ken Schwaber and Jeff Sutherland. "The scrum guide". In: *Scrum Alliance* 21 (2011), p. 19.

[44] Rainer Storn and Kenneth Price. "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces". In: *Journal of global optimization* 11.4 (1997), pp. 341–359.

[45] Ashish Sureka. "Requirements prioritization and next-release problem under non-additive value conditions". In: *2014 23rd Australian Software Engineering Conference*. IEEE. 2014, pp. 120–123.

[46] Jifeng Xuan et al. "Solving the large scale next release problem with a backbone-based multilevel algorithm". In: *IEEE Transactions on Software Engineering* 38.5 (2012), pp. 1195–1212.

[47] Yusliza Yusoff, Mohd Salihin Ngadiman, and Azlan Mohd Zain. "Overview of NSGA-II for optimizing machining process parameters". In: *Procedia Engineering* 15 (2011), pp. 3978–3983.

[48] Carlos Mario Zapata Jaramillo and Fernando Arango Isaza. “The UNC-method: a problem-based software development method”. In: *Ingenierıa e Investigación* 29.1 (2009), pp. 69–75.

[49] Yuanyuan Zhang, Mark Harman, and S Afshin Mansouri. “The multi-objective next release problem”. In: *Proceedings of the 9th annual conference on Genetic and evolutionary computation*. 2007, pp. 1129–1137.