A Hybrid Approach for Dynamic Multi-Objective Optimization: Combining the Knowledge Guided Bayesian Classification Algorithm with Reinforcement Learning

Master's Thesis

for the acquisition of the academic degree
Master of Science (M.Sc.)

University Trier
FB IV - Computer Science
Chair of Business Informatics I

Supervisor:	Prof. Dr.-Ing. Ingo J. Timm
Supervisor:	M.Sc. Veronika Kurchyna

Submitted on January 15, 2023

Charles David Mupende
[redacted]

Acknowledgements

I am deeply grateful to **Prof. Dr.-Ing. Ingo J. Timm**, who always had an open ear for my questions and through whom I was able to gain access to computer modeling and subsequently the field of optimization. My utmost respect for his commitment enabling me to carry out this project.

Also, I would like to express my appreciation to **Dr.-Ing. Jan Ole Berndt** for his valuable advice during my thesis.

I owe a great deal of gratitude to **M.Sc. Veronika Kurchyna** for her assistance and guidance throughout the planning and execution of this research. Without her support, this project would not have been possible.

Abstract

Solving dynamic multi-objective optimization problems (DMOPs) is becoming increasingly important due to the growing need in industrial applications to make optimal decisions under changing conditions. Problems of this type involve multiple competing objectives that change over time, making it difficult to track the Pareto front effectively. This thesis proposes RL-KGB-DMOEA as a new algorithm to address this challenge. The proposed algorithm uses a Reinforcement Learning-based framework (RL-DMOEA), allowing an agent to choose from a set of prediction methods selectively. Here, the Knowledge-Guided Bayesian Classification Algorithm (KGB-DMOEA), as a nonlinear prediction mechanism that uses clustering of historical Pareto optimal solutions and naive Bayesian classification to predict promising solutions in new environments, is integrated into RL-DMOEA. The goal is to reduce the computational time of KGB-DMOEA with minimal impact on performance. Therefore, an agent is tasked to selectively choose between the computationally more intensive KGB-DMOEA prediction method and less expensive linear prediction methods, namely the Knee-Based Method and the Center-Based Method. In addition, a boundary detection mechanism is used to exclusively select KGB-DMOEA in boundary regions of the decision space, as linear prediction methods tend to be less effective in such situations. In combination with a decaying epsilon-greedy action selection policy, with the intent to promote early exploration, promising empirical results were obtained. It could be demonstrated that RL-KGD-DMOEA can significantly reduce execution time compared to using standalone KGB-DMOEA, while maintaining comparable performance on DF1 benchmark problems.

Keywords: *Dynamic Multi-Objective Optimization, Evolutionary Algorithm, Cluster algorithm, Reinforcement Learning, Prediction Model*

Zusammenfassung

Die Lösung von dynamischen Mehrkriteriellen-Optimierungsproblemen (DMOPs) gewinnt zunehmend an Bedeutung, da es in Industrieanwendungen vermehrt notwendig ist, optimale Entscheidungen unter sich verändernden Bedingungen zu treffen. Probleme dieser Art beinhalten mehrere konkurrierende Ziele, die sich im Laufe der Zeit ändern, was es schwierig macht die Pareto-Front effektiv zu verfolgen. In dieser Arbeit wird ein neuer Algorithmus, RL-KGB-DMOEA, vorgeschlagen, um dieser Herausforderung zu begegnen. Der vorgeschlagene Algorithmus verwendet einen auf Reinforcement Learning basierendes Framework (RL-DMOEA), welches es einem Agenten ermöglicht, aus einer Reihe von Vorhersagemethoden selektiv zu wählen. Dabei wird der Knowledge-Guided Bayesian Classification Algorithm (KGB-DMOEA), als nichtlinearer Vorhersagemechanismus, welcher mittels Clustering von historischen Pareto-Optimal-Lösungen und der Verwendung eines naiven Bayes'schen Klassifikator, vielversprechende Lösungen in neuen Umgebungen vorhersagt, in RL-DMOEA integriert. Um die Rechenzeit des KGB-DMOEA bei minimaler Auswirkung auf die Leistung zu reduzieren, wird ein Agent beauftragt, selektiv zwischen der rechenintensiveren KGB-DMOEA Vorhersagemethode und weniger teuren linearen Vorhersagemethoden, namentlich der Knee-Based Methode und der Center-Based Methode, zu wählen. Zusätzlich wird ein Mechanismus zur Erkennung von Randsituationen eingesetzt, um in Randbereichen des Entscheidungsraums ausschließlich KGB-DMOEA auszuwählen, da lineare Vorhersagemethoden sich in solchen Situationen als weniger effektiv erweisen. In Verbindung mit einer abklingenden Epsilon-Greedy-Aktionsauswahlstrategie, die dazu dient eine frühe Exploration zu fördern, wurden vielversprechende empirische Ergebnisse erzielt. Es konnte gezeigt werden, dass RL-KGD-DMOEA die Ausführungszeit im Vergleich zur alleinigen Verwendung von KGB-DMOEA signifikant reduzieren kann, wobei die Leistung in den getesteten DF1-Benchmark-Problemen vergleichbar bleibt.

Schlüsselwörter: *Dynamische Mehrkriterielle-Optimierung, Evolutionärer Algorithmus, Cluster-Algorithmus, Reinforcement Learning, Vorhersagemodell*

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

DMOEA Dynamic Multi-Objective Evolutionary Algorithm

DMOO Dynamic Multi-Objective Optimization

DMOP Dynamic Multi-Objective Optimization Problem

EA Evolutionary Algorithm

GD Generational Distance

HV Hypervolume

IGD Inverted Generational Distance

KGB-DMOEA Knowledge Guided Bayesian Classification DMOEA

KRE Knowledge Reconstruction-Examination

MDP Markov Decision Process

MHV Mean Hypervolume

MIGD Mean Inverted Generational Distance

MOEA Multi-Objective Evolutionary Algorithm

MOP Multi-Objective Optimization Problem

NBC Naive Bayesian Classifier

NSGA-II Non-Dominated Sorting Genetic Algorithm

POF Pareto Optimal Front

POS Pareto Optimal Set

RL Reinforcement Learning

SMOEA Static Multi-Objective Evolutionary Algorithm

1. Introduction

Darwinism is dynamic. It is about change, not stasis; about process, not pattern; about tales, not tableaux; about becoming, not being.

Henry Gee [Gee01]

Dynamic Multi-Objective Optimization Problems (DMOPs) involve the optimization of multiple time-dependent objectives and constraints. They have attracted increasing attention due to their widespread practical applications, such as priority scheduling [IT18], vehicle routing [LYZ21], or model calibration [Mos+18], highlighting their real-world benefits. However, solving Dynamic Multi-Objective Optimization Problems is generally considered a challenging endeavor due to the dynamic nature of these problems. Therefore, it is of great theoretical and practical significance to propose effective methods to address DMOPs.

Research in the field of Dynamic Multi-Objective Optimization (DMOO) involves proposing algorithms capable of effectively tracking changes in the dynamic problem environment. The goal is to adhere to environmental changes detected during the optimization process and to recover adaptation losses due to environmental changes as quickly as possible. This concept of dynamic evolutionary adaptation is illustratively shown in Fig. 1.1. Traditionally Static Multi-Objective Evolutionary Algorithms (SMOEAs), inspired by biological evolution and natural selection, have been widely used to solve Multi-Objective Optimization Problems (MOPs) in static, as well as in dynamic contexts [ABB17]. However, as SMOEAs are not well-equipped to handle dynamics, they often perform poorly in solving DMOPs [Ye+22].

For better handling of dynamics, Dynamic Multi-Objective Evolutionary Algorithms (DMOEAs) have been proposed, extending existing SMOEA approaches with additional dynamic processing techniques. DMOEAs can roughly be divided into three categories. Diversity-Based enhancement strategies, Pprediction-Based mechanisms, and Memory-Based methods [JZY22], which are all approaches that aim to adapt the optimization process to changes in the problem environment.

Most notably, Prediction-Based DMOEAs have demonstrated competitive performance in handling environmental changes [Ye+22]. These performance increases are achieved by building prediction models based on historical and existing information. Researchers using Prediction-Based DMOEAs argue that, although objectives or constraints might dynamically change in DMOPs, the dynamic nature of problems in adjacent times may still share certain similarities. Thus, if properly exploited,

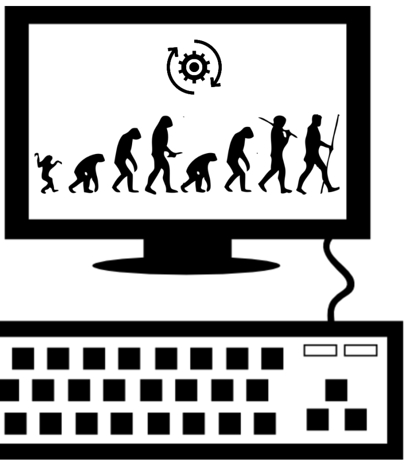


Figure 1.1: Illustration of the dynamic nature of evolutionary algorithms applied to computer simulations, emphasizing the process of adaptation to changing environments. Own illustration.

search experiences gained from previous environments can provide knowledge that can be leveraged for accelerated optimization of adapted problem environments.

With the Knowledge Guided Bayesian Classification DMOEA (KGB-DMOEA), Ye et al. [Ye+22] proposed a novel DMOEA that uses a Naive Bayesian Classifier (NBC) as a prediction model to predict promising solutions when environmental change is detected. This approach has shown to be effective in solving DMOPs, surpassing most previous Prediction-Based DMOEAs in terms of performance. The remarkable results of KGB-DMOEA stem from a fundamental difference setting it apart from other Prediction-Based DMOEAs. Unlike most other Prediction-Based DMOEAs, KGB-DMOEA uses information from *all* previously encountered problem environments. This is done, by examining historical solutions and transferring extracted knowledge to new environments, rather than only using information from the most recent environments, as was previously the most common approach [ABB17].

However, the KGB-DMOEA also has limitations. Most notably, the use of a single prediction model. This is problematic, as the suitability of a prediction model depends on various factors, such as the severity of the environmental change in general, and the linearity or non-linearity of change patterns exhibited by a given optimization problem, in particular [JZY22]. Thus, using a single prediction model may not be suitable for all encountered dynamic problem environments and may lead to suboptimal performance in some cases.

Another potentially limiting aspect of KGB-DMOEAs is the knowledge transfer process. As KGB-DMOEAs always examines all historical information, albeit compressed as clusters, the optimization process may be computationally expensive, especially when the number of historical environments is large. This computational cost may be unnecessary in cases where the environmental change is not severe or behaves linearly, where simpler prediction models may be sufficient.

The problem of not dynamically adapting the prediction model used to the problem environment is not unique to the KGB-DMOEA. This problem is a generally acknowledged issue in the current body of research and addressed in various publications

[SLM16] [RMP17]. However, most approaches that aim at dynamically adjusting the employed prediction model lack learning capabilities or feedback mechanisms [Zou+21], which can be very valuable in helping to ensure correct prediction model selection after detecting environmental changes.

Reinforcement Learning (RL) is a machine learning technique effective in solving a wide range of problems in fields such as robotics [KBP13], automation [Par+22], or calibration [Tia+20]. In recent times, RL also garnered the interest of researchers in the DMOEA community. Zou et al. [Zou+21] attribute this to the sequential decision-making process of RL systems, where an agent's goal is to exacerbate behavior that maximizes a reward function by interacting with its environment. In other words, the agent tries to learn which actions to take to reach a defined goal. This goes hand in hand with dynamically adapting the prediction model used in a DMOEA to a given environment, as an agent's goal can be defined as choosing a prediction model best suited to a given environment state.

RL techniques have been used in evolutionary optimization literature to enhance algorithm performance and solve real-world problems. However, integrating RL methods into DMOEAs is still considered in its infancy [Zou+21]. Therefore, the motivation of this thesis is to integrate KGB-DMOEA into a RL-based DMOEA framework as a decision-making module that can dynamically adapt the prediction model used to the environment at hand.

The assumption here is that a RL agent should be able to learn, from a set of prediction models, the best prediction model to use for a given environment state when change is detected. A similar approach has been most notably proposed by Zou et al. [Zou+21] in which a RL-based DMOEA framework is used to choose from a given set of prediction models based on the severity of environmental changes detected. However, the author's approach does not take the computational time of the prediction models into account, which can be very valuable in helping to ensure overall computational efficiency. In addition to that, the authors do not explicitly inquire about managing the trade-off between *exploration* and *exploitation*, which is a critical factor in RL. As RL uses sequential decision-making, the agent must maintain a balance between *exploration* and *exploitation*. In other words, the RL agent must be guided in such a way that there is a balance between exploring the potential benefits of given actions (prediction model choice) and exploiting actions already known to be beneficial for a given environment state [SB98].

1.1 Research Question and Key Contributions

This thesis aims to propose an improved version of the KGB-DMOEA that maintains its benefits, which mainly are sophisticated solution prediction through evaluation of all historical information available, while mitigating its potential drawbacks, i.e., high computational cost, with RL. Based on this, the following research question is formulated:

"How can Reinforcement Learning be used to decrease the computational cost of the KGB-DMOEA while maintaining its performance?"

For this, a novel algorithm is proposed that combines KGB-DMOEA with RL. The goal is to have a RL agent as a decision-maker that dynamically adapts the prediction model used to the environment at hand. This is done based on the severity of the environmental change detected. Depending on that, a RL agent can either choose to use KGB-DMOEA as a prediction method or more straightforward and computationally less expensive prediction models.

Exerting computational pressure while maintaining performance is one of the main goals of the proposed algorithm. For this, the agent is rewarded based on the time taken and the quality of solutions predicted by chosen prediction methods. Furthermore, the proposed algorithm checks for boundary situations, e.g. situations where the direction of solutions for the following environment is unclear. For these cases, KGB-DMOEA is the default prediction model. With this, the risk of false predictions damaging algorithm performance is mitigated, especially in high-dimensional problem spaces.

As a last, a decaying epsilon-greedy action selection probability is used [NTB21] to balance the exploration and exploitation trade-off. The idea here is that the agent should explore the benefits of prediction models early in the optimization process while exploiting those methods known to provide good results later in the optimization process. This is especially important for high-dimensional problems when using KGB-DMOEA, as the performance of the KGB-DMOEA algorithm needs to have sufficient historical information available.

The components presented have, to the best of my knowledge, not been previously used in combination with KGB-DMOEA and therefore, key contributions of this thesis are summarized as follows:

1. A hybrid DMOEA that combines KGB-DMOEA with RL is proposed to improve KGB-DMOEA performance while mitigating computational cost.
2. The introduction of a reward function, depending on the prediction speed and solution quality of a chosen prediction model, to promote high overall algorithm performance.
3. A boundary detection mechanism to use KGB-DMOEA as a default prediction model in boundary situations, mitigating the risk of reduced performance due to inaccurate predictions.
4. Employing a decaying epsilon-greedy action selection probability to promote broad exploration at early algorithm stages to capture potentially useful historical solutions.

1.2 Thesis Structure

Following this introduction, an overview of the required knowledge for the topics of this thesis is given in **Chapter 2**. The chapter leads gradually towards DMOEAs and concludes with an introduction to the core elements of RL. **Chapter 3** presents the literature relevant to this thesis and outlines the research gap addressed with this work. Subsequently, the proposed algorithm is described in **Chapter 4**. First, the general framework is presented, followed by the individual components of the algorithm. In **Chapter 5**, an experimental study is conducted, in which the performance of the proposed algorithm is compared to an implementation of KGB-DMOEA. **Chapter 6** gives an outlook to potential future work. Finally, the thesis is concluded in **Chapter 7**.

2. Theoretical Background

The following chapter is divided into four major sections. Section 2.1 is dedicated to optimization and the challenges of solving optimization problems. First, the general optimization problem is introduced and then extended by a multi-objective definition in subsection 2.2 and a dynamic multi-objective definition in subsection 2.3. The last section 2.4 is devoted to Reinforcement Learning and briefly introduces its core elements.

2.1 Optimization

Optimization is a key aspect of the decision-making process that involves the systematic identification and selection of the optimal solution to a given problem with regard to a set of constraints [Dan96]. It is a fundamental concept that is central to many fields, including engineering [MN21], economics [HN87] or operations research [SKN11].

At its core, the optimization process involves evaluating various solution options and selecting the one that is considered the most optimal by min- or maximizing an objective function using tools and techniques drawn from fields such as computer science, mathematics, or statistics. This process applies to a wide range of problems, such as strategically planning infrastructure [Hug+05], determining the most effective method for early disease detection [Su+21], or the most efficient schedule for completing tasks [AJA16]. Essentially, any question that seeks to identify the best or most favorable option can be considered an optimization problem.

Definition: The Single-Objective Optimization Problem.

In general, a single-objective optimization problem, with the goal of minimizing a given objective function, can be mathematically formulated as follows [Lia+15]:

$$\begin{aligned} &\min_x f(x) \\ \text{s.t. } & g_i(x) \leq 0, && i = 1, 2, \ldots, m \\ & h_j(x) = 0, && j = 1, 2, \ldots, p \\ & l_k \leq x_k \leq u_k, && k = 1, 2, \ldots, n \end{aligned} \tag{2.1}$$

Where $f(x)$ is the objective function to be minimized, $g_i(x)$ and $h_j(x)$ are the inequality and equality constraints, respectively, with l_k and u_k being lower and upper bounds of the variables x_k.

Thereby the *decision space* of a given optimization problem is defined by all possible value combinations of the decision variables x_k. The *objective space*, on the other hand, is defined by all possible values that the objective function $f(x)$ can take.

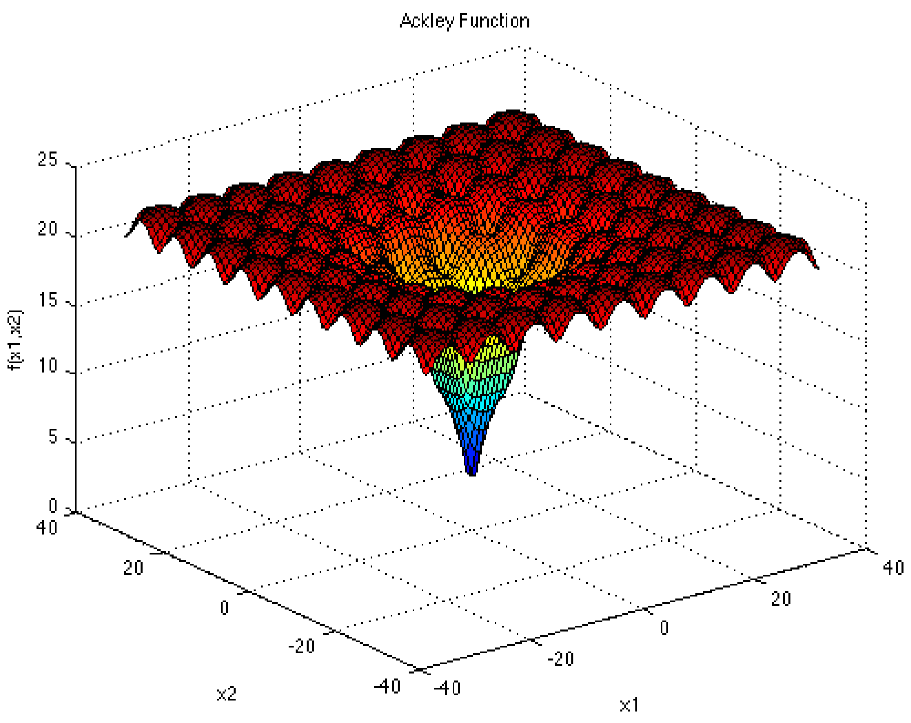


Figure 2.1: 3D plot of the Ackley function,
by Surjanovic and Bingham [SB13].

The solution of an optimization problem, where one would want to minimize a given objective function, is the set of decision variables x_k^* that minimizes the objective function $f(x)$ with regard to given constraints and variable bounds. This solution is known as the *optimal solution*.

The Ackley function, a widely used test function in optimization, can be used as an example of an optimization problem [Ack12]. The function exhibits a flat outer region and a deep hole at the center when plotted in three dimensions, as depicted in Fig. 2.1. The function is defined as follows [Ack12]:

$$f(x) = -20e^{-0.2\sqrt{0.5(x_1^2+x_2^2)}} - e^{0.5(cos(2\pi x_1)+cos(2\pi x_2))} + 20 + e \tag{2.2}$$

Where $x = [x_1, x_2]$ is a two-dimensional vector of decision variables, and e is the base of the natural logarithm. The function has a global minimum at $x = [0, 0]$, where $f(x) = 0$. The optimization of this function can be challenging, especially for classical optimization algorithms such as Hill-Climbing, which are prone to get trapped in one of the many local minima of the function [KWC01].

2.1.1 Implications of Problem Hardness for Algorithm Selection

In the context of optimization, problem hardness refers to the difficulty of finding the *optimal solution* to a given optimization problem [SLC11]. This difficulty can be due to various factors, such as the complexity of the objective function or the size of the *decision space* [WKW16]. Where in some cases, it is possible to directly solve for the optimal solution by taking the derivative of the objective function, this may not be feasible for more complex or unknown objective functions. In such cases, the optimal solution can often only be approximated iteratively by using specialized optimization algorithms.

The time complexity of an algorithm, as a function of the size and features of the input, is a standard measure of problem hardness [Wei+09]. Thereby, *easy* problems are defined as problems that can be solved in polynomial time [Wei+09], meaning that the time required to solve the problem increases at most polynomial with the size of the inputs, thus allowing problems to be solved in a reasonable amount of time, even for large problem scales. To understand the significance of polynomial time complexity, consider the plot in Fig. 2.2, which compares the growth rates of various functions as input size increases.

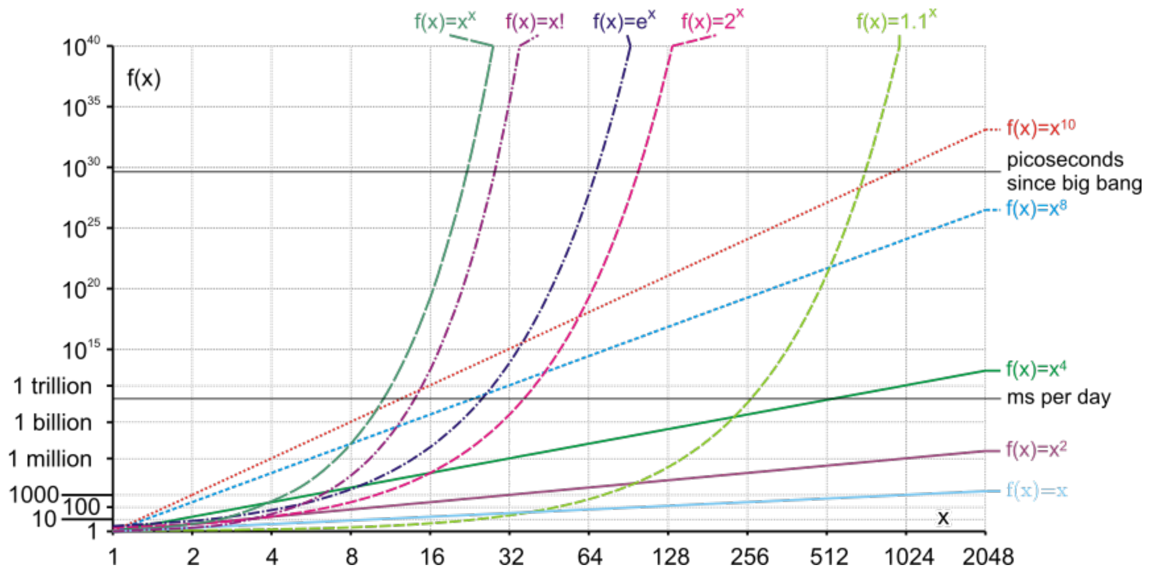


Figure 2.2: A log-scaled plot comparing growth rates of various functions as input size increases, by Weise et al. [Wei+09].

Alternatively, *hard* problems are defined as problems that require exponential time or worse to solve, making them infeasible for large inputs, even for relatively small problem scales [Wei+09]. One example of a hard optimization problem is the single-source-shortest path problem [Tho99], in which the goal is to find the shortest path between a given source vertex and all other vertices in a graph. This problem is typically solved using algorithms such as Dijkstra's algorithm [Dij59], which uses a priority queue to explore the vertices in the graph in an order that guarantees that the shortest path is discovered first. While Dijkstra's Algorithm has a time complexity of $\mathcal{O}(V^2)$ for a graph with V vertices [Cor+01], this can still be infeasible for extensive graphs. In such cases, faster but less accurate algorithms, such as the A* Search Algorithm [HNR68] or the Bellman-Ford Algorithm [BG09], may be used to find a satisfactory, though not necessarily optimal, solution to the shortest path problem.

With this in mind, the implications of problem hardness in optimization are significant, as many real-world optimization problems are in fact hard. Despite this, it is often necessary to solve these hard optimization problems in a reasonable amount of time to make informed decisions and improve processes and systems. As a result, it is important to consider the problem hardness of a given optimization problem and choose the appropriate algorithms to find a satisfactory solution with regard to time or computational budgets.

2.1.2 Metaheuristics as a Solution Approach for Hard Problems

When faced with an optimization problem, it is essential to consider the problem's hardness and select the appropriate algorithm to find a satisfactory solution with regard to time complexity. Therefore, most of the time, rather than striving for the optimal solution, the goal is to find a reasonably good solution in an acceptable amount of time [Wei+09]. This often leads to a trade-off between solution quality and algorithm runtime, as shown in Fig. 2.3.

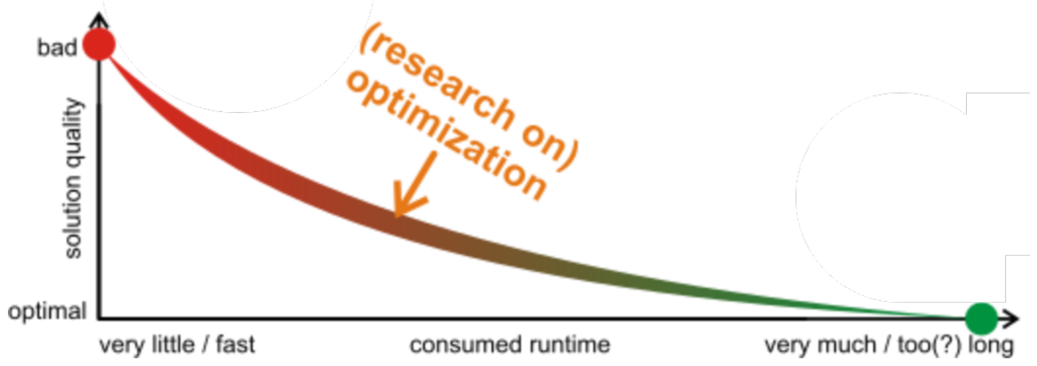


Figure 2.3: Illustration of the optimization time vs. solution quality trade-off, original by Weise et al. [Wei+09], adapted.

Metaheuristics are a class of algorithms that have gained widespread popularity due to their ability to effectively find good, though not necessarily optimal, solutions to hard optimization problems within a reasonable time frame [Wei+09]. These algorithms work by iteratively exploring the decision space and using heuristics to guide the search process. Thereby, this class of algorithms can effectively search the decision space and find near-optimal solutions even for problems with large decision spaces or non-derivable objective functions [Wei+09]. Examples of metaheuristics include simulated annealing [vLA87], which uses a random search process inspired by the annealing of metals; genetic algorithms [KCK21], which use principles of natural evolution to generate and evolve solutions; and particle swarm optimization [BM17], which uses the movement of a group of particles to search for solutions. Even though these algorithms do not guarantee finding the optimal solution, they are able to find solutions that are close to optimal in a reasonable amount of time and are hence widely used for solving a variety of optimization problems.

2.2 Multi-Objective Optimization

Multi-Objective optimization problems involve finding a solution that simultaneously optimizes multiple conflicting objectives [Mie99]. An example of a multi-objective optimization problem is the design of an airplane, where the goals may include minimizing airplane weight, to reduce fuel consumption and increase range, while also maximizing structural strength for passenger safety. These two objectives conflict with each other, as increasing structural strength typically increases weight. Problems of this type are common in real-world applications, as they often involve complex systems with multiple competing goals and constraints.

Definition: The Multi-Objective Optimization Problem.

The Multi-Objective Optimization Problem can be formally defined as follows [Ben09]:

For a given set of decision variables $x \in \mathbb{R}^n$, with lower and upper bounds $l_i \leq x_i \leq u_i$, and a set of conflicting objective functions $X = \{f_1(x), f_2(x), ..., f_m(x)\}$, find a solution x_i^* that simultaneously optimizes *all* objective functions subject to given problem constraints.

This problem can be written as:

$$
\begin{aligned}
\min_{x \in X} & f(X) = [f_1(x), f_2(x), ..., f_m(x)] \\
\text{s.t.} \ & g_1(x) \leq 0, g_2(x) \leq 0, ..., g_p(x) \leq 0 \\
& h_1(x) = 0, h_2(x) = 0, ..., h_q(x) = 0 \\
& l_i \leq x_i \leq u_i,
\end{aligned}
\tag{2.3}
$$

As Multi-Objective Optimization Problems (MOPs) typically involve multiple conflicting objectives that need to be optimized simultaneously, it is not possible to find a single solution from a given decision space that optimizes all the objectives in the objective space simultaneously [Wei+09]. Instead, there are usually multiple possible solutions that represent trade-offs between the different objectives, requiring a decision-maker to determine which solution should be implemented. The decision-maker, who is expected to be an expert in the problem domain, must consider the trade-offs between conflicting objectives and choose the solution that best aligns with their preferences and problem-specific constraints.

Over the years, various approaches have been developed to solve MOPs, which can be classified into four categories, including No-Preference Methods, A Priori Methods, A Posteriori Methods, and Interactive Methods, which mostly differ in the way they involve preference information from the decision-maker [Hwa79].

- **No-Preference Methods**: No-Preference Methods do not involve the decision-maker and identify a neutral compromise solution without preference information [Hwa79].
- **A Priori Methods**: A Priori Methods require the decision-maker to clearly express their preferences as a weighting before starting the optimization process, [Hwa79]. However, this can be challenging as the system's complexity and limited understanding may render it difficult for a decision-maker to formulate preferences beforehand [Hwa79].

- **A Posteriori Methods**: A Posteriori Methods generate a set of Pareto-Optimal solutions from which the decision-maker must choose a preferable solution. This approach allows for the understanding of the complete Pareto-Optimal Front. However, this approach can be resource-intensive as preferences are unclear beforehand, and more solutions need to be analyzed [Hwa79].
- **Interactive Method**: Interactive Methods involve the decision-maker in a dynamic way, alternating between preference allocation and optimization, progressively and iteratively refining the solution determination towards the domain of interest [Hwa79].

A Posteriori Methods, which do not require previous preference information from a decision-maker, can provide insight into the entire solution space of the problem and are therefore often preferred for solving multi-objective optimization problems [Mie98]. These methods aim to compute all or a representative set of Pareto-Optimal solutions, which is a set of solutions that represents the best possible trade-offs between different objectives.

Definition: The Pareto-Optimal Set (POS).

A solution $x^* \in \Omega$ is called Pareto-Optimal solution if there is no other solution $y \in \Omega$ such that $F(y)$ is strictly preferred to $F(x^*)$. The set of all such Pareto-Optimal solutions is defined as [Ben09]:

$$\text{POS} = \{x \in \Omega \mid \nexists y \in \Omega, F(y) \prec F(x)\} \tag{2.4}$$

In other words, the Pareto Optimal Set (POS) is the set of all solutions that cannot be improved upon in terms of the performance measures represented by the objective functions of $F(x)$, without making some performance measures worse.

Definition: The Pareto-Optimal Front (POF).

The Pareto Optimal Front (POF) is the representation of the POS in the objective space, defined as follows [Ben09]:

$$\text{POF} = \{F(x) \mid x \in \text{POS}\} \tag{2.5}$$

The POF is used to visualize the trade-off between different objectives, typically taking the form of a curve when there are two objectives, a surface when there are three objectives, and a hyper-surface when there are more than three objectives.

In Fig. 2.4, the relation between the POS and the POF is depicted. The green-marked points in the decision space represent solutions that strictly dominate the remaining solutions. By connecting the function values of these solutions in the objective space, one can approximate the POF, which represents the best trade-offs between the given objectives.

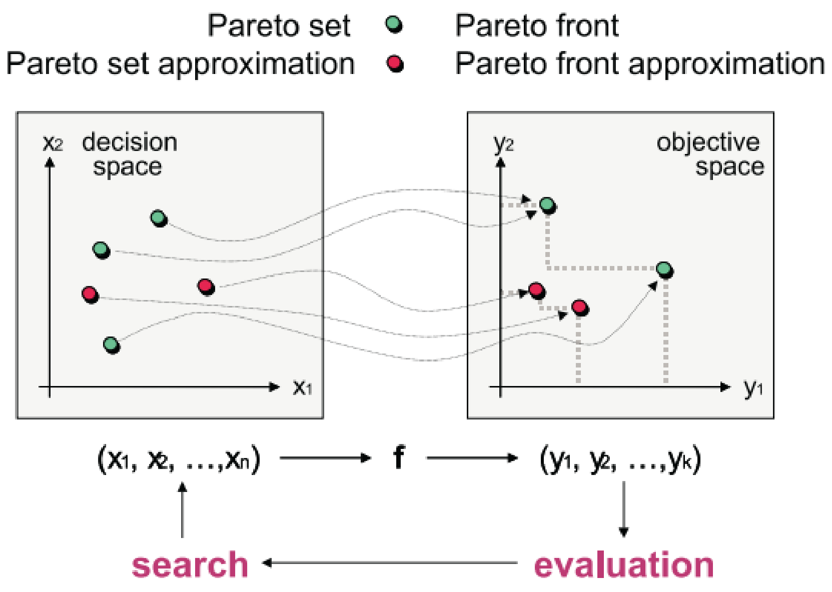


Figure 2.4: The relation between decision space and objective space for MOPs, by Montalvo et al. [Mon+10].

In this regard, it should be noted that solving MOPs a posteriori is a challenging task, as it usually involves iteratively uncovering the POF by searching the decision space for those variable combinations that yield an improvement concerning the objective function values. This can be particularly costly when the decision space is large or when the objective functions being evaluated are computationally intensive, as is often the case when large simulation models are used as objective functions, to give an example. Furthermore, it is important to note that, except for well-defined test problems, the true POF for MOPs of any difficulty is not explicitly known. Therefore, in multi-objective optimization, one usually refers to approximating the POF. Consequently, the main goal of research in multi-objective optimization is to develop efficient algorithms that allow the POF to be determined with minimal computational resources [Wei+09].

2.2.1 Multi-Objective Evolutionary Algorithms

In recent years, Evolutionary Algorithms (EAs) have gained popularity as a population based optimization metaheuristics due to their effectiveness in solving various optimization problems across different fields [ČLM13]. One area where EAs have seen significant growth is in the field of multi-objective optimization, where the use of EAs was first suggested in the 1960s [ES15a], but it wasn't until the mid-1980s that the first actual implementation of a Multi-Objective Evolutionary Algorithm (MOEA) was developed [DFS97].

MOEAs are particularly well-suited for solving problems that are impossible to solve using traditional optimization methods, as they do not require the optimization problem to be differentiable, convex, or even have a closed-form solution [MH21]. Instead, MOEAs rely on a search process that involves generating and evaluating numerous candidate solutions, known as a population. These algorithms work by simulating the process of natural selection, in which the fittest individuals of a population are selected to reproduce and pass on their traits to the next generation. Thus, MOEAs are effective in solving MOPs since they deliver an approximation of the POS in a single algorithm run, rendering them as a good choice for a posterior MOP solving.

This class of metaheuristic algorithms has been shown to be effective at finding reasonably good solutions to a wide range of optimization problems. Thus, MOEAs have been used to solve optimization problems in various fields, including engineering [ACM12], finance [CC07], and biology [MBM11]. However, they can be computationally expensive as they usually require many function evaluations to uncover the POF. Therefore, it is essential to carefully design MOEAs to balance the trade-off between solution quality and computational cost.

2.2.2 General Multi-Objective Evolutionary Algorithm Framework

In both research and real-world application, MOEAs share more or less the same framework [KCK21]. Thereby, the objective of a MOEA is to iteratively converge to the true POF of a problem, which typically consists of a diverse set of points from the problem's decision space.

At each iteration, the fitness (objective function values) of each individual in the current population is evaluated, and a set of Pareto-Optimal solutions is determined and referred to as $P_{current}(t)$, where t represents the generation number. Most implementations of multi-objective evolutionary algorithms also use a secondary population to store all or some of the Pareto-Optimal solutions found throughout the generations [Pan+11]. This secondary population is called $P_{known}(t)$, to reflect potential changes in its membership during the execution of the algorithm. $P_{known}(0)$ is defined as $\emptyset$ (the empty set), and after a defined termination condition is met, which usually is a maximum amount of generations to evaluate, P_{known} as the final overall set of POS is returned by the algorithm.

To converge to the POF, variations of the three main operations, selection, crossover, and mutation, are usually performed in each iteration of MOEAs [Has+19]:

- **Selection**: Selection refers to the process of choosing which individuals in the current population will be selected to produce an offspring for the next generation [Has+19]. Various selection methods are known in the literature with some examples including tournament selection, where a fixed number of individuals are chosen at random from the current population, and the best (based on their fitness) are chosen to produce offspring [MG95]. Whereas, roulette wheel selection involves assigning a probability to each individual in the population based on their fitness and selecting individuals to produce offspring with a probability proportional to their given probability [LL12]. The choice of selection method can have a significant impact on the efficiency and effectiveness of the optimization process.
- **Crossover**: Crossover, also known as recombination, refers to the process of combining the genetic material (decision space variables) of two individuals to produce offspring with traits from both parents. For instance, one common crossover method is single-point crossover, which involves randomly selecting a single crossover point and swapping the genetic material between the two parents at that point to create the offspring [Gwi].
- **Mutation**: Mutation refers to the process of randomly altering the genetic material of an individual, bringing diversity into the population, which can be beneficial for the optimization process as it allows the algorithm to explore new areas of the decision space and potentially find better solutions. One common mutation method is bit-flip mutation, which involves randomly selecting a single bit in the binary representation of the decision variables and flipping it to create the offspring [ES15b].

The general framework of a MOEA introduced above is illustrated in Fig. 2.5. The algorithm begins with the creation of an initial, usually randomly generated, population of individuals, each with a unique set of decision space variables. The fitness of each individual is then evaluated according to the objectives of the optimization problem. Selection is then performed to choose which individuals in the current population will produce offspring for the next generation. Crossover combines the genetic material of two individuals to create offspring with traits from both parents. Mutation randomly alters the genetic material of an individual to introduce diversity into the population. The process is repeated with the new generation of individuals until a satisfactory solution is found or a predetermined stopping criterion is met.

With the general MOEA framework in mind, the following two main issues generally have to be considered when designing MOEAs [ABS17]:

- How can individuals be selected such that non-dominated solutions are preferred over those which are dominated?
- How can diversity be maintained to retain as many elements of the POS as possible in the population?

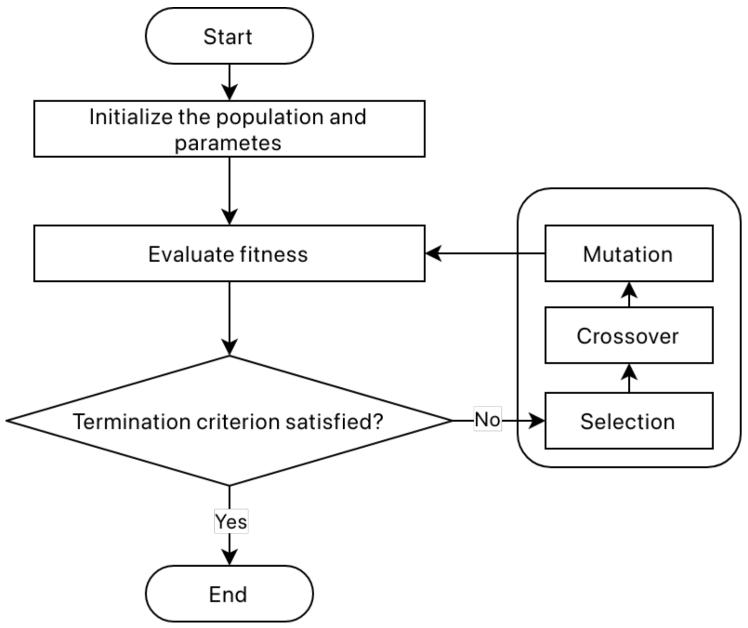

Start
Initialize the population and parametes
Evaluate fitness
Termination criterion satisfied?
No
Yes
Selection
Crossover
Mutation
End

Figure 2.5: Flowchart of the working principle of (Multi-Objective) Evolutionary Algorithms. Own illustration.

To address the first question, specific selection schemes that prioritize non-dominated solutions over dominated ones are typically used in MOEAs. For instance, one evident approach would be selecting non-dominated solutions with a higher probability after each algorithm iteration, making them more likely to be selected as parents for reproduction. Another selection scheme would be ”crowding”, in which solutions that are more densely populated in the decision space are given a higher probability of being selected as parents [Deb+02].

Regarding the second question, various diversity maintenance techniques can be used, such as introducing random perturbations to the population [Deb+02] or using ”elitism” to preserve a fixed number of the best solutions in the population [Deb+02]. These techniques help to ensure that the population remains diverse and prevents the algorithm from converging too quickly on a suboptimal solution.

Balancing these two objectives however, can pose a demanding challenge, as they generally conflict with each other. For example, to maintain diversity in the population, it may be necessary to include individuals that are not as fit, in terms of the objectives being optimized. However, this also results in a population that includes more dominated solutions, hindering the convergence towards the true POF. On the other hand, focusing more on selecting non-dominated solutions may result in a population that is less diverse but has a higher overall quality. Finding the right balance between these two objectives can therefore be a challenging task, as it may require making trade-offs in order to achieve a satisfactory result. These issues at hand are illustrated in Fig. 2.6.

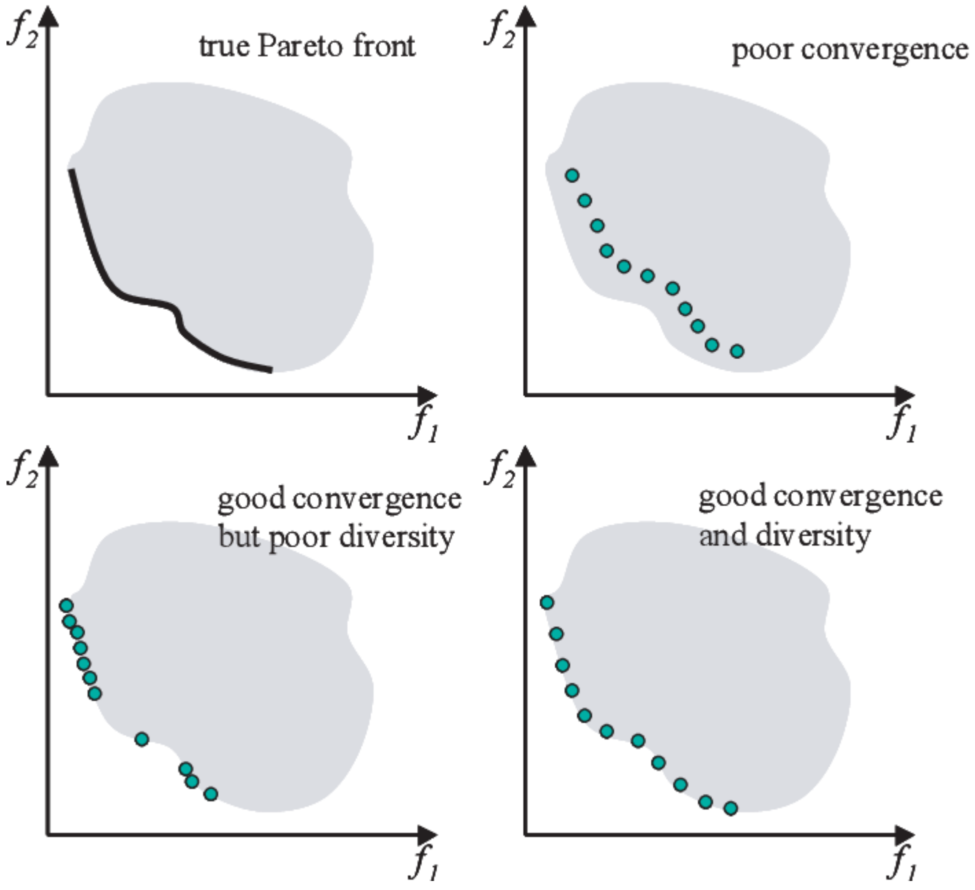


Figure 2.6: Visualization of the dynamics between convergence and diversity, by Das and Panigrahi [DP09].

2.3 Dynamic Multi-Objective Optimization

Real-world problems not only involve the challenge of optimizing multiple objectives at the same time, but they also often involve dynamic changes. Using the aircraft example presented in Section 2.2, we can now modify the example to consider the aircraft's wear and tear additionally. Thus, the goal is not only to minimize the aircraft's weight to reduce fuel consumption while maximize structural strength for passenger safety, but also to determine the POF that varies with the wear of the aircraft. Thus, when the additional dimension of time is considered, the problem is extended to locate the *moving* POF in the objective space.

Definition: The Dynamic Multi-Objective Optimization Problem (DMOP).

A DMOP is defined as the problem of finding a vector of decision variables $\mathbf{x}(t)$ that satisfies a set of constraints and optimizes a vector of objective functions that change over time. Without loss of generality, a typical DMOP can be formalized as follows [ABB17]:

$$\begin{aligned} \min_{\mathbf{x}(t)} \quad & F(\mathbf{x},t) = \{f_1(\mathbf{x},t), f_2(\mathbf{x},t), ..., f_M(\mathbf{x},t)\} \\ \text{s.t.} \quad & g_1(x,t) \leq 0, g_2(x,t) \leq 0, ..., g_p(x,t) \leq 0 \\ & h_1(x,t) = 0, h_2(x,t) = 0, ..., h_q(x,t) = 0 \\ & l_i \leq x_i \leq u_i, \end{aligned} \tag{2.6}$$

Where t represents the time or the dynamic nature of the problem, M is the number of objectives to be minimized, and the functions g and h represent the set of inequality and equality constraints, respectively. It is to be noted that even though dynamic multi-objective optimization also encompasses dynamically changing inequality and equality constraints, the focus of this thesis will be on unconstrained problems only.

Definition: The Dynamic Pareto Optimal Solution.

For a given decision vector $\mathbf{x}^*(i,t)$, it is said to be a Pareto-Optimal solution if there is no other feasible decision vector $\mathbf{x}(j,t)$ such that [ABB17]:

$$f(j,t) \prec f(i,t)^* \setminus f(j,t) \in F_M \tag{2.7}$$

Where $\prec$ represents the Pareto Dominance relation.

Definition: The Dynamic Pareto-Optimal Set (DPOS).

Therefore, the set of all Pareto-Optimal solutions in the decision space at time t is referred to as the Dynamic Pareto-Optimal set, denoted as $POS(t)^*$ such that [ABB17]:

$$POS(t)^* = \{x_i^* \mid \nexists f(j,t) \prec f(x_i^*,t)^*, f(x_j,t) \in F_M\} \tag{2.8}$$

Definition: The Dynamic Pareto-Optimal Front (DPOF).

The Dynamic Pareto-Optimal Front at time t, denoted as $POF(t)^*$, is the set of POS with regard to the objective space at time t such that [ABB17]:

$$POF(t)^* = \{f(i,t)^* \mid \nexists f(j,t) \prec f(i,t)^*, f(j,t) \in F_M\} \tag{2.9}$$

Definition: Change Severity.

To define the dynamic nature of a problem, it is also necessary to determine what change severity is. Change severity measures how significant the environmental changes are in terms of their magnitude. It compares the landscape before and after a change to determine the relative strength of the landscape change [Ric13].

Definition: Change Frequency.

Furthermore, the change frequency is also an essential factor to consider when defining the dynamic nature of a problem. The change frequency refers to how frequently the environment changes. It is commonly measured as the number of generations or the number of fitness function evaluations between two consecutive landscape changes [Ric13].

Classifications of DMOPs.

There are several ways to classify DMOPs based on various characteristics of the changes that occur in the problem. These classifications are often based on the frequency of changes, the severity of changes, the predictability of changes, and the relationship between the POF and the POS [ABB17].

- **Frequency-Based classification**: In Frequency-Based classification, DMOPs are classified based on the frequency with which changes occur in the problem [ABB17]. When the frequency of changes is high, it becomes more difficult to adapt to them as there is limited time to adapt. Conversely, suppose the frequency of change is low. In that case, algorithms have more time to adapt and converge to the POF of a given environment, resulting in a potentially easier problem to solve.
- **Severity-Based classification**: Severity-Based classification is a way to categorize DMOPs based on the magnitude of changes occurring in the problem [ABB17]. When the severity of changes is low, it is generally easier for algorithms to adapt and converge to the POF. This is because the information from the previous environment can be used to improve the convergence speed of the algorithm. On the other hand, if there are many severe changes, each problem instance may be unrelated to the next. In this case, it may not be possible to use information from the previous environment, sometimes even requiring a complete restart of the algorithm [ABB17].

- **Predictability-Based classification**: Predictability-Based classification represents another method of categorizing DMOPs based on the regularity of changes that occur in the problem [ABB17]. When changes are predictable, they follow a consistent pattern or trend. These changes can be further divided into two categories: cyclic changes, which are periodic and repeat over time, and acyclic changes, which are not periodic and do not repeat [ABB17]. Predictability can be an important factor to consider when trying to solve a DMOP, as it can impact the ability of the algorithm to adapt to the changing environment. For example, if changes are predictable, the algorithm may be able to anticipate and prepare for said changes, leading to faster convergence towards the POF. On the other hand, if changes are random and not predictable, the algorithm may struggle to adapt, leading to slower convergence or failure to find the POF.

- **POF-POS Relation-Based classification**: Farina, Deb and Amato [FDA04a] proposed a classification system for DMOPs based on the relationship between the POF and the POF in the presence of changes. They identified four different types of DMOPs based on this relationship:
  - **Type I**: The POS changes while the POF is unaffected.
  - **Type II**: Both the POF and the POS are affected by the changes.
  - **Type III**: The changes affect the POF, but the POS is unaffected.
  - **Type IV**: The changes affect neither the POF nor the POS.

Farina, Deb and Amato [FDA04a] also noted that, even in Type IV DMOPs, where the POS and the POF are fixed and do not change over time, other regions of the fitness landscape may still be changing. This can occur for example, when the feasible space changes over time, as seen in Fig. 2.7.

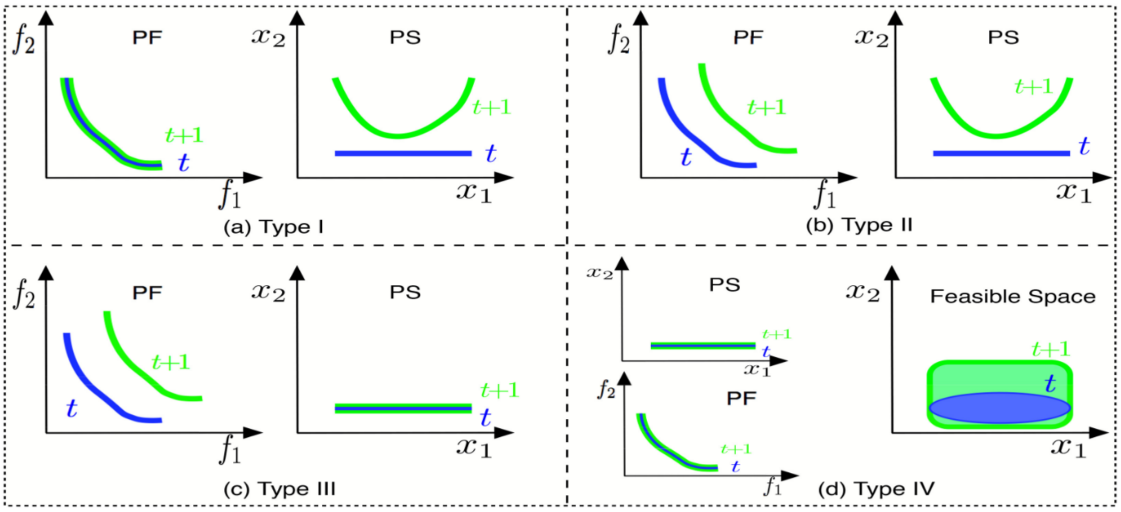


Figure 2.7: Visualization of different types of DMOPs, by Jiang, Zou and Yao [JZY22].

2.3.1 Dynamic Multi-Objective Evolutionary Algorithms

DMOPs have garnered significant attention in recent years due to their widespread practical applications. As EAs were first applied to static problems and later to dynamic but single-objective optimization problems, only more recent efforts have been made to extend these approaches to the dynamic multi-objective case [JZY22]. SMOEAs have been traditionally used for solving DMOPs [JZY22], but they often lack in performance due to their inability to account for the dynamic nature of said problems. To address this issue, researchers have proposed DMOEAs by extending existing SMOEAs with additional techniques for processing dynamic environments. While other optimization techniques, such as Particle Swarm Optimizers (PSO) [Zhe+22], have also been adapted for dynamic environments, EAs remain the most widely used method in this field [JZY22].

It is to be noted that MOEA and SMOEA refer to the same types of algorithms. For clarity, SMOEA will be used to refer to MOEAs for the remainder of this thesis.

2.3.2 General Dynamic Multi-Objective Evolutionary Algorithm Framework

The general framework of a DMOEA is shown in Alg. 1 [Che+22]. The framework consists of three main components: the actual optimization process, which usually employs a SMOEA, the detection of environmental changes, and the response strategy to said environmental changes. In line 1 an initial, usually random, population is generated of size **N**. With line 2, the optimization process is executed using any SMOEA. While the stopping criterion is not met, the algorithm checks for environmental changes as seen in lines 3-4. If the environment has changed, the algorithm executes a response strategy in line 5. With this response strategy the goal is to relocate the population, so that quick convergence to the POF of the new environment is achieved. If the environment has not changed, this response strategy is skipped with line 6, and a SMOEA is employed again with line 7.

Algorithm 1 General Dynamic Multi-Objective Evolutionary Algorithm (DMOEA)

```
Input: Population size $N$
Output: Population $P$
1: initpop($N$);
2: optimize the MOP by using a SMOEA;
3: while stopping criterion not met do
4:     if environment changes then
5:         response();
6:     else
7:         optimize the MOP by using a SMOEA;
8:     end if
9: end while
```

The response to environmental changes is executed only if the environment has changed and can generally be executed in two ways: (1) by restarting the optimization process from scratch or (2) by using knowledge about the previous search to accelerate optimization after a change [ABB17]. However, the second approach requires algorithms to maintain adaptability in order to respond to changes in the environment properly. This is because convergence during the optimization process may lead to a lack of diversity, which can impact the algorithm's ability to adapt to changing environments [ABB17]. Balancing the conflicting goals of convergence and diversity is crucial in order to effectively track the POF in dynamic environments.

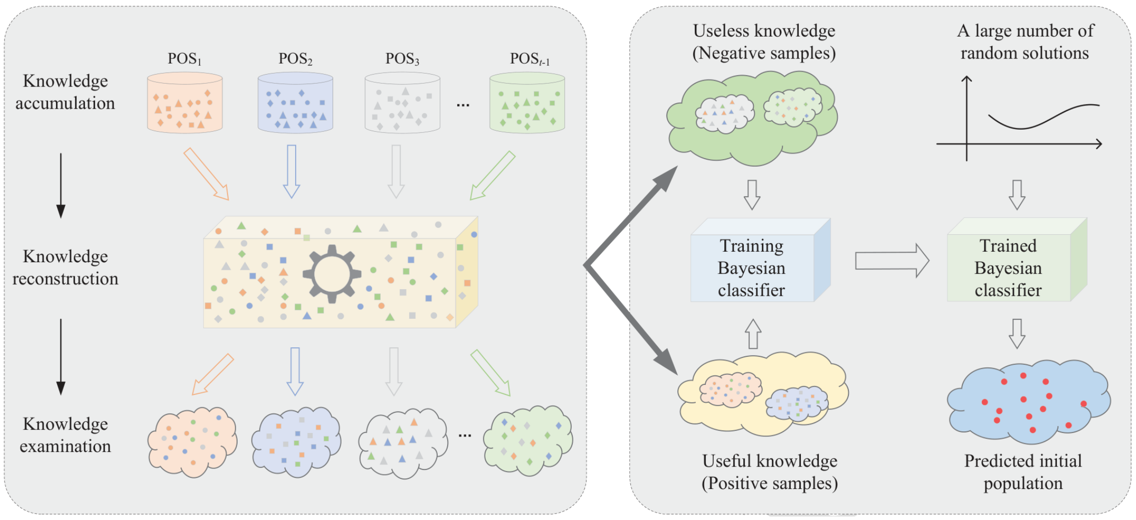
The response strategies used to adapt to changing environments can broadly be classified into three main categories [ABB17]: **Diversity-Based strategies** aim to maintain a diverse population to enable adaptation; **Memory-Based strategies** utilize information from previous searches to guide optimization in new environments; **Prediction-Based strategies** use prediction models to predict patterns of environmental change and guide the optimization process accordingly. Each approach has its strengths and limitations, and the most suitable strategy depends on the characteristics of the dynamic optimization problem at hand [ABB17].

However, as Prediction-Based DMOEAs learn a trend prediction model which is used to guide the evolutionary search in subsequent environments, they exhibit advantages in convergence and diversity, especially when environmental changes exhibit predictable patterns [Ye+22] [JZY22].

2.3.3 Knowledge Guided Bayesian Classification Algorithm

The Knowledge Guided Bayesian Classification DMOEA (KGB-DMOEA) is a novel Prediction-Based DMOEA proposed by Ye et al. [Ye+22]. KGB-DMOEA has demonstrated exceptional performance in various commonly used DMOO benchmark problems. By using transfer learning [WKW16], which constitutes human learning inspired methods to transfer knowledge from one problem domain to another, Ye et al. [Ye+22] combine EAs with knowledge transfer and prediction-methods in order to effectively solve DMOPs. Unlike most previously existing Prediction-Based DMOEAs, KGB-DMOEA does not only reuse knowledge from adjacent environments to learn its prediction model, but rather it retains knowledge over all earlier search experiences [Ye+22]. This positive knowledge transfer of all historical solutions significantly improves the performance of KGB-DMOEA compared to previously used methods.

It does so by accumulating knowledge from each optimization episode and on the detection of change in the environment, previously accumulated knowledge is reconstructed, examined and then used to guide the search for the POS of the new environment. The search experience of each historical environment is thereby treated as a piece of knowledge, and all historical optimal solutions are reconstructed and examined using a Knowledge Reconstruction-Examination (KRE) strategy to determine their usefulness. The useful solutions are then used as positive samples to train a probabilistic classifier, specifically a Naive Bayesian Classifier (NBC), which can filter out high-quality solutions from numerous randomly generated ones, thereby guiding the search for optimal solutions in the new environment. This process is illustrated in Fig. 2.8.



Knowledge accumulation
Knowledge reconstruction
Knowledge examination
POS1
POS2
POS3
POSt-1
Useless knowledge (Negative samples)
A large number of random solutions
Training Bayesian classifier
Trained Bayesian classifier
Useful knowledge (Positive samples)
Predicted initial population

Figure 2.8: Process schematic of the Knowledge Guided Bayesian Classification Algorithm, by Ye et al. [Ye+22].

Algorithm 2 Knowledge Guided Bayesian DMOEA (KGB-DMOEA)

Input: The target DMOP $\mathbf{F_t}(\mathbf{x})$.
Output: The set of POS in different environments: **PS**.

```
 1: Initialize $\mathbf{PS} = \varnothing, t = 0$;
 2: Randomly initialize population: $\mathbf{initPop}(\mathbf{N})$;
 3: while stopping condition is not met do
 4:     $\mathbf{POS_t} = SMOEA(\mathbf{F_t}(\mathbf{x}), \mathbf{initPop})$;
 5:     $\mathbf{PS} = \mathbf{PS} \cup \mathbf{POS_t}$;
 6:     if environment changes then
 7:         $t = t + 1$;
 8:         $(\mathbf{POP_{useful}}, \mathbf{POP_{useless}}) = KRE(\mathbf{PS})$;
 9:         $\mathbf{C_{NBC}} = NBC(\mathbf{POP_{useful}}, \mathbf{POP_{useless}})$;
10:         $\mathbf{X_{test}} \leftarrow$ Generate numerous random solutions;
11:         $\mathbf{Y_{test}} \leftarrow$ Predict the class of $\mathbf{X_{test}}$ with $\mathbf{C_{NBC}}$;
12:         $\mathbf{initPop} \leftarrow$ The solutions $x \in \mathbf{X_{test}}$ that are marked as "+1" in $\mathbf{Y_{test}}$;
13:     end if
14: end while
```

The Overall KGB-DMOEA Framework

To further illustrate the process of KGB-DMOEA, the pseudocode is given in Alg. 2 [Ye+22]. The algorithm begins in line 1 by initializing an empty set **PS**, which is used to store the approx. POS of each environment at time t, initialized as 0. In line 2, an initial population **initPop** with size N is randomly generated and is used as the starting point for a SMOEA to search for $\mathbf{POS_t}$ of the current environment in line 4. In line 5, the $\mathbf{POS_t}$ is then added to **PS** and if an environmental change occurs, t is incremented by one as seen in lines 6 and 7. **PS** is then fed into the KRE strategy, which divides the solutions into useful ($\mathbf{POP_{useful}}$) and useless ($\mathbf{POP_{useless}}$) sets with line 8. These sets are subsequently used to train a Naive Bayesian Classifier ($\mathbf{C_{NBC}}$), which is then used to predict the labels ($\mathbf{Y_{test}}$) of randomly generated test samples ($\mathbf{X_{test}}$). The solutions in $\mathbf{X_{test}}$ that are labeled as useful in $\mathbf{Y_{test}}$ are then selected to construct a high-quality initial population for the new environment.

Algorithm 3 Knowledge Reconstruction-Examination (KRE)

```
Input: The historical POS set $\mathbf{PS}$
Output: The solution sets with useful knowledge $\mathbf{POP_{useful}}$ and with useless knowledge $\mathbf{POP_{useless}}$.
 1: Initialize the number of clusters $N_c, size = |\mathbf{PS}|$;
 2: Set $p_i \in \mathbf{PS}$ as a cluster $C_i$ and centroid $c_i$;
 3: while $size > N_c$ do;
 4:     Find two clusters with the smallest Euclidean distance among all clusters with Eq. (2.10), noted by $C_i$ and $C_j$;
 5:     $C_i = C_i \cup C_j$;
 6:     Remove $C_j$;
 7:     Update the centroid $c_i$ with Eq. (2.11);
 8:     $size--$;
 9: end while
10: $\mathbf{c} = c_1 \cup c_2 \cup \cdots \cup c_{N_c}$;
11: $(\mathbf{F_1}, \mathbf{F_2}, \ldots, \mathbf{F_h}) = Nondominated\_Sorting(\mathbf{c})$;
12: The clusters $C_i$ with corresponding centroids $\in \mathbf{F_1}$ are merged into $\mathbf{POP_{useful}}$;
13: The clusters $C_j$ with corresponding centroids $\notin \mathbf{F_1}$ are merged into $\mathbf{POP_{useless}}$;
```

Knowledge Reconstruction-Examination (KRE)

Using KRE, when change in the optimization problem is detected, it is determined which historically optimal solutions might be useful for the current optimization episode. This process is explained in more depth with Alg. 3 [Ye+22]. The algorithm begins in line 1 with initializing the number of clusters $\mathbf{N_c}$ and **size** as the length of the historical POS archive **PS**. In line 2, each solution in **PS** is considered as a cluster $\mathbf{C_i}$ with its corresponding centroid $\mathbf{c_i}$ being initialized as itself. Starting from line 4, the algorithm iteratively merges the two clusters with the smallest Euclidean distance, until the number of clusters is equal to $\mathbf{N_c}$ as seen in lines 5-9.

The Euclidean distance between clusters $\mathbf{C_k}$ and $\mathbf{C_z}$ is calculated using the following equation [Ye+22]:

$$D_{C_k,C_z} = \sqrt{\sum_{l=1}^{d}(c_k^l - c_z^l)^2} \tag{2.10}$$

Where $\mathbf{c_k}$ and $\mathbf{c_z}$ are the centroids of clusters $\mathbf{C_k}$ and $\mathbf{C_z}$, respectively, and d are the dimensions of the decision variable. The two most similar clusters, $\mathbf{C_i}$ and $\mathbf{C_j}$, are then combined into $\mathbf{C_i}$, and $\mathbf{C_j}$ is removed. The value of the new centroid $\mathbf{c_i}$ is subsequently updated as follows [Ye+22]:

$$c_i = \frac{1}{|C_i|}\sum_{l=1}^{|C_i|} x_l \tag{2.11}$$

Where $|\mathbf{C_i}|$ refers to the number of solutions in $\mathbf{C_i}$ and x_l is the lth solution in cluster $\mathbf{C_i}$. Then the **size** is decreased by one in line 8. Subsequently, the centroids of the condensed clusters are evaluated in the new problem environment to perform non-dominated sorting [Bao+17], which is used to divide the clusters into multiple sets: $(\mathbf{F_1}, \mathbf{F_2}, \ldots, \mathbf{F_h})$ with line 11. Clusters in $\mathbf{F_1}$ strictly dominate all other clusters. This means that solutions contained in the clusters $\mathbf{C_i}$ with centroids $\in \mathbf{F_1}$ are more likely to yield better objective function values in the current environment and are thus merged into $\mathbf{POP_{useful}}$, with all other clusters in $(\mathbf{F_2}, \ldots, \mathbf{F_h})$ being merged into $\mathbf{POP_{useless}}$ as seen in lines 12 and 13.

Naive Bayesian Classifier (NBC)

The NBC is a probabilistic classifier, based on the application of Bayes' theorem with the assumption of naive feature independence [Leu07]. It has been shown to be effective in several practical applications, such as data classification [SY19] and medical diagnosis [BK12], and performs well on small-scale data [Ye+22]. Another advantage of NBC as a prediction model is short training times, which renders them good prediction models for usage within DMOEAs [Ye+22].

The theory behind naive Bayesian classification can be summarized as follows [Ye+22]:

Given a set of classes $S = \{y_1, y_2, ..., y_f\}$ and a sample $x = \{a_1, a_2, ..., a_d\}$, which is a d-dimensional vector representing the values of d attributes $a_1, a_2, ..., a_d$, the goal of the NBC is to predict the class y_i to which x belongs with the highest a posteriori probability, i.e. x is predicted to be class y_i if and only if there exists i such that [Ye+22]:

$$P(y_i|x) = \max\{P(y_1|x), P(y_2|x), ..., P(y_f|x)\} \tag{2.12}$$

This can be done using Bayes' theorem [Ye+22]:

$$P(y_i|x) = \frac{P(x|y_i)P(y_i)}{P(x)} \tag{2.13}$$

As the denominator $P(x)$ in equation 2.13 is the same for all classes, to compare the value of $P(y_i|x)$, it can be assumed that the values of the different feature attributes are conditionally independent [Leu07], meaning the occurence of a sample x_a does **not** influence the occurrence of a sample x_b hence the name *naive*, i.e.,

$$P(x|y_i) \approx \prod_{k=1}^{d} P(a_k|y_i) \tag{2.14}$$

The probabilities $\{P(x_1|y_i), P(x_2|y_i), ..., P(x_d|y_i)\}$ can be estimated from a training set. Using these probabilities, the trained NBC can then classify an unlabeled sample x.

The process of training and utilizing the NBC is outlined in Alg. 4. The algorithm starts by labeling all solutions in $\mathbf{POP_{useful}}$ as $+1$ and all solutions in $\mathbf{POP_{useless}}$ as -1. These labeled solutions are then combined into $\mathbf{X_{train}}$ and $\mathbf{Y_{train}}$, and the NBC is trained using these data. The trained classifier can then be used to predict the class of randomly generated solutions in new environments to identify high-quality solutions that can guide the search for optimal solutions in the new environment [Ye+22].

Algorithm 4 Naive Bayesian Classifier (NBC)

Input: The solution sets with useful knowledge $\mathbf{POP_{useful}}$ and with useless knowledge $\mathbf{POP_{useless}}$.
Output: A trained Naive Bayesian classifier $\mathbf{C_{NBC}}$.

```
1: $\mathbf{Label_{useful}} \leftarrow$ label all solutions $x \in \mathbf{POP_{useful}}$ as "+1"
2: $\mathbf{Label_{useless}} \leftarrow$ label all solutions $x \in \mathbf{POP_{useless}}$ as "-1";
3: $\mathbf{X_{train}} = \mathbf{POP_{useful}} \cup \mathbf{POP_{useless}}$;
4: $\mathbf{Y_{train}} = \mathbf{Label_{useful}} \cup \mathbf{Label_{useless}}$;
5: $\mathbf{C_{NBC}} \leftarrow$ train NBC by using training data $(\mathbf{X_{train}} \rightarrow \mathbf{Y_{train}})$;
```

2.4 Reinforcement Learning

Reinforcement Learning (RL) refers to a subclass of machine learning algorithms involving a so-called agent interacting with an environment to achieve a desired goal [SB98]. The concept of RL is based on the idea that an agent can improve its behavior over time in a given environment through feedback received in the form of rewards or punishments. As such, RL is a form of learning where the agent is not given explicit instructions on achieving its goal but instead has to learn from its own experience, which closely resembles human learning.

For instance, in robotics RL can be beneficial, as it allows robots to learn behaviors that are difficult to explicitly formulate in software [KBP13]. In the context of DMOO, RL has been recognized as a promising approach for solving dynamic problems that involve multiple conflicting objectives [Zou+21]. As traditional MOEAs have proven ineffective for tackling these kinds of issues, RL has recently been used by a few researchers to guide MOEAs in tracking the POF in dynamic scenarios [Zou+21] [Hua+20].

This section provides a brief overview of the basic concepts of RL without delving into more advanced techniques. This decision allows us to focus on the core principles of RL and establish a foundation for the methods used in this work.

2.4.1 Elements of Reinforcement Learning

A Reinforcement Learning system consists of two primary components: an agent and an environment, which interact through an interface with each other. The agent is the system that acts within the environment in order to achieve a specific task. The agent performs actions based on the current state it is in. The set of all possible actions the agent can take at any given time can be defined as the action space $\mathbf{A} = \{a_1, a_2, \dots, a_n\}$, where a_n represents all possible actions [SB98]. The set of all possible states the agent can find itself in can be defined as the state space $\mathbf{S} = \{s_1, s_2, \dots, s_m\}$, where s_m represents all possible states [SB98]. Both the action space $\mathbf{A}$ and the state space $\mathbf{S}$ can be either discrete or continuous, however for the purpose of this thesis, only discrete action and state spaces will be considered.

At each discrete time step t, the agent receives a reward r from the environment based on the action it took in the previous time step. The reward r is a scalar value that serves as a feedback signal to the agent. Thereby, the agent's goal is to maximize the cumulative reward received throughout a given task [SB98].

Two key concepts are essential for determining the action an agent takes at a given time step, the *policy* and the *value function*. The *policy* describes the strategy employed by an agent to decide *what* action to take for a given state. This policy can be either deterministic or stochastic [SB98], meaning that it can either provide an explicit action or a probability distribution over the action space **A**, respectively.

The second key concept is the so-called *value function*, which assesses the value of a given state s. This can be either done with a State-Value-Function $v_\pi(s)$ or a State-Action-Value-Function $q_\pi(s,a)$ [SB98]. The State-Value Function $v_\pi(s)$ specifies how high the expected reward for the agent is based on the state s. Wherein the State-Action-Value function $q_\pi(s,a)$ determines how good it is for an agent to choose an action a in a state s and then follow its policy [SB98]. The concept of policy and value function is illustrated in Figure 2.9.

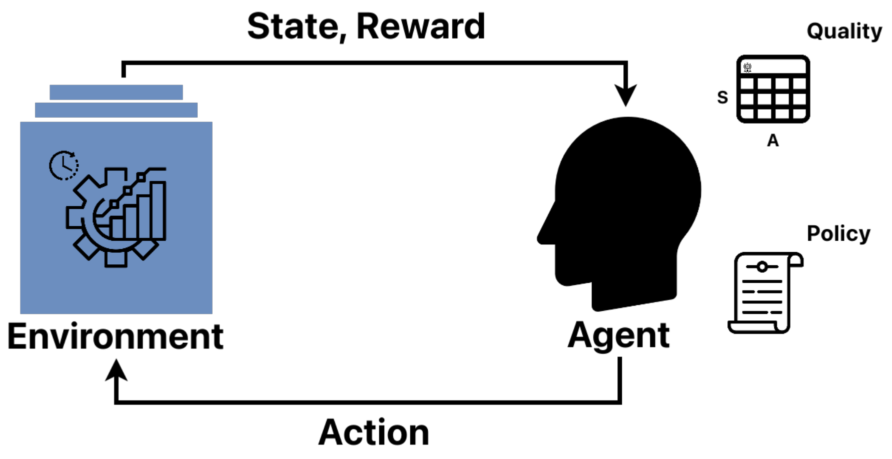


Figure 2.9: A diagram showing the RL process. Own illustration.

Some RL algorithms allow an agent to learn about its environment by building a model of it [SB98]. This allows the agent to make predictions about future states or the rewards it can expect to receive for specific actions. These methods are known as model-based approaches [Moe+23]. In contrast, model-free algorithms do not attempt to learn the dynamics of the environment but instead focus on directly or indirectly learning a policy that maximizes rewards through trial and error [SB98]. In this thesis, only model-free approaches will be considered, as they pose a reduced scope of environment complexity.

One of the challenges in RL is balancing the conflicting goals of *exploration* and *exploitation* [SB98]. Exploitation refers to the agent's tendency to choose actions that it knows will lead to high rewards. In contrast, exploration refers to the agent's need to try out new actions and discover potentially even more rewarding actions. If the agent only focuses on exploitation, it may miss out on better opportunities for reward that could be discovered through exploration. On the other hand, if the agent spends too much time exploring, it may not make the most of the knowledge it has already gained. Finding the right balance between exploration and exploitation is crucial for the agent to achieve the highest possible long-term reward [SB98].

2.4.2 Markov Decision Processes

A Markov Decision Process (MDP) is the mathematical framework used to formalize a RL problem [SB98]. Thereby, a MDP is defined as the stochastic process that satisfies the Markov Property [SB98], which states that the future state of the process depends only on the current state and not on the sequence of events that preceded it [SB98]. In other words, the future state of the process is independent of the past states, given the present state. This can be formalized with the tuple (S, A, P, R, γ) consisting of a state set S, an action set A, the transition probability function P, the reward function R, and the weighting factor γ [SB98].

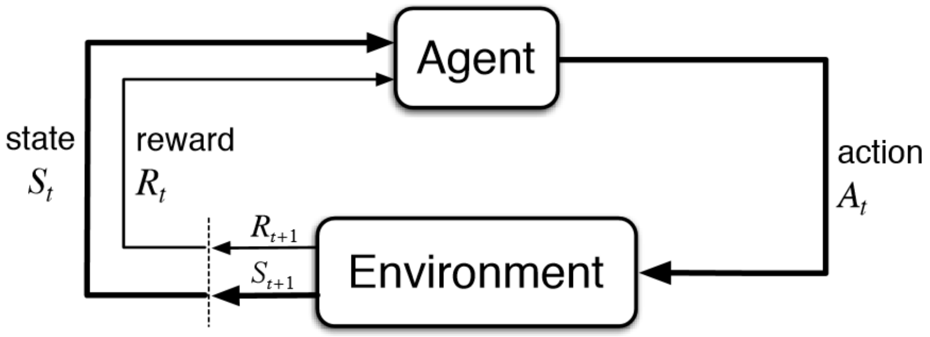


Figure 2.10: The interactions between Agent and Environment, by Sutton and Barto [SB98].

Figure 2.10 illustrates this with the Agent-Environment interaction in RL systems. At each time step t, the agent chooses an action $A_t \in A(s)$ based on its current state $S_t \in S$. As a result, the agent receives a reward $R_{t+1} \in R$ and a new state $S_{t+1} \in S$ from the environment. This generates a so-called trajectory with the following form [SB98]:

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, ... \tag{2.15}$$

For a RL problem to be considered a MDP, and therefore formally correct, it must satisfy the Markov Property, which postulates that a given state S_t must be only dependent on the previous state S_{t-1} and the action A_{t-1} [SB98]. In other words, a state has the Markov Property if it contains all future-influencing information of the past.

For this thesis, the Markov Property is assumed to hold for the given problem of using a RL agent as a prediction method selector for a DMOEA, as this significantly reduced the complexity of the environment description. However, for a more general approach to RL in DMOEA, the Markov Property would have to be proven.

The probability of transitioning from one state s to another state s' by performing an action a is described by the transition probability function p [SB98]:

$$p(s'|s,a) = Pr\{S_t = s'|S_{t-1} = s, A_{t-1} = a\} = p(s', r|s,a) \tag{2.16}$$

The resulting reward r is then given by [SB98]:

$$r(s,a) = E[S_{t-1} = s, A_{t-1} = a] \tag{2.17}$$

With E being the expected return of the agent. The goal of an agent is to increase the expected return of the reward received until reaching a terminal state. This can be also described as to maximize the expected return G_t, which can in the simplest case be defined as the sum of the rewards [SB98]:

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + ... + R_T \tag{2.18}$$

Where T denotes the last time step and therefore the end of a so-called episode.

In episodic tasks, the interaction with the environment is divided into independent sections on its own. Tasks without terminal states are referred to as continuous tasks. In order to avoid the calculated return from Equation 2.18 approaching infinity due to an unrestricted time horizon, generally future rewards are weighted as follows [SB98]:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{2.19}$$

Where $0 \leq \gamma \leq 1$ denotes the weighting factor, which indicates how strongly future rewards are included. With a weighting factor of $\gamma = 0$, only the current reward is considered. With $\gamma = 1$, no weighting is applied, so all future rewards are included equally. Setting a weighting factor between 0 and 1, results in the relative importance of future rewards decreasing.

Ultimately, the goal of an agent is to maximize the expected return G_t by choosing the best action a_t for a given state s_t [SB98]. There are numerous approaches in the RL literature with differing methods to maximize the expected return G_t, all generally varying in the use of *policies* and *value functions* used to reach maximum G_t [SB98]. For this thesis, Q-Learning as one of the most used approaches in RL is used [CL20], as it a simple model-free approach and yet effective for RL [Zou+21].

2.4.3 Q-Learning Algorithm

As described in subsection 2.4.1 and mathematically formalized in subsection 2.4.2 in RL, an agent aims to learn feedback by interacting with a dynamic environment. In other words, RL focuses on how the agent determines the action in the environment to achieve maximum cumulative reward [SB98].

Q-Learning is a model-free, Reinforcement Learning algorithm, that is generally considered as one of the breakthroughs algorithms in RL [CL20]. With *off-policy* algorithms such as Q-Learning, the agent learns based on experience tuples that were collected with a given policy [CL20]. This is then used to estimate the accumulative rewards of performing an action a for a given state s [CL20].

The Q-Learning algorithm is based on the following mathematical formalization [CL20]:

Given, $S = \{s_1, s_2, ..., s_n\}$ denoting a set of states, and $A = \{a_1, a_2, ..., a_n\}$ denoting a set of actions, and r_t as the immediate reward for executing action a_t, one can iteratively update the Quality-value Q for a given combination of state s_t and action a_t, at time t with following formula [CL20]:

$$\underbrace{Q(s_t, a_t)}_{\text{new quality}} \leftarrow \overbrace{Q(s_t, a_t)}^{\text{current quality}} + \underbrace{\alpha}_{\text{learning rate}} [\overbrace{r_t}^{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \overbrace{\max_a Q(s_{t+1}, a)}^{\text{estimate of next quality}} - \overbrace{Q(s_t, a_t)}^{\text{current quality}}] \tag{2.20}$$

Where α is the learning rate and γ is the discount factor. The learning rate α is a parameter that controls the extent to which the newly acquired information overrides older information. The discount factor γ is a parameter that controls the importance of future rewards.

For further clarification, the Q-Learning algorithm is depicted in Alg. 5 [CL20]. The algorithm begins by initializing the Q-Values for each State-Action pair in a so-called Q-Table and any necessary parameters with line 1. It then enters a loop for each episode, where an episode is a single run of the agent through a given environment, with line 2. The algorithm then initializes the current state s_t, at the start of each episode in line 3. With line 4, it then enters a loop for each step t of a current episode. At each step, the algorithm chooses an action, a_t, from the current state, s_t, using a policy as seen in line 5. Continuing with line 6, the agent then takes this action and observes the reward, r_{t+1}, and the next state, s_{t+1}. The Q-Value for the previous State-Action pair is then adjusted using the observed reward and the maximum Q-Value of the possible actions in the next state. This is done using the equation shown in 2.20:

A defining step of the Q-Learning algorithm is to choose the action a_t depending on s_t using a policy as shown in line 5. As introduced in 2.4.1 a policy can be broadly defined as a rule set which action to take. As described with the Q-Learning algorithm, the agent iteratively updates (learns) the quality pairs $Q(S, A)$, meaning the expected reward of executing an action $a \in A$ at a given state $s \in S$. Now a fundamental problem is *how to choose* the action a_t at each step t of the episode.

Algorithm 5 Q-Learning Algorithm

```
1: Initialize $Q_(s_t, a_t)$ arbitrarily and parameters;
2: **for** each episode **do**
3:     Initialize $s_t$;
4:     **for** each step of episode **do**
5:         Choose $a_t$ from $s_t$ using *policy*;
6:         Take action $a_t$ and observe $r_{t+1}$ and $s_{t+1}$;
7:         Adjust Q value with 2.20;
8:         $s_t \leftarrow s_{t+1}$;
9:     **end for**
10: **end for**
```

This is where the policy comes into play. One widely used Q-Learning policy is the ϵ-greedy action selection policy [CL20], which is a policy that chooses the action with the highest Q-Value with probability ϵ and a random action with probability $1 - \epsilon$ [CL20]. Meaning with an $\epsilon = 1$ a RL agent will always choose the action it deems best at a given time t and with an $\epsilon = 0$ it will always choose a random action from $Q(S, A)$. In other words, ϵ balances the *exploitation* of the current knowledge of an agent versus *exploring* new actions, that could potentially provide higher quality. The ϵ-greedy policy is depicted in Alg. 10 [CL20].

Algorithm 6 ϵ-greedy Action Selection

```
**Input**: The current Q-Table $Q(S, A)$, the current state $s_t$
**Output**: An action $a_t$
1: $n \leftarrow random.uniform(0, 1)$;
2: **if** $n < \epsilon$ **then**;
3:     $a_t \leftarrow$ random action $a \in A$;
4: **else**
5:     $a_t \leftarrow max\ Q(s_t, .)$;
6: **end if**
```

3. Related Works

In recent decades, a variety of DMOEAs have been proposed as efficient tools for DMOPs. This chapter gives an overview of the most relevant DMOEA approaches. First, Diversity-Based approaches are presented in section 3.1. Subsequently, section 3.2 presents DMOEA approaches using Memory-Based response strategies. After that, an overview of Prediction-Based strategies is given in section 3.3. A summary of relatively new Dynamics-Based approaches, where RL falls under, is presented in section 3.4. Finally, in section 3.5, the research gap filled with this thesis is outlined.

3.1 Diversity-Based Strategies

In the DMOEA literature, two approaches for introducing diversity, namely *diversity increase* strategies as well as *diversity maintenance* strategies have been widely used [JZY22]. Diversity increase strategies involve immediately adding randomly generated solutions to a given population as soon as change is detected [JZY22]. Whereas diversity maintenance strategies refer to the hypermutation of some existing solutions on the detection of change [JZY22].

The difference between both strategies can be exemplified with the DNSGA-II algorithm by Deb et al. [DRK07], where the authors integrated these two approaches into their own previously proposed Non-Dominated Sorting Genetic Algorithm (NSGA-II) [Deb+02]. NSGA-II is a SMOEA which is similar to the general MOEA framework presented in section 2.2.2 and is often used as a SMOEA for comparison purposes of DMOEA implementations [Zou+21]. With DNSGA-II-A and DNSGA-II-B, Deb et al. proposed two varying DMOEAs, with the main difference between the two being that on the detection of environmental change, DNSGA-II-A replaces approx. 20% of a given population with randomly generated solutions, unrelated to the existing population [JZY22]. Wherein contrast, DNSGA-II-B replaces approx. 20% of the population with mutated solutions of randomly chosen existing solutions. With this, the algorithm generates adds solutions to the population that are related to the existing one.

DNSGA-II-A has shown to be well-suited for handling severe environmental changes that result in a significant loss of diversity, as the algorithm significantly diversifies the population through the introduction of new solutions [JZY22]. In contrast, DNSGA-II-B is more appropriate for dealing with subtle environmental changes, as it introduces small variations to the population through the use of mutated solutions [JZY22]. As a result both diversity introduction mechanisms have been integrated into numerous classical EAs for handling DMOPs according to Jiang, Zou and Yao [JZY22] [ABB17] [RY13].

With this in mind, in the following subsection 3.1.1 first, DMOEA research employing diversity increase is presented, followed by diversity maintenance strategies in subsection 3.1.2.

3.1.1 Diversity Increase Strategies

Goh and Tan [GT09] proposed a more rigorous method for introducing diversity into the population. Rather than simply replacing existing solutions with randomly generated ones, their approach involves having the randomly generated solutions compete against representatives of the existing solutions [GT09]. Only the winners of these competitions are then used to populate the new environment.

Zhou et al. [Zho+07a] evaluated two new approaches for introducing diversity into dynamic multi-objective optimization problems: random reinitialization and Gaussian mutation. These approaches involve completely replacing the entire population with new solutions, rather than just making changes to a small portion of it. Additionally, no past solutions are retained for new environments. Studies have shown that simply randomly reinitializing the entire population is not effective in dynamic environments, as it lacks previous knowledge [Zho+07a]. However, using a Gaussian mutation technique on the entire population has been found to be effective, especially in situations where the environment is not rapidly changing.

An exploration strategy for increasing diversity was introduced by Peng et al. in 2015, which is based on the population centroids of the previous and current environments [Pen+15a]. The approach consists of determining the direction in which each variable is moving towards the optima, and then using this information to sample new variable values along those directions.

Zheng et al. [Zhe+22] proposed an approach for introducing diversity through the use of both hypermutation and random reinitialization. This approach involves creating a new population for the new environment using both hyper-mutated solutions of past elite solutions from an archive and randomly generated solutions. The probability of using past elite solutions in the new environment, when using hypermutation, is correlated with the number of elite solutions in the archive. Specifically, when there are more elite solutions from the previous environment, the likelihood of them being used in the new environment increases.

Wang and Li [WL10] proposed a diversity introduction approach similar to that of Zheng et al. [Zhe+22]. Their approach also involves a combination of hypermutation and random reinitialization, but differs mainly in the hypermutation schemes used. Furthermore, the authors include incorporating a specified proportion of Gaussian-modified solutions from previous solutions into the population. When combined with other techniques, this strategy has shown to be effective on a range of DMOPs, as reported in [WL10].

Ma et al. [Ma+21] recently proposed a method for introducing new, randomly generated solutions to the population that are distributed across different regions of the objective space, in order to ensure diversity in the population.

In summary, a variety of approaches have been proposed for introducing diversity into the population in DMOPs. These approaches range from simple techniques, such as random reinitialization [Zho+07a] or Gaussian mutation [WL10] to more complex methods involving competition between existing and randomly generated solutions [GT09], or sampling new solutions based on the population centroids of previous and current environments [Pen+15a].

3.1.2 Diversity Maintenance Strategies

According to Jiang, Zou and Yao [JZY22], the ALife-inspired algorithm [AF05] was the first DMOEA to utilize Pareto dominance coupled with diversity maintenance strategies for dynamic multi-objective optimization. The algorithm uses artificial operators to emulate interactions between individuals in a population with variable size. These interactions include meeting, fighting, and reproduction and help to preserve diversity within the population resulting in the capability to adapt to changes of dynamic environments.

Zeng et al. [Zen+17] introduced an orthogonal design-based evolutionary algorithm that is able to maintain high diversity over time for dynamic environments. The reproduction procedure for this algorithm involves randomly selecting one of two types of crossover operations: orthogonal crossover [Zen+17] based on orthogonal design or linear crossover [Zen+17]. By introducing a diverse set of solutions into the offspring population, the algorithm is able to explore a wider range of the search space. This method of maintaining diversity has been shown to be successful in solving bi-objective dynamic multi-objective optimization problems, as demonstrated in research conducted by Zeng et al. [Zen+17].

Chen et al. [Che+20] proposed the use of an additional objective focused on maintaining population diversity during the search for dynamic multi-objective optimization problems. The authors propose to measure population diversity using an entropy metric, as described in [Che+20]. The goal is to optimize both the main objective functions and the diversity objective, thus maintaining a diverse population throughout the optimization process. Studies have shown that this approach is effective for certain types of benchmark problems, as reported in [Che+20].

Wang and Dang [WD08] proposed the use of a uniform design-based crossover for population reproduction in order to maintain high population diversity over time. The key idea behind this approach is that uniform design can prevent close parents from being crossed over, enabling wider exploration in the search space and helping the algorithm to adapt to environmental changes. Additionally, they also transformed dynamic multi-objective optimization problems into smaller ones through objective redefinition in order to improve search efficiency [WD08].

Chen, Li and Yao [CLY18] proposed a two-archive structure with one archive focused on maintaining diversity and the other focused on population convergence. The algorithm utilizes two archives that evolve together over time, enabling it to manage a varying number of objectives. While the results reported by the authors are encouraging, the computational inefficiency of this structure was acknowledged in [CLY18].

Overall, the research on diversity maintenance strategies has shown that diversity maintenance provides a valid alternative to diversity increase strategies.

3.2 Memory-Based Strategies

Another approach for handling environmental changes in dynamic multi-objective optimization is to store solutions from past environments in a memory pool [JZY22]. These solutions may be useful for the current environment, mainly if it is similar to a past one [JZY22]. This approach involves selectively saving solutions from the evolving population and then retrieving and merging them back into the population as needed. It is important to note that both Memory-Based strategies and Prediction-Based approaches rely on past solutions to guide the search in new environments. However, Memory-Based strategies differ in that they do not use a predictive model. This section will briefly cover Memory-Based approaches to provide a comprehensive overview of current research.

Koo, Goh and Tan [KGT10] proposed storing non-dominated sets found in previous environments in a memory pool. Upon the occurrence of a change, a specified number (γ) of past solutions are uniformly sampled from the memory pool by iteratively removing the most crowded solution in the decision space from a copy of the memory pool until γ solutions remain. These γ past solutions, along with other solutions generated using normal mutation, make up a large portion of the population for new environments.

Goh and Tan [GT09] proposed a DMOEA that utilizes a memory pool to store solutions from previous environments. In this approach, similar to Koo, Goh and Tan [KGT10] proposed algorithm [JZY22], γ random solutions are selected from the previous population, with a preference for extreme solutions along the objectives. These solutions are then added to a memory pool of fixed size just before an environmental change. When the memory pool is full, the oldest solutions are removed to make room for new ones.

An approach was proposed in Peng et al. [Pen+15b] where non-dominated solutions for each previous environment are stored in a memory pool with a fixed size. When the memory pool reaches its capacity, the first-in-first-out rule is applied. In new environments, the solutions in the memory pool are utilized to replace the lowest-performing individuals of the current population.

In their work, Zhou, Jin and Zhang [ZJZ13] proposed a dynamic environmental evolutionary model that incorporates information about past environments and search experience in order to guide the search in new environments. This approach allows the population to build a record of its experiences and use this knowledge to adapt to changes in the environment.

3.3 Prediction-Based Strategies

Predictive models can be used to anticipate the location of new optimal solutions in dynamic environments that exhibit a regular pattern of predictability [JZY22]. By using historical information to predict the changing POS or POF in new environments, Prediction-Based DMOEAs can rapidly adjust the composition of a given population, leading to faster convergence.

Depending on the complexity of environmental changes the predictive models used can generally be divided into linear and non-linear models [JZY22]. In the following, the most common predictive models in DMOEA are reviewed first for linear models in 3.3.1 and thereafter for non-linear models in 3.3.2.

3.3.1 Linear Predictive Models

Zhou, Jin and Zhang [ZJZ13] introduced the population prediction strategy (PPS), which uses previous midpoints of POS and previous manifolds to predict the next midpoint and next manifold [ZJZ13]. The general assumption of this approach being, that some similarities exist between manifolds of previous environments. Then, the population for the new environment can be generated by combining the predicted POS center and manifold. The algorithm is widely adopted in numerous DMOEAs [PZZ14].

Koo, Goh and Tan [KGT10] proposed a predictive model that generates solutions as part of a population. The model includes the concept of a predictive gradient, which is a weighted sum of the previous gradient and changes in the population centroid calculated from a memory pool.

Wu, Jin and Liu [WJL15] proposed a special directed search strategy (DSS) [WJL15]. In this method, the predicted direction of which the Pareto front is used to reinitialize a new population for solving DMOPs.

Multiple linear predictive models have also been proposed in order to improve prediction accuracy. rongMultidirectionalPredictionApproach2019 developed a model selection approach in which the type of change in the Pareto-Optimal set is detected and the most suitable predictive model is used based on the detected type. This approach involves using multiple predictive models for population prediction.

Liang et al. [Lia+19] proposed a method for dealing with environmental changes using predictive models based on their similarity to historical changes. They developed a procedure for assessing the similarity of changes and classifying them accordingly. If a change is similar to any historical changes, a Memory-Based technique is used for population prediction. If the change is not similar to any historical changes, a simple predictive model is used.

3.3.2 Nonlinear Predictive Models

Nonlinear predictive models are designed to represent more complex and non-linear nature of environmental changes [JZY22]. These models generally are more effective at capturing the intricacies of such changes, but they may also be more difficult to compute during algorithm execution with the requirement of more data to accurately model the relationships between variables compared to linear predictive models.

Jiang et al. [Jia+18a] argued that incorporating transfer learning approaches into evolutionary algorithms can greatly improve the ability to track the moving POS and POF of DMOPs [Jia+18a]. They employed a domain adaptation method to build a predictive model that learns from past populations to locate the POS [Jia+18a]. This idea has also been combined with other strategies and applied to estimation of distribution algorithms for solving DMOPs.

Liu et al. [Liu+14] proposed a combination of the population prediction strategy approach [ZJZ13] and transfer learning [WKW16] to improve population prediction. This approach aims to improve the accuracy of predicting the changing

Fan, Li and Tan [FLT20] used transfer learning to address the challenges of expensive DMOPs. In their study [FLT20], transfer learning was employed to improve prediction accuracy by leveraging ”knee points”, or specific points of inflection in a curve. This approach may be particularly useful for situations in which the cost of evaluating the objective functions is high and efficient optimization is necessary [FLT20].

In more recent times, Ye et al. [Ye+22] introduced Knowledge Guided Bayesian Classification for DMOEA (KGB-DMOEA). This method involves clustering historical solutions based on similarity and using Bayesian classification to classify randomly generated solutions based on their potential usefulness. The resulting initial population can then guide a standard SMOEA in a new problem environment.

3.4 Dynamics-Based Strategies

Generally, having an understanding of environmental changes can help DMOEAs at effectively addressing them. Dynamics-Based approaches constitute a relatively new branch of response strategies in DMOEA [JZY22], attempting to cope with environmental changes by evaluating their impact on the optimization procedure.

One way for algorithms to adapt response strategies is by dynamically adjusting those strategies based on the severity of change detected in the optimization environment [JZY22]. The most common Dynamics-Based strategies do this based on the severity of detected change and are reviewed in the following subsection. It is to be noted that other approaches for Dynamics-Based response strategy adaption are known in literature [JZY22], however in the interest of maintaining the scope and focus of this thesis, only change severity-based strategies will be reviewed.

Change Severity Based Strategies

Sahmoud and Topcuoglu [ST19] developed an algorithm for solving DMOPs that is based on detecting the type of environmental change. The algorithm calculates the magnitude of the change by comparing the number of non-dominated solutions in the previous and current environments. If the ratio of the difference to the population size is greater than a predefined threshold, the change is classified as Type-I or Type-II [ST19]. Otherwise, it is considered a Type-III or Type-IV change. The algorithm subsequently uses DNSGA-II-A [DRK07] in the first case, and NSGA-II [Deb+02] with hypermutation in the latter.

Zhang [Zha08] developed an environmental recognition rule that is based on the severity of changes and can be used to classify a new environment as identical, similar, or dissimilar to the past environment. The classification result determines the appropriate actions to be taken.

Azzouz, Bechikh and Said [ABS17] proposed a severity-based change response approach that is based on the idea that if a severe change occurs, it is more likely that a better solution can be generated from the current solution because the current solution is more likely to be very close to the new POF. They employed a local search technique to create trial solutions from the population and gauge the degree of the change. When the change is minor, it is then advised to use more archived solutions than randomly generated ones in the new population. In case of severe changes, more randomly generated solutions are suggested. The effectiveness of this approach in monitoring the shifting POS and POF when incorporated into NSGA-II was shown by Azzouz, Bechikh and Said [ABS17].

Recently, Zou et al. [Zou+21] put forward a method for identifying the degree of environmental changes by assessing the extent of variation in the objective values of detectors. They constructed a Reinforcement Learning strategy to adapt to these changes based on the classification of the change into three levels: light, medium, and severe. These categories are considered as states in the Reinforcement Learning procedure. Three actions including a knee-based prediction, a center-based prediction, and local search are applied to shift the population to the new POF. The Reinforcement Learning strategy chooses a sequence of actions based on the states provided.

3.5 Research Gap

The field of DMOEA has seen significant research in recent years, focusing on developing new algorithms and strategies to handle environmental changes during the optimization process. One of the main approaches for addressing these changes is Prediction-Based methods, which rely on historical data to anticipate the location of new optimal solutions. These methods, coupled with diversity maintenance mechanics, effectively solve DMOPs [JZY22]. However, there is relatively little research on Dynamics-Based strategies for addressing environmental changes in DMOPs.

Dynamics-Based approaches aim to cope with environmental changes by evaluating their impact on the optimization procedure, and have shown promising results in identifying changes in the optimization environment. Work by Zou et al. [Zou+21] recently addressed the issue that most existing Dynamics-Based methods apply prediction response mechanisms at the time of environmental change without considering the prior history of the environmental change, resulting in the waste of valuable environmental information as Zou et al. [Zou+21] state. Their approach uses RL to formulate a goal-directed agent that chooses an optimal prediction method from a set of Prediction-Based methods depending on the magnitude of the environmental change, thereby improving the agent's ability to gauge the utility of a prediction method through experiences. However, the approach's shortcoming is that it does not take the computational time of the prediction models into account, which can be very valuable in ensuring overall computational efficiency. Furthermore, there is a lack of explicit inquiry into managing the trade-off between exploration and exploitation, which is a critical factor in RL.

On the other hand Ye et al. [Ye+22] proposed a novel DMOEA that, through the use of a KRE coupled with a NBC as a prediction model presented promising results in dynamic benchmark problems. KGB-DMOEA has shown to be effective in solving DMOPs, surpassing most previous Prediction-Based DMOEAs in terms of performance. This is achieved through the use of and transferring extracted knowledge to new environments.

Despite the noted successes of the KGB-DMOEA, it does possess certain limitations that merit consideration. One such limitation is the utilization of a single prediction model with the NBC. This presents an issue as the appropriateness of a prediction model is dependent on various factors, such as the intensity of the environmental change and the nature of the change pattern exhibited by a particular optimization problem, as discussed by Jiang, Zou and Yao [JZY22]. Thus, the exclusive utilization of a single prediction model may not be appropriate for all dynamic problem environments encountered and may result in suboptimal performance. Additionally, another possible constraint of KGB-DMOEA is the knowledge transfer process. As the algorithm consistently examines all historical information, compressed into clusters, the optimization process may be computationally intensive, particularly when the number of historical environments is substantial. This computational burden may be unnecessary in situations where the environmental change is not severe or follows a linear pattern, for which simpler prediction models may suffice.

Therefore, the main motivation of this thesis is to address this gap by proposing a new algorithm that combines the strengths of Prediction-Based methods and Dynamics-Based strategies.

4. Proposed Algorithm

The following chapter presents a detailed description of the proposed algorithm. In the first section 4.1, the general framework of the proposed algorithm is introduced. Following this, section 4.2 presents the change severity detection method, on which basis environmental states are defined. Section 4.3 lays out the possible actions the RL agent can take based on the detected severity of a given problem landscape change. A boundary detection mechanism is introduced in section 4.4 to mitigate the risk of false predictions in high-dimensional decision spaces. In section 4.5, the State-Action table of the proposed algorithm is presented. The reward function, which is used as a feedback signal in RL, is presented in section 4.7. Section 4.6 describes the policy applied for the RL agent. Finally, relevant hyperparameters are reviewed in section 4.8.

4.1 General Framework of the Proposed Algorithm

DMOPs are a challenging class of optimization problems that involve multiple conflicting and time-varying objectives. The purpose of this thesis is to study improvement potentials of the KGB-DMOEA as a Prediction-Based DMOEA by wrapping KGB-DMOEA into RL, with the premise that a RL agent should learn from dynamic changes in the environment and then determine the appropriate action to take in the new environment. The overall approach is illustrated in Fig. 4.1.

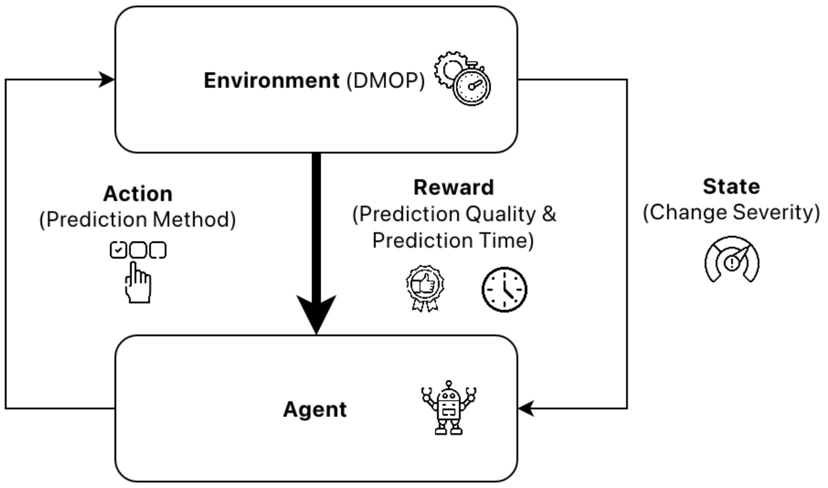


Figure 4.1: Concept of the general proposed algorithm. Own illustration.

For this, Q-Learning is used where the agent asses the quality of executing an action for a given environmental state and then decides on an action based on its policy. One of the key contributions of this work is the addition of a decaying ϵ-greedy policy, which is used to encourage comprehensive exploration at the start of optimization runs and will be introduced in more detail in section 4.6. At this stage, it is also important to clarify that the optimization problem and the computer system on which the same problem is solved are both considered as the environment for Q-Learning.

The proposed algorithm is based on the RL framework proposed by Zou et al. [Zou+21] and extends it with KGB-DMOEA. In Fig. 4.2, the flowchart of the proposed algorithm is depicted. The flow of the proposed algorithm can be divided into four parts. The first part consists of the initialization of the starting population and all algorithm parameters. The second part of the proposed algorithm is the main outer loop that follows the general framework of DMOEAs of checking for environmental change, and in case of no detected change optimizing the given DMOEA with a standard SMOEA. As soon as change is detected, the proposed algorithm jumps into the third part of the algorithm, which is checking for boundary situations. This boundary detection mechanism is a key change compared to the by approach Zou et al. [Zou+21] introduced in more detail with 4.4, as this gives the possibility to use other methods for boundary situations.

For this work, the KGB-DMOEA algorithm is always used in boundary situations, with the assumption that a knowledge-based algorithm should always result in better handling of boundary situations than simple linear prediction algorithms. This assumption is based mainly on two observations. (1) For many optimization problems, especially when constrained, optimal solutions often fall on the boundaries of a given feasible space [SM97]. Simple linear prediction models usually base the predicted population by evaluating a distance measure based on the POF or POS, from a last environment to a new changed environment. However, in dynamic scenarios, the *severity* of change is a key factor, as it is possible for change to be severe enough that a simple linear model would "overshoot" the boundaries of a given problem and therefore land directly on the boundaries of a given decision space. This can result in significant performance degradation in cases of the true POS lying just enough before the boundaries of a problem that subsequent SMOEA has to revert the "overshooting" of the linear prediction model. This problem is further explained in section 4.4 and visualized with Fig. 4.5. (2)Furthermore, in dynamic contexts, *history* matters, especially at boundary situations. To exemplify this issue, one can use the concept of residual stress found in metals as an analogy [Wit07]. In engineering, residual stress refers to remaining stress in a solid material even after the original cause of the stresses has been removed [Wit07]. This can happen when a material is bent over a certain threshold, hindering it from relaxing to its original position. Transferring this back to boundary situations in DMOO, one can argue that when optimizing for a given time period t and hitting a variable boundary (threshold), the system under optimization might behave differently in the subsequent periods $t_{+}1, ...t_{n}$. Having this in mind it stands to reason, to use the KGB-DMOEA exclusively in such a situation, as the training of the NBC employed in KGB-DMOEA should better capture the behavior of the system under optimization in such situations.

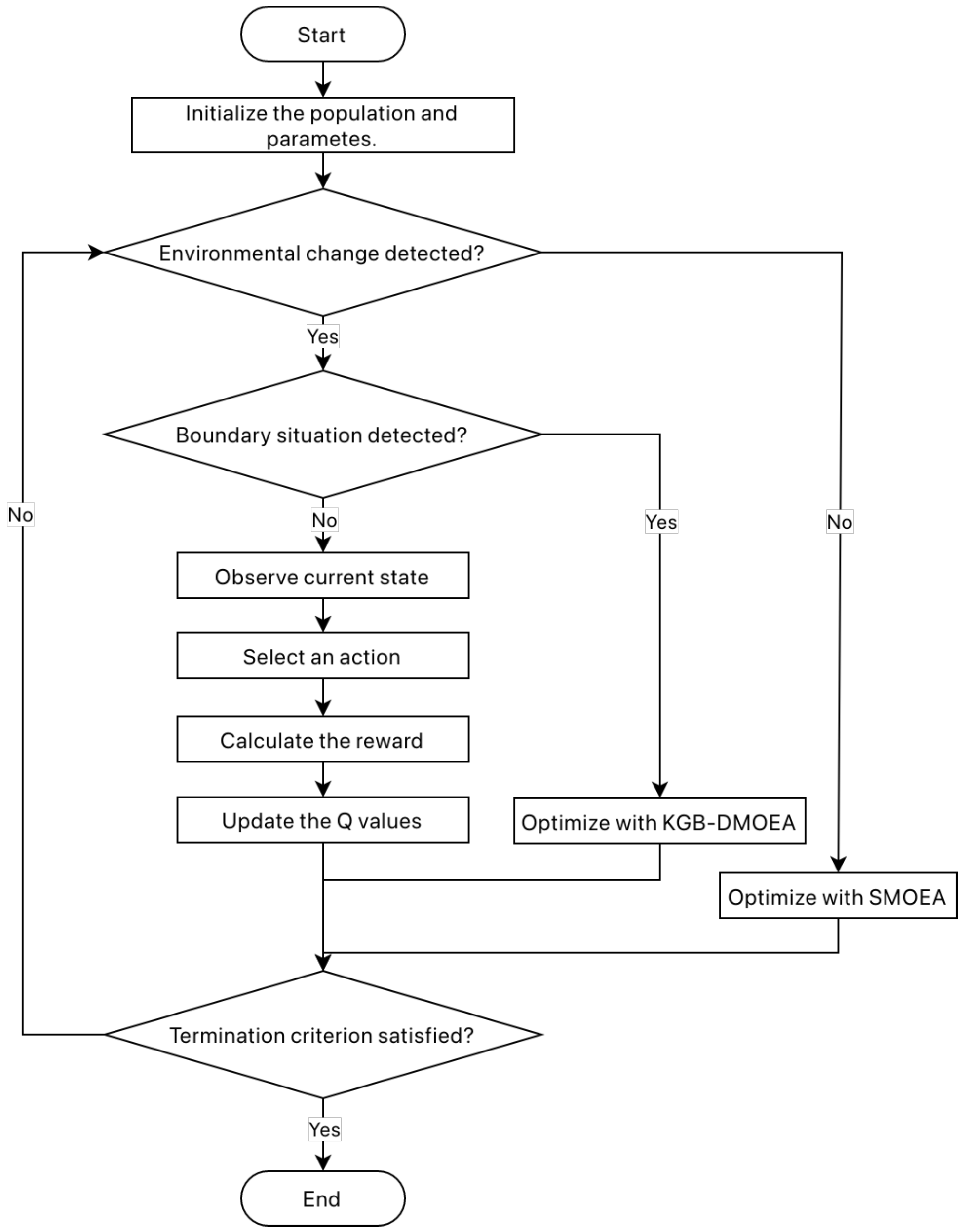


Figure 4.2: Illustration of the RL framework for the proposed algorithm. Own illustration.

If no boundary situation is detected, the fourth part of the proposed algorithm is executed, which is the RL part. In this part, the RL agent is trained with Q-Learning and the action selection policy is applied. The proposed algorithm is then repeated until a given termination criterion is met. With this structure, the proposed algorithm should for the most time run the outer loop and optimize a given problem landscape with a standard SMOEA. In case of a detected change, the RL part of the algorithm is executed. Only for the exception to boundary situations, where the RL part of the algorithm is skipped and KGB-DMOEA is used as a fail-safe mechanism.

For a more precise description the pseudocode in Alg. 7 presents a general outline of the proposed algorithm.The algorithm begins with the initialization of the historical solution archive, $\boldsymbol{PS}$, that is initialized as an empty set, the discrete time variable $\boldsymbol{t}$ is initialized to 0, and the variables $\boldsymbol{\rho_{max}}$ and $\boldsymbol{\mu_{max}}$ are initialized to 0 in line 1. With line 2 the sets of Q-Values, $\boldsymbol{Q(s, a)}$, for each pair of state, $\boldsymbol{s}$, and action, $\boldsymbol{a}$, are initialized as empty sets. The sets of Q-Values are stored in a table which can be referred as the Q-table. Line 3 generates an initial population of size $\boldsymbol{N}$ using the ***initpop*** function. The algorithm then enters a while loop that continues until a stopping criterion is met as seen in line 4. If a change is detected (line 5), the variable $\boldsymbol{t}$ is incremented by 1 (line 6). If a boundary situation is detected (line 7), the DMOP is optimized using the KGB-DMOEA method (line 8). If there is no boundary situation detected (line 9), the current state, $\boldsymbol{s}$, is observed using a change severity detection method (line 10). An action, $\boldsymbol{a}$, is then selected from the set of actions, $\boldsymbol{\{a_1, a_2, a_3\}}$, using an action selection policy (line 11). The prediction score, $\boldsymbol{\rho_t}$, and execution time, $\boldsymbol{\mu_t}$, are then calculated (line 12). The values of $\boldsymbol{\rho_{max}}$ and $\boldsymbol{\mu_{max}}$ are updated if $\boldsymbol{\rho_t}$ is greater than $\boldsymbol{\rho_{max}}$ or $\boldsymbol{\mu_t}$ is greater than $\boldsymbol{\mu_{max}}$, respectively (lines 13 and 14). A reward, $\boldsymbol{r(\rho_{max}, \rho_t, \mu_{max}, \mu_t)}$, is then calculated based on the maximum prediction score and execution time, as well as the current values of prediction score and execution time (line 15). The next state, $\boldsymbol{s'}$, is observed (line 16) and the Q-Values, $\boldsymbol{Q(s, a)}$, are updated using a Q-Learning update rule (line 17). If no change is detected (line 5), the MOP is optimized using a SMOEA method (line 18). The algorithm continues to iterate until the stopping criterion is met.

A similar approach was proposed by Zou et al. [Zou+21] for integrating RL into DMOEA. However, the proposed algorithm of this thesis differs from the approach devised by Zou et al. [Zou+21] in the following key aspects:

1. The proposed algorithm integrates KGB-DMOEA, as a non-linear prediction model to respond to environmental changes. As KGB-DMOEA explicitly transfers historical information (POS) of *all* previous problem landscapes, it can be assumed that KGB-DMOEA offers better solution prediction capabilities compared to simpler linear response strategies, as introduced in the work of Zou et al. [Zou+21].

2. As the goal of this thesis is to examine methods to reduce computational cost of KGB-DMOEA, without compromising too much on performance, the proposed algorithm employs a unique reward function based on predicted solution quality *and* computational cost measured in wall clock time. The rationale here being that by rewarding a RL agent not only for prediction solution quality, but also for the time it took to predict those solutions an agent should be able

Algorithm 7 Algorithm of the proposed algorithm

Input: DMOP, $\boldsymbol{F_t(x)}$; population size, $\boldsymbol{N}$; population sample size, $\boldsymbol{\eta}$; learning rate, $\boldsymbol{\alpha}$; discount factor, $\boldsymbol{\gamma}$; exploration probability, $\boldsymbol{\epsilon}$; exploration factor, $\boldsymbol{\epsilon_d}$; cluster size, $\boldsymbol{N_c}$; boundary threshold, $\boldsymbol{\theta}$;
Output: approximated POS;

```
1: Initialize $\boldsymbol{PS} = \emptyset$, $\boldsymbol{t} = 0$, $\boldsymbol{\rho_{max}} = 0$, $\boldsymbol{\mu_{max}} = 0$;
2: Initialize sets of Q-Values $\boldsymbol{Q(s,a)} = \emptyset$ for each $\boldsymbol{(s,a)}$ pair;
3: Randomly generate initial population $\boldsymbol{initpop(N)}$;
4: while stopping criterion not met do
5:     if change detected then
6:         $\boldsymbol{t} = \boldsymbol{t} + 1$;
7:         if boundary situation detected then
8:             Optimize DMOP with KGB-DMOEA;
9:         else
10:            Observe current state $\boldsymbol{s}$ with change severity detection method;
11:            Select action $\boldsymbol{a}$ from $\boldsymbol{(a_1, a_2, a_3)}$ with action selection policy;
12:            Calculate prediction score $\boldsymbol{\rho_t}$ and execution time $\boldsymbol{\mu_t}$;
13:            Update $\boldsymbol{\rho_{max}} = \boldsymbol{\rho_t}$ if $\boldsymbol{\rho_t} > \boldsymbol{\rho_{max}}$;
14:            Update $\boldsymbol{\mu_{max}} = \boldsymbol{\mu_t}$ if $\boldsymbol{\mu_t} > \boldsymbol{\mu_{max}}$;
15:            Calculate reward $\boldsymbol{r(\rho_{max}, \rho_t, \mu_{max}, \mu_t)}$;
16:            Observe next state $\boldsymbol{s'}$;
17:            Update Q-Values $\boldsymbol{Q_(s,a)}$ with Q-Learning update rule;
18:        end if
19:    else
20:        Optimize MOP with a SMOEA;
21:    end if
22: end while
```

to learn what prediction method is *sufficiently good* given a change event in the optimization problem.

3. Furthermore, a boundary detection mechanism is introduced as an essential component of the proposed algorithm to mitigate the risk of false predictions. In high-dimensional decision spaces, one can generally assume boundary situations to be more likely and costly to algorithm performance. Simple linear prediction models, struggle to predict the *direction* of change at boundary situations. Change in such situations can either approach marginally toward the boundaries of the decision space or revert in a direction more towards the center of the decision space. To mitigate this problem, the proposed algorithm employs a boundary detection mechanism to detect boundary situations and uses KGB-DMOEA in such situations. As KGB-DMOEA is a non-linear prediction method, it is able to stabilize the prediction of the POS in such situations, if it has previously encountered similar situations.

4. As a last significant difference to the approach presented by Zou et al. [Zou+21] a *decaying* epsilon-greedy action selection probability is used for the proposed algorithm of this thesis. The epsilon decay in the proposed algorithm encourages the RL agent to explore more freely in the early stages of the optimization process and balance the trade-off between exploration and exploitation.

4.2 Change Severity Detection Mechanism

Detecting change is a crucial part in dynamic multi objective evolutionary optimization and can broadly be defined as the identification of a change in the problem landscape. Farina, Deb and Amato [FDA04b] proposed to classify change as change the relation between the POS and POF for a given time step and the previous time step, as described in section 2.3.

To be able to specify change, we can retake the POF-POS Relation-Based classification proposed in 2.3. There Farina, Deb and Amato [FDA04b] proposed to classify change as change the relation between the POS and POF for a given time step and the previous time step. With following Types of changes:

- **Type I**: The POS changes while the POF is unaffected.
- **Type II**: Both the POF and the POS are affected by the changes.
- **Type III**: The changes affect the POF, but the POS is unaffected.
- **Type IV**: The changes affect neither the POF nor the POS.

For the change types from type I to type III having the same POS_t at the next time step $t + 1$ would result in deviation. Having this in mind we can define change as the deviation of the objective function values for the same population which was calculated for this thesis as follows:

$$\Delta POF_t = F_t(POS_{test}) - F_{t-1}(POS_{test}) \tag{4.1}$$

Where POS_{test} is a randomly sampled percentage η, evaluated in the new environment $F_t(POS)$, of the previous population $POS_t - 1$. This percentage η is an important parameter of the change detection method, as it determines the computation cost of the change detection method.

Furthermore, we can define the severity of change by comparing it to the mean of changes encountered so far for a given optimization problem:

$$\delta = \overline{\Delta POF} - \Delta POF_t \tag{4.2}$$

Where $\overline{\Delta POF}$ is the mean of the change encountered so far and ΔPOF_t is the change encountered in the current time step. δ is consequently a measure of the change severity relative to the mean changes encountered so far.

With this we define change relative to the change encountered so far. This is used for the proposed algorithm to define three states for Reinforcement Learning, similar to the approach of Zou et al. [Zou+21]:

- **State 1**: Low severity change: $\delta < \Delta POF$.
- **State 2**: Mid severity change: $\Delta POF \leq \delta \leq \Delta POF * 1.5$.
- **State 3**: High severity change: $\delta \geq \Delta POF * 1.5$.

4.3 Actions of the Proposed Algorithm

The main limitation of the KGB-DMOEA that this thesis seeks to address is the comparatively high computational burden of the KGB-DMOEA. This computational burden stems from the fact that the algorithm by Ye et al. [Ye+22] has to extract and examine knowledge before using this knowledge as inputs for the NBC. In practice, this means the comparison of various solution clusters accumulated so far. This results in a certain overhead, that simple linear prediction models do not encounter, as they predict the next population by directly comparing the previous two populations only. This overhead of the KGB-DMOEA is to be avoided, especially when a linear prediction model could have been sufficient to predict the next population to a comparable accuracy as KGB-DMOEA. For this reason, two additional linear prediction methods are added, besides to KGB-DMOEA, for the proposed algorithm to be chosen with the help of an agent during change detection. In the following sections, these linear models are briefly presented and explained.

4.3.1 Knee-Based Prediction

One popular linear predictor used in DMOEA is the Knee-Based Prediction method [Zou+21] [Bra+04]. The method has proven to be a simple yet effective approach to predict the next population of a problem at hand [ZTJ15] [Zou+21]. To predict the next population, the knee points of two adjacent approximated POFs are first determined with a distance measure such as the Minimum Manhattan Distance (MMD) as devised by Zou et al. [Zou+21]. The algorithm for MMD knee selection is shown in Alg. 8. With this algorithm, the knee points of a POF are determined by calculating the minimum and maximum value of each objective function for the current approximated POF and then calculating the distance of each solution to the minimum and maximum value of each objective function.

Algorithm 8 Minimum Manhattan Distance Knee Selection (MMD)

```
Input: the current approximated $POS_t$;
Output: the MMD Knee solution set, $Q$;
1: for $m = 1$ to $M$ do
2:     Determine the minimum value $f_m^{Min}$ in objective $m$ from $POS_t$;
3:     Determine the maximum value $f_m^{Max}$ in objective $m$ from $POS_t$;
4:     for $i = 1$ to $|A|$ do
5:         $Dist_i = Dist_i + \frac{f_m^i - f_m^{Min}}{f_m^{Max} - f_m^{Min}}$;
6:     end for
7: end for
8: Chose the MMD knee solution with the smallest value of $Dist_i$;
9: Copy the MMD knee solution into set $Q$;
```

With these Knee-Points the moving direction of the POF can be determined by calculating the distance between the knee-point clusters K_t to the knee-point clusters $K_t - 1$ as such [Zou+21]:

$$D_t = ||K_t - K_{t-1}|| \tag{4.3}$$

Where D_t is the Euclidean distance computed between the knee-point clusters K_t and K_{t-1}. The location of the next POF can then be predicted by shifting all individuals of the current POS by D_t in the direction of the knee-point clusters K_t and K_{t-1} [Zou+21]:

$$POS_{t+1} = POS_t + D_t + \xi_t \tag{4.4}$$

Where $\xi_t \sim N(0, d)$ refers to Gaussian noise with a mean of zero and a standard deviation d [Zou+21]. The process of predicting the shifting direction of the POF is shown in Fig. 4.3.

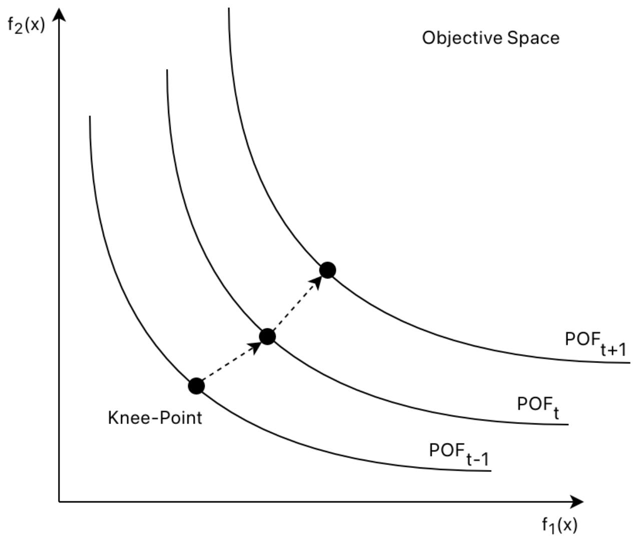


Figure 4.3: Illustration of predicting the shifting POS using the Knee-Point of previous Pareto fronts. Own illustration.

4.3.2 Center-Based Prediction

As a second linear prediction Method, the Center-Based Prediction method is used [Zho+07b]. This prediction method is similar to the Knee-Based prediction method but varies in the base that is taken for distance calculation. Instead of calculating the distance between the knee-points of the previous and current POF, the distance between the cluster centroids of the previous and current POS is calculated with the following formula [Zou+21]:

$$C_t = \frac{1}{|\text{POS}t|} \sum x_t \in \text{POS}t x_t \tag{4.5}$$

$$xt + 1 = x_t + k|C_t - C_{t-1}| + e_t \tag{4.6}$$

Where C_t and C_{t-1} represent the cluster centroids for the time steps t and $t-1$, respectively. x_{t+1} represents the predicted individual at $t+1$ time step. This process is shown in Fig. 4.4.

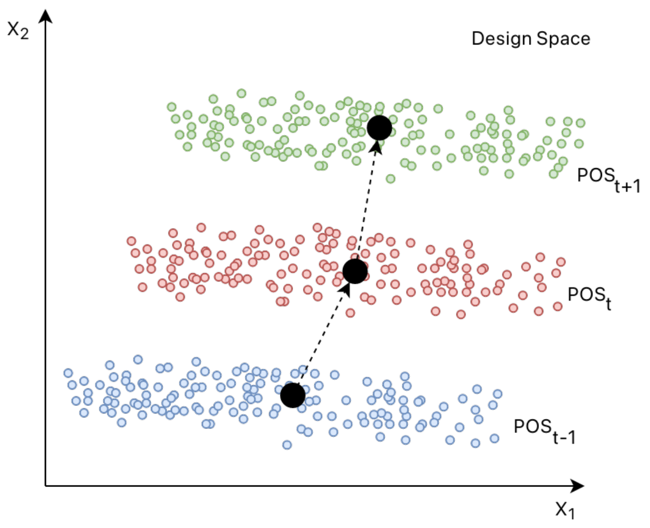


Figure 4.4: Illustration of the Center-Based prediction method for different time steps t. Own illustration

4.4 Boundary Detection Mechanism

The primary objective of this thesis is to develop an algorithm that exhibits comparable performance to the KGB-DMOEA algorithm while mitigating the computational overhead. This is to be achieved by using simpler linear methods as devised in the sections 4.3.1 and 4.3.2. However, these models are struggles when dealing with boundary situations. When a population reaches a boundary during the optimization process, simple linear models disregard the direction of the population movement, which can result in a population that falls directly on the boundaries of a problem. In dynamic context, this is especially problematic in situations where the population "overshoots" the current true POS, as the SMOEA employed in DMOEA has then to correct for the overshooting. Especially in fast, high-dimensional environments, this can lead to a complete loss of the tracking ability of the DMOEA

To preserve performance, it is, therefore, crucial to mitigate the risks of false predictions, particularly in border regions. Therefore, a boundary detection mechanism is introduced into the proposed algorithm, aiming to detect such situations and exclusively use KGB-DMOEA for such cases. The rationale behind this decision is that KGB-DMOEA, as a nonlinear prediction model, should be better capable of handling situations at the extremes of the decision space.

The boundary detection mechanism is implemented by adding an additional parameter θ, that is used as a threshold towards the boundaries of a given problem. Once the centroid of a POS_t crosses this threshold, the boundary detection is triggered and KGB-DMOEA is always used. Fig. 4.5 illustrates this concept and Alg. 9 shows the pseudo code of the boundary detection mechanism.

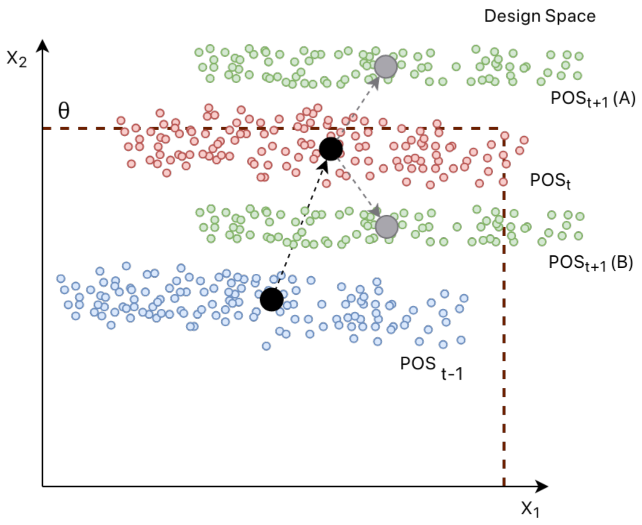


Figure 4.5: Illustration of the devised boundary detection method. Once the centroid of a population crosses the threshold, θ KGB-DMOEA is used. Own illustration.

Algorithm 9 Boundary Detection

```
Input: $POS_t$; $x_l$; $x_u$;
Output: Boolean Boundary Situation
1: $C_t = 1$ / len(POS) * $\sum$(POS, axis=0);
2: dist to $x_l \leftarrow C_t$ - $x_l$;
3: dist to $x_u \leftarrow x_u$ - $C_t$;
4: if any(dist to $x_l < \theta$) or any(dist-to-xu $< \theta$) then
5:     return True;
6: else
7:     return False;
8: end if
```

4.5 State-Action Table

The State-Action table is a crucial component of the proposed algorithm. It is used to store the information about the State-Action pairs $Q(S, A)$ of a Q-Learning algorithm. As described in section 4.2, the states used in the Q-Learning algorithm are defined by the severity of change an agent faces when change is detected in the environment. Table 4.1 shows the states and actions used in for this thesis.

State	Action		
	$a_1(CBP)$	$a_2(KBP)$	$a_3(KGB)$
$s_1(\delta < \bar{\delta})$	$Q(s_1, a_1)$	$Q(s_1, a_2)$	$Q(s_1, a_3)$
$s_2(\bar{\delta} \leq \delta \leq \bar{\delta} * 1.5)$	$Q(s_2, a_1)$	$Q(s_2, a_2)$	$Q(s_2, a_3)$
$s_3(\delta > \bar{\delta} * 1.5)$	$Q(s_3, a_1)$	$Q(s_3, a_2)$	$Q(s_3, a_3)$

Table 4.1: State-Action Table (Q-Table).

When environmental change is detected the RL agent chooses an action a_t based on the current state s_t and the Q-Table. The Q-Table is updated after each episode, as described in section 2.4.3. The Q-Table is initialized with zeros for every algorithm run and updated after each step using the Q-Learning update formula described in 5.

4.6 Action Selection Policy

Another main component of the predefined algorithm is the decaying epsilon greedy action selection strategy. Unlike the classical epsilon greedy, the probability of the agent to perform an action that is not optimal according to Q value in the current state is reduced over time. Thus, the goal is to determine the rewards of different prediction methods at the beginning of an algorithm run, and in the later course of an algorithm run to increasingly use prediction methods that have proven successful in the past. Essentially, the probability of an agent to pick an action that is not optimal for a given state decreases exponentially. Algorithm Alg. 10 depicts the adapted decaying epsilon greedy policy, where.

Algorithm 10 Decaying ϵ-greedy Action Selection

Input: The current Q-Table $Q(S, A)$; action selection probability ϵ; epsilon decay ϵ_d;

Output: An action a_t

```
1: $n \leftarrow random.uniform(0, 1)$;
2: if $n < \epsilon$ then;
3:     $a_t \leftarrow randomactiona \in A$;
4: else
5:     $a_t \leftarrow maxQ(S, .)$;
6: end if
7: $\epsilon \leftarrow \epsilon * \epsilon_d$;
```

4.7 Reward Function

Reinforcement Learning agents learn via feedback, which is given by the environment by means of an reward or punishment. This reward is a scalar value that allows the agent to learn which action returns the most reward in relation to a state. Thus, the reward function is key to controlling the behavior of an RL agent and is mainly responsible for the behavior of an agent.

Since this thesis aims to obtain a similar performance as with the KGB-DMOEA algorithm but to decide which prediction method to choose depending on the time, two main factors are defined to determine the reward for the agent. One is the prediction quality, which can be measured with any measure, and the wall clock time, which measures how long the execution of the prediction method took.

For the purpose of this thesis following reward function is used:

$$r_t = \frac{\rho_t}{\rho_{max}} - \frac{\mu_t}{\mu_{max}} \tag{4.7}$$

Where ρ_t is the prediction quality of the prediction method used in the current step, ρ_{max} is the maximum prediction quality of all prediction methods used in the current algorithm run, μ_t is the wall clock time of the prediction method used in the current step and μ_{max} being the maximum wall clock time of all prediction methods used in the current algorithm run. With this, the reward function is normalized to the best-determined prediction quality of a given run and the worst-determined wall clock time of a given run. Higher prediction quality is rewarded, which serves as a base reward, and the wall clock time acts as a detrimental factor. As the metric for prediction quality Hypervolume is used as introduced in 5.2.4

4.8 Hyperparameters

Another important point that should not be neglected are the used hyperparameters of the proposed algorithm, which can have an influence on the algorithms' performance depending on the optimized problem. As the proposed algorithm combines RL with the KGB-DMOEA, the hyperparameters of the proposed algorithm employ the same hyperparameters. Additionally, the hyperparameter ϵ_d for the epsilon greedy action selection decay as well as the hyperparameter ρ as the boundary threshold are added. Tab. 4.2 lists the hyperparameters of the proposed algorithm.

Hyperparameter	**Description**
Population size, N	Number of individuals in a population.
Population sample size η,	Percentage of population that will be sampled to detect change.
Cluster Size, N_c	Number of Knowledge Clusters for KGB-DMOEA
Boundary Threshold ρ,	Minimum distance a given POS_t is required to have towards the decision space bounds.
Learning Rate α,	Learning Rate of a RL agent. Higher values result in better performance but slower learning convergence.
Discount Factor γ,	Weight factor of future rewards.
Epsilon ϵ,	Probability that a RL agent does not choose $maxQ_t(a)$. Manages the exploration vs. exploitation trade-off.
Change Detection Threshold, δ,	Delta a sampled population objective values from do have to pass to detect change.

Table 4.2: Parameters of the proposed algorithm.

5. Experimental Study

This chapter presents the results of the experimental study conducted for this thesis. Section 5.1 first introduces the benchmark problems used in the experiments. Subsequently, the metrics used to determine performance are introduced in Section 5.2. The implementation of the tested algorithms and any necessary adaptations for conducting this work is described in Section 5.3. In Section 5.4 the implemented algorithm is validated. The experimental setup, with experimental results and a discussion, are outlined in Section 5.5.

5.1 Benchmark Problems

Artificial benchmark problems have played a crucial role in evaluating the effectiveness of DMOAs for solving DMOPs. These benchmark problems allow for the analysis and understanding of the strengths and limitations of a DMOA, enabling modifications to be made for performance improvements. Additionally, using benchmark problems enables the comparison of different DMOAs, facilitating the identification of the most suitable DMOA for a given problem class.

The introduction of benchmark problems for DMOPs is a relatively recent development and a research area in the field of DMOP in and of itself. The first benchmark problems for DMOPs were introduced in 2001 with the works of Jin and Sendhoff [JS04]; the authors proposed the first benchmark problems by dynamically varying the weights for multi-objective functions. Since then, several benchmark problems have been proposed [JY17] [Jia+18b], gradually increasing in complexity and realism.

Jiang et al. [Jia+18b], 2018 introduced the DF problem suite, a widely used benchmark suite containing 14 different dynamic multi-objective problems. Thereof are nine bi-objective and five tri-objective problems. The differences between each problem are summarized in Table 5.1, as described by the original authors Jiang et al. [Jia+18b]. With the introduction of this problem suite, the authors tackled the lack of a standardized benchmark suite for DMOPs that is representative of real-world scenarios, represented in irregular Pareto fronts, disconnectivity, and time-dependent POF shapes [Jia+18b]. For this reason, the DF problem suite is used in this thesis.

To better understand the problem suite, one can look at Fig. 5.1, which depicts the DF4 problem. DF4 has dynamics both on the POF and POF, with the length and curvature of the POF and the POS changing over time.

Problem	Objective	Dynamics	Remarks
DF1	2	mixed convexity-concavity, location of optima	dynamic POF and POS
DF2	2	position-related variable switch, location of optima	static convex POF, dynamic POS, severe diversity loss
DF3	2	variable-linkage, mixed convexity-concavity, location of optima	dynamic POF and POS
DF4	2	variable-linkage, POF range, bounds of POS	dynamic POF and POS
DF5	2	number of knee regions, local of optima	dynamic POF and POS
DF6	2	mixed convexity-concavity, multimodality, location of optima	dynamic POF and POS
DF7	2	POF range, location of optima	convex POF, static POS centroid, dynamic POF and POS
DF8	2	mixed convexity-concavity, solutions distribution, location of optima	static POS centroid, dynamic POF and POS, variable-linkage
DF9	2	number of disconnected POF segments, location of optima	dynamic POS and POF, variable-linkage
DF10	3	mixed convexity-concavity, location of optima	dynamic POS and POF, variable-linkage
DF11	3	size of POF region, POF range, location of optima	dynamic POS and POF, concave POF, variable-linkage
DF12	3	number of POF holes, location of optima	dynamic POS, static concave POF, variable-linkage
DF13	3	number of disconnected POF segments, location of optima	dynamic POS and POF, POF can be continuous convex, concave, or several disconnected segments
DF14	3	degenerate POF, number of knee regions, location of optima	dynamic POS and POF variable-linkage

Table 5.1: Summary of the DF problem suite, by Jiang et al. [Jia+18b].

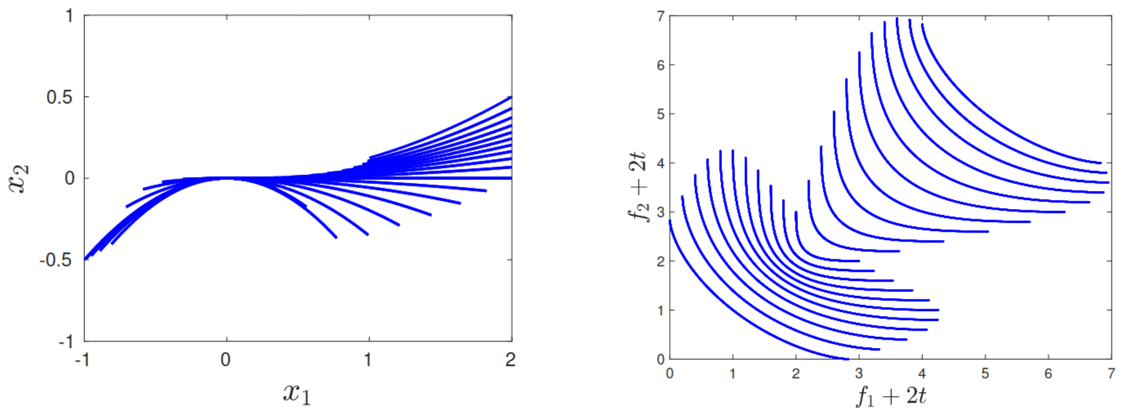


Figure 5.1: Depiction of the POS and POF for the DF4 problem. As t increases, the length and curvature of both POS and POF changes, by Jiang et al. [Jia+18b].

5.2 Performance Metrics

In order to measure the performance of DMOEAs in solving benchmark problems, there is a need to define a set of performance metrics. This subsection will discuss the most common performance metrics used in the literature.

5.2.1 Generational Distance

The Generational Distance (GD) is a commonly used performance indicator that measures the distance from an *approximated* POF_t to the true POF_t at the time t.

For a given set of calculated objective value vector $A = \{a_1, a_2, \ldots, a_{|A|}\}$ and a set of reference points (true POF_t), $Z = \{z_1, z_2, \ldots, z_{|Z|}\}$. It can be formally defined as follows [Liu+19]:

$$GD(A) = \frac{1}{|A|}\left(\sum_{i=1}^{|A|} d_i^p\right)^{1/p} \tag{5.1}$$

Where d_i represents the euclidean distance ($p = 2$) from a_i to its nearest reference point in Z. Basically, this results in the average distance from any point A to the closest point in the POF_t. Fig. 5.2 (a) depicts the working principle of the GD between the true POF_t and the approximated POF_t.

5.2.2 Inverted Generational Distance

The Inverted Generational Distance (IGD) is the inverse of the GD. It measures the distance from any point in Z to the closest point in A. It is defined as follows:

$$IGD(A) = \left(\frac{1}{|Z|}\sum_{i=1}^{|Z|} \hat{d}_i^p\right)^{1/p} \tag{5.2}$$

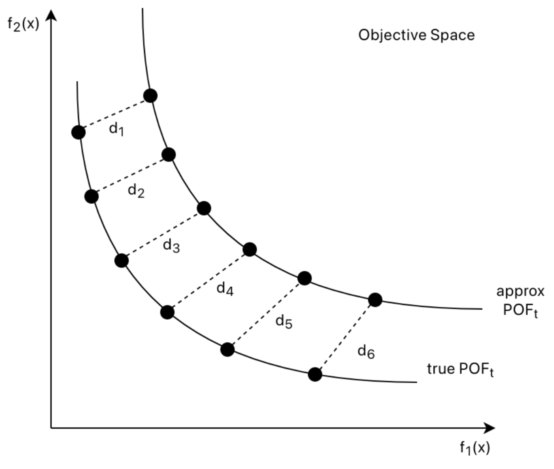
Where $\hat{d}_i$ represents the euclidean distance ($p = 2$) from z_i to its nearest reference point in A. Fig. 5.2 depicts the working principle of the IGD, any point z_i to its nearest reference point of the true POF_t.

5.2.3 Mean Inverted Generational Distance

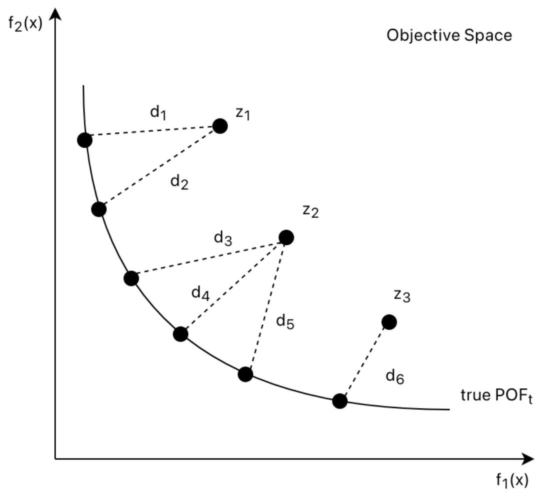
Taking the average of the IGD over an optimization run results in the Mean Inverted Generational Distance (MIGD). With this all historical environments are considered. It can be defined as follows [Ish+15]:

$$MIGD(POF_t^*, POF_t) = \frac{1}{|T|}\sum_{t\in T} IGD(POF_t^*, POF_t) \tag{5.3}$$

With POF_t^* being the true POF of a given DMOPat time t and POF_t being the approximated POF at time t. With T being the set of discrete-time steps in a given run and $|T|$ being the cardinality of T. Therefore, $MIGD$ is nothing else than the average of the IGD over all time steps, concluding that a lower value for $MIGD$ indicates better performance of a given DMOEA in terms of convergence and diversity.



(a) fig 1



(b) fig 2

Figure 5.2: Illustration of the GD and IGD. Own illustration .

5.2.4 Hypervolume

So far, all performance indicators have been based on the assumption that the true POF_t is known at time t. However, in real-world applications, this is generally not the case. Therefore, the Hypervolume (HV) was introduced as a performance metric that is independent of the true POF_t. It is defined as follows [GFP21]:

$$HV(POF_t) = VOL(\bigcup_{x \in POF_t} [f_1(x), z_1^t] \times ... \times [f_m(x), z_m^t]) \tag{5.4}$$

With VOL being the Lebesgue measure and $z_i^t(i = 1, 2, ...m)$ referring to a reference point used for the HV computation. In other words, the HV is the volume of the area dominated by the POF_t and a given reference point. Fig. 5.3 depicts the working principle of the HV. The area A which is delimited by the points $z_1(a) - z_3(a)$ constitutes a smaller HV than the area B delimited by the points $z_1(b) - z_3(b)$.

5.2.5 Mean Hypervolume

The Mean Hypervolume (MHV) is the average HV over an optimization run and is defined as follows [GFP21]:

$$MHV(POF) = \frac{1}{|T|} \sum_{t \in T} HV(POF_t) \tag{5.5}$$

With $|T|$ being the set of discrete-time steps in a given run and $|T|$ being the cardinality of T. Therefore, MHV can be described as the average of the HV over all time steps, concluding that a higher value for MHV indicates better performance of a given DMOEA in terms of convergence and diversity.

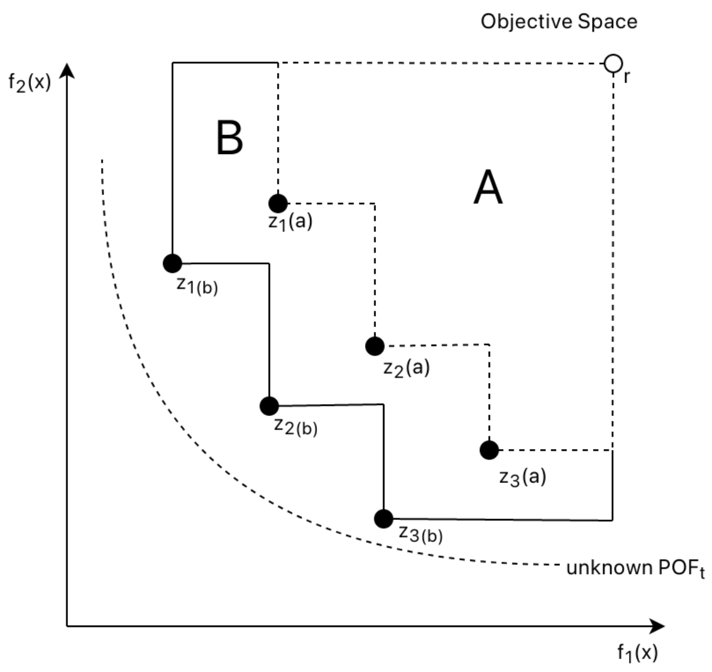


Figure 5.3: Illustration of the HV metric. Own illustration.

5.3 Implementation

Both the KGB-DMOEA and the proposed algorithm were implemented in Python [RD06], a versatile programming language with a large scientific community and libraries that facilitate the implementation process. An essential library used for the implementation and testing of the algorithms is pymoo (Multi-Objective Optimization in Python) [BD20], which was developed by Blank and Deb [BD20] and provided a standard interface for implementing multi-objective evolutionary algorithms. For this thesis, this library was mainly used to access the DF benchmark suite, the performance indicators described in section 5.2, and the NSGA-II as a SMOEA for both the KGB-DMOEA and the proposed algorithm.

Another library used for the implementation was the sci-kit learn library [Kra16], which provides a wide range of machine learning algorithms. The implementation of the NBC is based on the naive Bayes classifier from this library.

The KGB-DMOEA implementation was based on the algorithm as described by Ye et al. [Ye+22] in their original paper, with the only difference being the use of the NSGA-II as the SMOEA for optimization once change is detected. In the original paper, the authors used the Multi-Objective Evolutionary Algorithm Based on Decomposition (MOEA/D) [ZL07] instead. This change was made as the NSGA-II was readily available in the pymoo library while MOEA/D was not, however it has no impact on the validity of the results as both algorithms used the same SMOEA.

Furthermore, it is to be noted that the implementation of the Knee-Based prediction method as well as the Center-Based prediction method are based on the descriptions provided by [Zou+21].

Table 5.2 gives an overview of the software and hardware used for the implementation and subsequent experiments. To ensure maximum reproducibility, all experiments

Software / Hardware	**Version**
OS	Ubuntu 22.04.1
Kernel	5.15.0-56
CPU	Intel i7-4790K (8) @ 4.400GHz
RAM	24.00Gb
Python	3.10.6
pymoo	0.6.0
Scikit-learn	1.2.0

Table 5.2: Software and Hardware used for the implementation, testing and experiments.

for this thesis were conducted on a single core of a desktop computer without parallel processing under the same conditions with no other system load.

5.4 Validation

Before the proposed algorithm can be used for an experimental study, the implementation must be validated. For this, the DF1 problem is used. With DF1, the POS moves upwards on the x_2-axis as t increases, as depicted in Fig. 5.4. This benchmark problem is tested with two decision variables $x = [x1, x2]$, and with an environmental change severity of $n_t = 10$, meaning a fast environment, and a change severity τ_t of 5, meaning low environmental change is used as a minimal example, to ensure that the algorithm is working as intended.

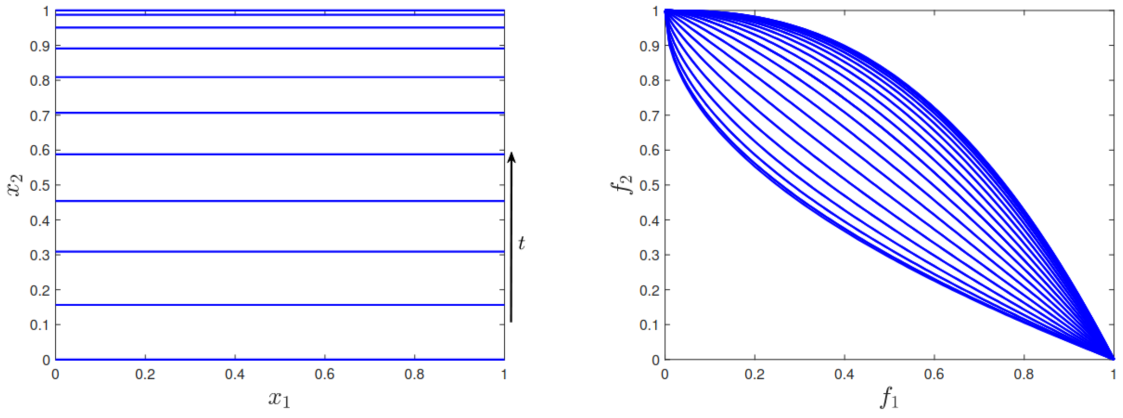
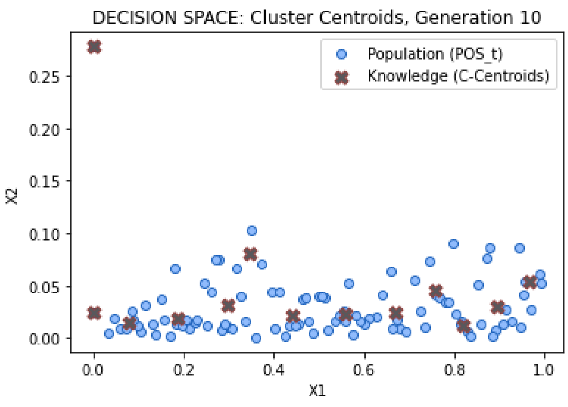
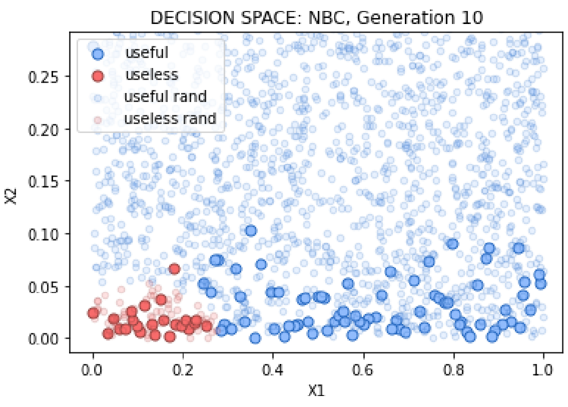


Figure 5.4: Depiction of the POS and POF for the DF1 problem, by Jiang et al. [Jia+18b].

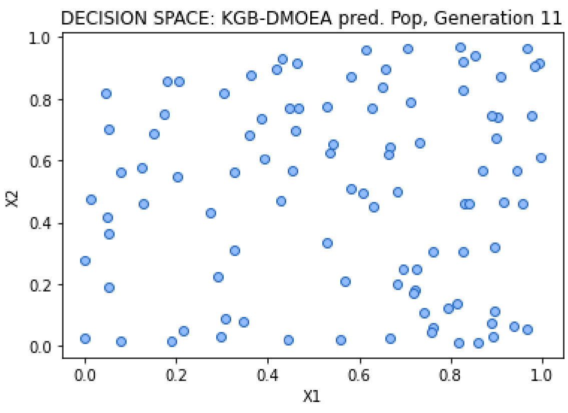
In Fig. 5.5 the evolution of the proposed algorithm can be seen. Fig (a) 1. is showing the proposed algorithm after 10 generations. The algorithm correctly detected change and one can see the current aproximated POS_t as blue marked points. As the true POS for the DF1 problem is expected to be at the lower end of the decision space, the algorithm correctly approximates the POS. Furthermore, it can be seen in Fig (a) 1. that the KGB-DMOEA successfully determined the knowledge clusters of the current problem state (red marked crosses). With these historical knowledge clusters KGB-DMOEA is capable of retaining knowledge of a previous POS. Fig (b) 2., one can see the NBC after training. By testing the historical knowledge cluster, in the new environment, and subsequently generating a lot of random numbers the NBC could succesfully discern from useless and usefull solutions, and use this knowledge to predict a new population as can be seen in Fig 3 (c), where the whole decision space is sampled. It is to be noted that the lower left boundaries of the DF1 decision space are not sampled as much, as the NBC successfully predicted that this region is most likely not the POS of the next change interation, therefore solutions are sampled more on the top right of the decision space. Fig (d) 4. shows the second change event, where the POS now moved to towards the top of the DF1 decision space. In this case, the KGB-DMOEA correctly predicted the next POS and the knowledge clusters are now distributed over two areas in the decision space. Therefore, one can say that the KGB-DMOEA is successfully retaining knowledge of the previous POS.



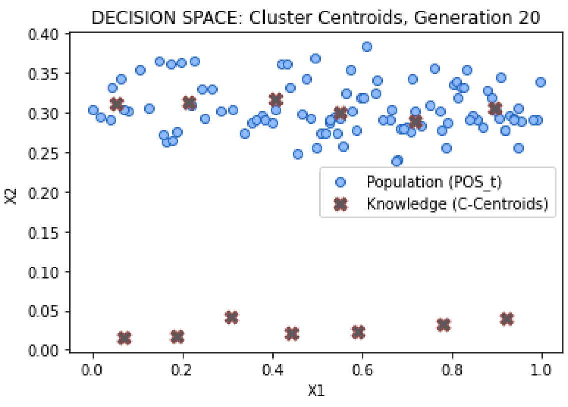
(a) Fig 1



(b) Fig 2



(c) Fig 3

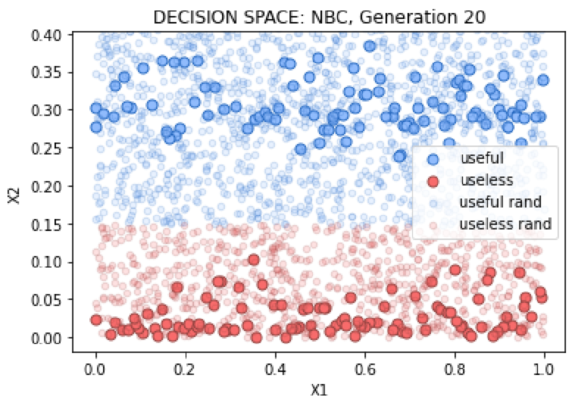


(d) Fig 4

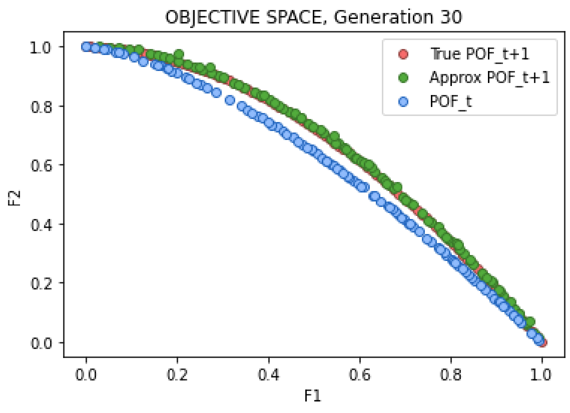
Figure 5.5: Evolution of the optimization process of the proposed algorithm (1).

In the second plot series in Fig. 5.6, the optimization run is continued. Here Fig 1 (b) one can see that the true *POF* is successfully predicted. Furthermore, in Fig. 3 (c), it can be clearly seen how the KGB-DMOEA algorithm works after a certain amount of change events. The algorithm successfully retaines knowledge over the whole decision space, as can bee seen with the knowledge clusters. This is useful information for nonlinear change in the problem, as KGB-DMOEA can use the fix points to guide the search in new environments. Fig 4 (d), shows the Center-Based prediction as an example for linear prediction models that is used. Ic can be seen how based on the previous POSs the moving direction of the decision space is successfully predicted.

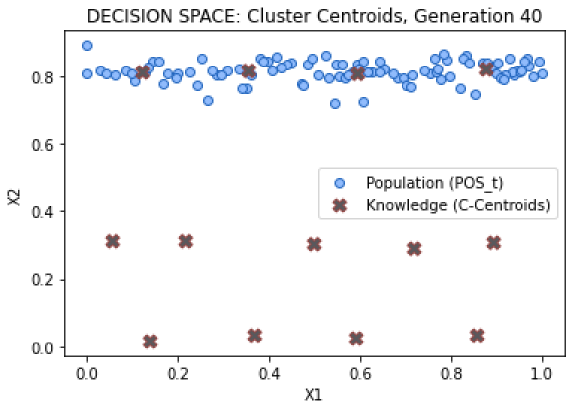
In summary, the evaluation shows that the proposed algorithm is correctly implemented and that the different components of the algorithm work as expected. With the proposed algorithm, can be used in an experimental study to compare the performance of the KGB-DMOEA algorithm with other the propsed algorithm.



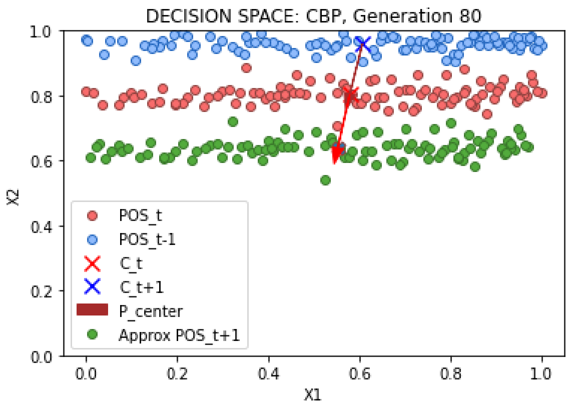
(a) Fig 1



(b) Fig 2



(c) Fig 3



(d) Fig 4

Figure 5.6: Evolution of the optimization process of the proposed algorithm (2)

5.5 Experimental Setup

In the following section the experimental setup is described and the experimental results are presented and discussed. The experimental setup is divided into two parts. The first part briefly describes the settings of the DF1 benchmark suite used in the experiments, while the second part describes the hyperparameter settings of the the proposed algorithm as well as KGB-DMOEA. Thereafter, the results are presented and subsequently discussed.

5.5.1 Benchmark Suite Parameter Settings

In terms of benchmark suite settings there are mainly the following two aspects to consider. The first aspect is the number of environmental changes that a given optimization problem will undergo, while the second aspect is the severity these changes have. For the purpose of this thesis the recommended benchmark suite settings as devised by Jiang et al. [Jia+18b] are used. These are as follows:

- Number of decision variables: 10
- Frequency of change / Generations between change (τ_t): 10 (fast changing environments), 30 (slow changing environments).
- Severity of change (n_t): 1 (small changes), 3 (medium changes), 5 (severe changes).
- Number of changes: 30
- Stopping criterion: $100(30\tau_t + 50)$ fitness evaluations
- Number of independent runs: 20

5.5.2 Algorithm Parameter Settings

The following subsection describes the hyperparameter settings used for the experimental study:

- Population size, N: 100
- Population sample size η: 0.2
- Cluster Size, N_c: 13
- Boundary Threshold ρ: 0.1
- learning rate α: 0.9
- discount factor γ: 0.6
- epsilon action selection ϵ: 1
- epsilon decay ϵ_d: 0.90
- boundary threshold ρ: 0.05

These hyperparameters are used for the proposed algorithm, to be compared with the KGB-DMOEA. Thereby, the cluster size N_c is the only shared hyperparameter between the two compared algorithms and is set to 13 for both KGB-DMOEA and the proposed algorithm. The hyperparameters value for the RL part of the algorithm are as described by Zou et al. [Zou+21].

5.5.3 Results

In the following section the experimental results of the study are presented. To obtain these results both the KGB-DMOEA and the proposed algorithm were tested on the full DF1 benchmark suite for each 20 individual runs, to mitigate the potential of randomness influencing the results. Each problem of the suite was run for 30 change events, with a change frequency of 10 and 30, and a change severity of 1, 3 and 5.

As an example Fig. 5.8 depicts the experimental results for the DF1 suite with a change frequency of 30, and a change severity of 3, meaning medium level change events. In the first graph in the top left corner the average runtime (over 20 runs), for each DF problem can be seen. In purple one can see the KGB-DMOEA algorithm and in orange the proposed algorithm. It can be seen that the proposed algorithm is faster than the KGB-DMOEA algorithm. This is due to the fact that the proposed algorithm has multiple faster prediction methods available with the knee-based prediction and the center based prediction. On the other hand KGB-DMOEA is fixed to a single prediction method and on top of that has to perform the knowledge-reconstruction examination on each change event. Taking a look at the second graph in the top row, one can see the average IGD of both algorithms on log scale. The log scale was used on this graph, to represent different IGD scales on the same graph. Taking a look it can be seen that the performs better on average compared to the proposed algorithm, but the proposed algorithm yield comparable results. This effect is also represented on the HV which can be seen in the third graph on the top right. It is important to note that for the HV, higher values translate to better performance, and it can again be seen the KGB-DMOEA has a slight performance edge. Taking a look at the bottom graphs of the figure it is evident where the proposed algorithm lacks. Compared to KGB-DMOEA the proposed algorithm has a higher standard deviation for all evaluated metrics, namely time, IGD and HV. This is due to the fact that the proposed algorithm has multiple prediction methods available, which can lead to more diverse results from change event to change event and therefore also result in higher deviation.

Similar results can be observed when examining the performance in faster environments, as shown in Fig. 5.8. Again as expected, the proposed algorithm performs considerably faster as KGB-DMOEA, while showing comparable performances. The main difference that can be observed between KGB-DMOEA and the proposed algorithm in fast environments is the higher runtime standard deviation of the proposed algorithm. Again this can be mainly attributed to the different prediction methods used, mainly when the proposed algorithm uses the integrated KGB-DMOEA often.

In table 5.3 the relative results of the proposed algorithm to the KGB-DMOEA are presented. For this the performance results of KGB-DMOEA were taken as the base and the performance metrics of the proposed algorithm therefore depict

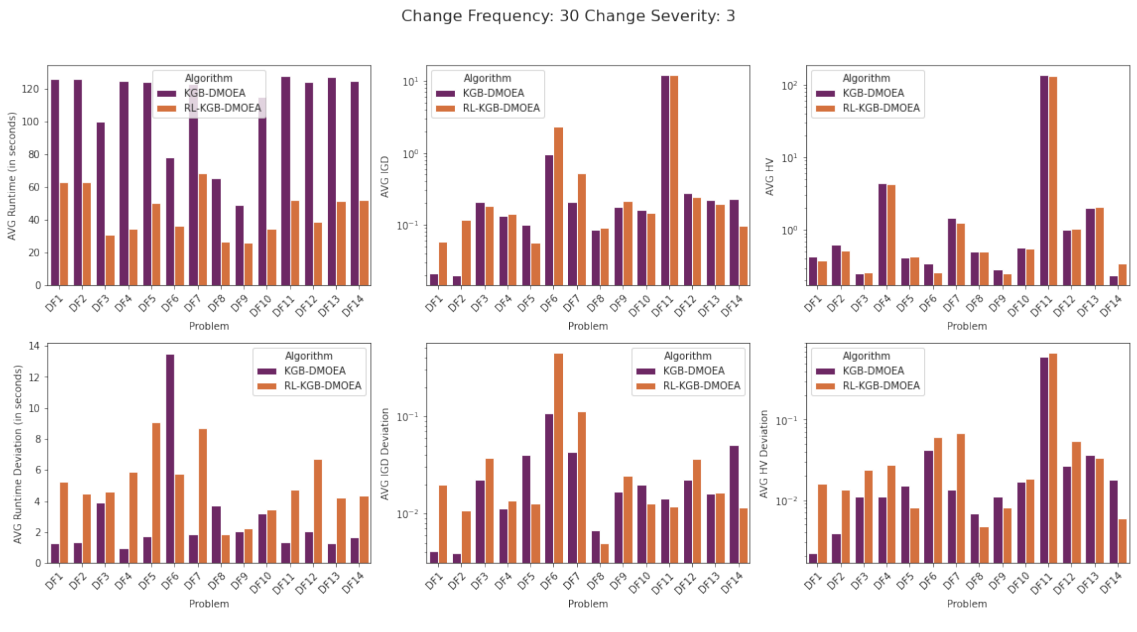


Figure 5.7: Results of the KGB-DMOEA and RL-KGB-DMOEA on the DF1 Benchmark Suite, Change Frequency 30, Change Severity 3

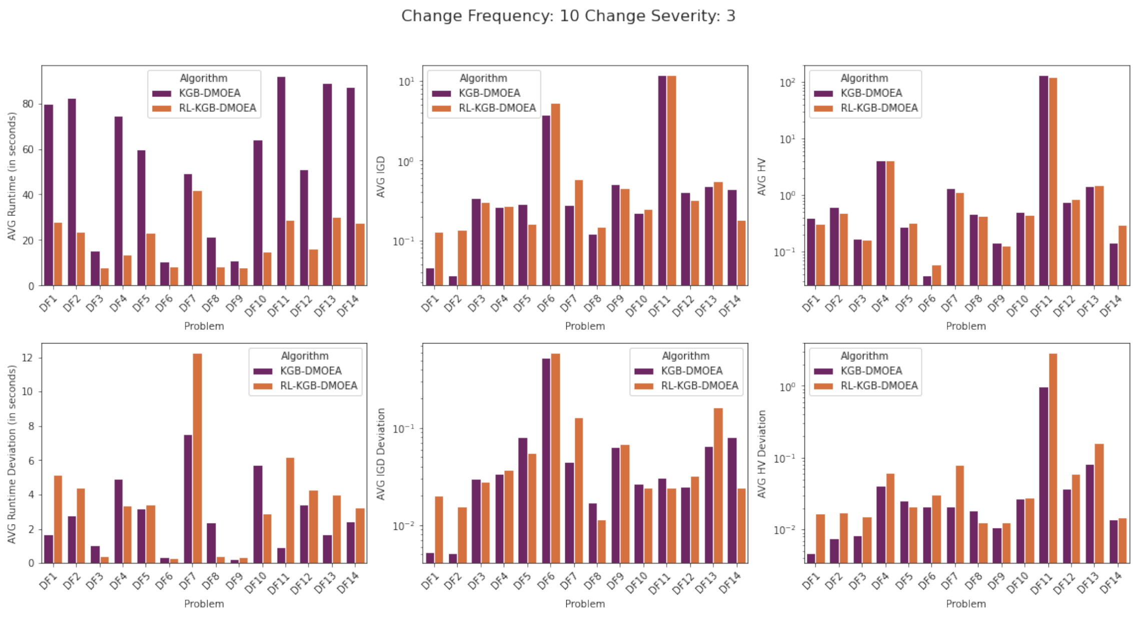


Figure 5.8: Results of the KGB-DMOEA and RL-KGB-DMOEA on the DF1 Benchmark Suite, Change Frequency 10, Change Severity 3

Problem	avg_time	avg_time_dev	avg_igd	avg_igd_dev	avg_hv	avg_hv_dev
DF1	0.3529	3.0815	2.7667	3.8491	0.7896	3.5319
DF2	0.2879	1.5892	3.7694	2.9808	0.7797	2.2338
DF3	0.5159	0.3916	0.8726	0.9145	0.967	1.7882
DF4	0.1847	0.6885	1.0152	1.0789	0.9879	1.5183
DF5	0.3897	1.0647	0.5606	0.6934	1.1905	0.8054
DF6	0.7895	0.7975	1.409	1.1382	1.6074	1.4597
DF7	0.8485	1.6208	2.138	2.8795	0.8536	3.8252
DF8	0.3894	0.1672	1.2038	0.6667	0.9208	0.6957
DF9	0.7133	1.2719	0.9144	1.0728	0.8953	1.1852
DF10	0.2318	0.5033	1.1251	0.9071	0.8841	1.0639
DF11	0.312	6.6963	1.0036	0.785	0.9418	2.9518
DF12	0.3161	1.2453	0.7994	1.3	1.1716	1.6076
DF13	0.3377	2.3498	1.1502	2.4599	1.0359	1.9496
DF14	0.3144	1.3513	0.4078	0.2988	2.1164	1.0576

Table 5.3: Relative Results of the RL-KGB-DMOEA on the DF Benchmark Suite, Change Frequency 10, Change Severity 3. Lower values indicate better performance, except for "avg_hv", where higher values indicate better performance.

the percentage increase or decreases compared to KGB-DMOEA. Taking a look at the column "avg_time" one can see that the proposed algorithm is faster than KGB-DMOEA in all cases. The speed increase range from up to 50% to 80% as can bee seen with the DF10 problem for example. Taking a look at the "avg_igd" column one can see that the performance losses are comparatively low in most cases, with significant performance losses only in DF1, DF2, DF6 and DF8 where the proposed algorithm showed up more than 20% worse performance. However for most benchmark problems the results show that the proposed algorithm is comparable in performance to KGB-DMOEA and sometimes even performs better with faster wall clock time as seen in DF5. Here the proposed algorithm was able to result in around 40% increase in performance while also exhibiting a 60% lower

Problem	**avg_time**	**avg_time_dev**	**avg_igd**	**avg_igd_dev**	**avg_hv**	**avg_hv_dev**
DF1	0.4989	4.1235	2.8381	4.7805	0.8991	7.3182
DF2	0.5011	3.3619	5.9095	2.7949	0.8096	3.4359
DF3	0.308	1.1715	0.8936	1.7182	1.0465	2.1429
DF4	0.2794	6.2241	1.0701	1.1842	0.994	2.4727
DF5	0.4066	5.2989	0.5675	0.3136	1.0418	0.5294
DF6	0.4683	0.427	2.4555	4.2081	0.7545	1.4606
DF7	0.5591	4.6427	2.5217	2.6449	0.857	4.9124
DF8	0.4141	0.5054	1.0728	0.7463	0.9879	0.6812
DF9	0.5344	1.0897	1.2084	1.4464	0.862	0.7364
DF10	0.3022	1.0885	0.9062	0.6382	0.962	1.0765
DF11	0.409	3.4889	0.9988	0.8322	0.9838	1.1285
DF12	0.3121	3.3079	0.8785	1.6425	1.0277	2.0524
DF13	0.405	3.3619	0.8959	1.0185	1.0381	0.9146
DF14	0.4151	2.6123	0.4238	0.2259	1.4557	0.3315

Table 5.4: Relative Results of the RL-KGB-DMOEA on the DF Benchmark Suite, Change Frequency 30, Change Severity 3. Lower values indicate better performance, except for "avg_hv", where higher values indicate better performance.

6. Future Work

In future work, the limitations of the approach used in this thesis for the proposed algorithm will be further investigated.

One major limitation is the lack of a parameter sensitivity study. A sensitivity study would provide a better understanding of how the proposed algorithm is affected by different hyperparameters, and would also help to confirm its viability as an alternative to the KGB-DMOEA method.

Another limitation is the limited scope of benchmark problems on which the algorithm has been tested. Testing the performance on a wider range of benchmark problems would provide a more comprehensive evaluation of the algorithm's capabilities.

Additionally, the proposed algorithm was only tested in environments without stochastic effects. Therefore, it would be beneficial to test the proposed algorithm under stochastic conditions to see if its performance potential holds in comparison to the KGB-DMOEA.

Another potential avenue for future research is the use of model-based reinforcement learning, where the agent also learns the behavior of the given optimization problem directly. This can help improve the algorithm's performance by focusing on the unique characteristics of the optimization problem.

Finally, future work could also include investigating other prediction method combinations and reward function designs, as well as exploring the potential for real-world applications of the algorithm. Further research could also investigate the use of more advanced reinforcement learning algorithms such as deep Q-learning [Gu+16], which would allow for more complex state and action spaces.

7. Conclusion

DMOPs are optimization problems that involve multiple time-dependent objectives with significant practical relevance. To handle dynamics, DMOEA have been proposed, using dynamic processing techniques to handle environmental changes.

Knowledge Guided Bayesian Classification is a state-of-the-art DMOEA that extracts knowledge from historical environments and uses naive Bayesian classification as a prediction method to predict promising solutions to adapt to environmental change. This method effectively solves DMOPs, outperforming most previous DMOEA in benchmark problems.

However, using this single prediction model is a significant limitation of KGB-DMOEA as it may not be suitable for all encountered changes in the problem environment. Furthermore, the processing of historical information needed to train the prediction model may slow down the optimization process, which can be an unnecessary overhead in cases with linear changes in the problem environment.

In this thesis, an improved version of KGB-DMOEA was proposed to reduce the running time of the KGB-DMOEA without sacrificing too much performance. For this purpose, a Q-Learning agent was used to select the appropriate prediction mechanism for environmental change based on the experience obtained from previous environments. With this approach, the agent could learn which prediction mechanism would render the best results in the current environment regarding prediction quality *and* running time.

For this, the needed fundamentals, including the general optimization problem, its extension to a multi-objective-, and a dynamically changing multi-objective definition, were first outlined in chapter 2. Furthermore, we introduced the Knowledge Guided Bayesian Classification Algorithm and the basics of Reinforcement Learning within this chapter.

In chapter 3, we discussed relevant related work of this research field and highlighted the research gap this thesis aimed to fill. In chapter 4, we proposed the improved algorithm, which uses a reinforcement learning framework to dynamically select appropriate response strategies based on both performance and runtime.

One essential contribution of this work was the introduction of a boundary detection mechanism to detect when the optimal solutions of a given problem are located at the boundaries of the problem's search space. This mechanism was used to always select KGB-DMOEA at the problem boundaries, assuming that problem behavior on the boundaries is less predictable and, therefore, always worth the computational overhead of extracting knowledge from historical environments. Another essential contribution was the introduction of decaying epsilon-greedy exploration to the Q-Learning agent, which was used to balance exploration and exploitation in the agent's decision-making process. This was done to ensure that the agent would widely explore the benefits of prediction methods for a given environment at the start of an optimization run but would exploit the knowledge gained from previous environments as the optimization process progressed.

In chapter 5, we introduced the DF benchmark problems as a tool to evaluate the performance of dynamic multi-objective evolutionary algorithms and presented commonly-used performance metrics. We then described the implementation of the algorithm and conducted a performance study on the DF benchmark suite. With extensive experiments in the commonly used DF benchmark suite, it was shown that the proposed algorithm was able to perform at comparable levels to the KGB-DMOEA while reducing the running time of the KGB-DMOEA by up to 60% in some cases. The experiments conducted verify the efficacy of the proposed algorithm.

Although the proposed algorithm was able to reduce the running time of the KGB-DMOEA without sacrificing too much performance, there is still room for further research in this area. For one, the proposed algorithm was only tested on the DF benchmark suite, a synthetic environment without stochastic effects. It would be interesting to see how the proposed algorithm would perform in a real-world scenario with a stochastic effect. Furthermore, the proposed agent was only trained using Q-Learning, which is a relatively simple reinforcement learning algorithm. It would be interesting to see how the proposed algorithm would perform if trained with more advanced reinforcement learning algorithms.

Bibliography

[1] David Ackley. *A Connectionist Machine for Genetic Hillclimbing.* Springer Science & Business Media, 6th Dec. 2012. 268 pp. ISBN: 978-1-4613-1997-9. Google Books: sx_VBwAAQBAJ.

[2] Entisar S. Alkayal, Nicholas R. Jennings and Maysoon F. Abulkhair. “Efficient Task Scheduling Multi-Objective Particle Swarm Optimization in Cloud Computing”. In: *2016 IEEE 41st Conference on Local Computer Networks Workshops (LCN Workshops).* 2016 IEEE 41st Conference on Local Computer Networks Workshops (LCN Workshops). Nov. 2016, pp. 17–24. DOI: 10.1109/LCN.2016.024.

[3] P. Amato and M. Farina. “An ALife-Inspired Evolutionary Algorithm for Dynamic Multiobjective Optimization Problems”. In: *Soft Computing: Methodologies and Applications.* Ed. by Frank Hoffmann et al. Advances in Soft Computing. Berlin, Heidelberg: Springer, 2005, pp. 113–125. ISBN: 978-3-540-32400-3. DOI: 10.1007/3-540-32400-3_9.

[4] Alfredo Arias-Montano, Carlos A. Coello Coello and Efrén Mezura-Montes. “Multiobjective Evolutionary Algorithms in Aeronautical and Aerospace Engineering”. In: *IEEE Transactions on Evolutionary Computation* 16.5 (Oct. 2012), pp. 662–694. ISSN: 1941-0026. DOI: 10.1109/TEVC.2011.2169968.

[5] Radhia Azzouz, Slim Bechikh and Lamjed Ben Said. “Dynamic Multi-objective Optimization Using Evolutionary Algorithms: A Survey”. In: *Recent Advances in Evolutionary Multi-objective Optimization.* Ed. by Slim Bechikh, Rituparna Datta and Abhishek Gupta. Adaptation, Learning, and Optimization. Cham: Springer International Publishing, 2017, pp. 31–70. ISBN: 978-3-319-42978-6. DOI: 10.1007/978-3-319-42978-6_2. URL: https://doi.org/10.1007/978-3-319-42978-6_2 (visited on 13/10/2022).

[6] Radhia Azzouz, Slim Bechikh and Lamjed Ben Said. “A Dynamic Multi-Objective Evolutionary Algorithm Using a Change Severity-Based Adaptive Population Management Strategy”. In: *Soft Computing* 21.4 (4 1st Feb. 2017), pp. 885–906. ISSN: 1433-7479. DOI: 10.1007/s00500-015-1820-4. URL: https://doi.org/10.1007/s00500-015-1820-4 (visited on 17/06/2022).

[7] Jørgen Bang-Jensen and Gregory Z. Gutin. “Hamiltonian, Longest and Vertex-Cheapest Paths and Cycles”. In: *Digraphs: Theory, Algorithms and Applications.* Ed. by Jørgen Bang-Jensen and Gregory Z. Gutin. Springer Monographs in Mathematics. London: Springer, 2009, pp. 227–274. ISBN: 978-1-84800-998-1. DOI: 10.1007/978-1-84800-998-1_6. URL: https://doi.org/10.1007/978-1-84800-998-1_6 (visited on 09/01/2023).

[8] Chunteng Bao et al. “A Novel Non-Dominated Sorting Algorithm for Evolutionary Multi-Objective Optimization”. In: *Journal of Computational Science* 23 (1st Nov. 2017), pp. 31–43. ISSN: 1877-7503. DOI: 10.1016/j.jocs.2017.09.015. URL: https://www.sciencedirect.com/science/article/pii/S1877750317310530 (visited on 10/01/2023).

[9] Harold P. Benson. “Multi-Objective Optimization: Pareto Optimal Solutions, propertiesMulti-objective Optimization: Pareto Optimal Solutions, Properties”. In: *Encyclopedia of Optimization*. Ed. by Christodoulos A. Floudas and Panos M. Pardalos. Boston, MA: Springer US, 2009, pp. 2478–2481. ISBN: 978-0-387-74759-0. DOI: 10.1007/978-0-387-74759-0_426. URL: https://doi.org/10.1007/978-0-387-74759-0_426 (visited on 10/01/2023).

[10] R Bhuvaneswari and K Kalaiselvi. “Naive Bayesian Classification Approach in Healthcare Applications”. In: *International Journal of Computer Science and Telecommunications* 3.1 (2012). ISSN: 2047-3338.

[11] Julian Blank and Kalyanmoy Deb. “Pymoo: Multi-Objective Optimization in Python”. In: *IEEE Access* 8 (2020), pp. 89497–89509. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2990567.

[12] Mohammad Reza Bonyadi and Zbigniew Michalewicz. “Particle Swarm Optimization for Single Objective Continuous Space Problems: A Review”. In: *Evolutionary Computation* 25.1 (1st Mar. 2017), pp. 1–54. ISSN: 1063-6560. DOI: 10.1162/EVCO_r_00180. URL: https://doi.org/10.1162/EVCO_r_00180 (visited on 09/01/2023).

[13] Jürgen Branke et al. “Finding Knees in Multi-objective Optimization”. In: *Parallel Problem Solving from Nature - PPSN VIII*. Ed. by Xin Yao et al. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2004, pp. 722–731. ISBN: 978-3-540-30217-9. DOI: 10.1007/978-3-540-30217-9_73.

[14] Ma. Guadalupe Castillo Tapia and Carlos A. Coello Coello. “Applications of Multi-Objective Evolutionary Algorithms in Economics and Finance: A Survey”. In: *2007 IEEE Congress on Evolutionary Computation*. 2007 IEEE Congress on Evolutionary Computation. Sept. 2007, pp. 532–539. DOI: 10.1109/CEC.2007.4424516.

[15] Liang Chen et al. “Dynamic Multiobjective Evolutionary Algorithm with Adaptive Response Mechanism Selection Strategy”. In: *Knowledge-Based Systems* 246 (21st June 2022), p. 108691. ISSN: 0950-7051. DOI: 10.1016/j.knosys.2022.108691. URL: https://www.sciencedirect.com/science/article/pii/S0950705122003185 (visited on 10/01/2023).

[16] Qingda Chen et al. “A Novel Evolutionary Algorithm for Dynamic Constrained Multiobjective Optimization Problems”. In: *IEEE Transactions on Evolutionary Computation* 24.4 (Aug. 2020), pp. 792–806. ISSN: 1941-0026. DOI: 10.1109/TEVC.2019.2958075.

[17] Renzhi Chen, Ke Li and Xin Yao. “Dynamic Multiobjectives Optimization With a Changing Number of Objectives”. In: *IEEE Transactions on Evolutionary Computation* 22.1 (Feb. 2018), pp. 157–171. ISSN: 1941-0026. DOI: 10.1109/TEVC.2017.2669638.

[18] Jesse Clifton and Eric Laber. “Q-Learning: Theory and Applications”. In: *Annual Review of Statistics and Its Application* 7.1 (2020), pp. 279–301. DOI: 10.1146/annurev-statistics-031219-041220. URL: https://doi.org/10.1146/annurev-statistics-031219-041220 (visited on 10/01/2023).

[19] Thomas Cormen et al. *Introduction to Algorithms*. MIT Press and McGraw–Hill, 2001, pp. 595–601.

[20] Matej Črepinšek, Shih-Hsi Liu and Marjan Mernik. "Exploration and Exploitation in Evolutionary Algorithms: A Survey". In: *ACM Computing Surveys* 45.3 (3rd July 2013), 35:1–35:33. ISSN: 0360-0300. DOI: 10.1145/2480741.2480752. URL: https://doi.org/10.1145/2480741.2480752 (visited on 10/01/2023).

[21] George B. Dantzig. *Mathematical Programming Glossary.* 1996. URL: https://glossary.informs.org/second.php?page=nature.html (visited on 09/01/2023).

[22] Sanjoy Das and Bijaya K. Panigrahi. "Multi-Objective Evolutionary Algorithms:" in: *Encyclopedia of Artificial Intelligence.* Ed. by Juan Ramón Rabuñal Dopico, Julian Dorado and Alejandro Pazos. IGI Global, 2009, pp. 1145–1151. ISBN: 978-1-59904-849-9 978-1-59904-850-5. DOI: 10.4018/978-1-59904-849-9.ch167. URL: http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/978-1-59904-849-9.ch167 (visited on 10/01/2023).

[23] Kenneth De Jong, David Fogel and Hans-Paul Schwefel. "A History of Evolutionary Computation". In: *Handbook of Evolutionary Computation, IOP Publishing Ltd.* 1st Jan. 1997, A2.3:1–12.

[24] K. Deb et al. "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II". In: *IEEE Transactions on Evolutionary Computation* 6.2 (Apr. 2002), pp. 182–197. ISSN: 1941-0026. DOI: 10.1109/4235.996017.

[25] Kalyanmoy Deb, Udaya Bhaskara Rao N. and S. Karthik. "Dynamic Multi-objective Optimization and Decision-Making Using Modified NSGA-II: A Case Study on Hydro-thermal Power Scheduling". In: *Evolutionary Multi-Criterion Optimization.* Ed. by Shigeru Obayashi et al. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2007, pp. 803–817. ISBN: 978-3-540-70928-2. DOI: 10.1007/978-3-540-70928-2_60.

[26] E. W. Dijkstra. "A Note on Two Problems in Connexion with Graphs". In: *Numerische Mathematik* 1.1 (1st Dec. 1959), pp. 269–271. ISSN: 0945-3245. DOI: 10.1007/BF01386390. URL: https://doi.org/10.1007/BF01386390 (visited on 09/01/2023).

[27] A. E. Eiben and J. E. Smith. "Evolutionary Computing: The Origins". In: *Introduction to Evolutionary Computing.* Ed. by A.E. Eiben and J.E. Smith. Natural Computing Series. Berlin, Heidelberg: Springer, 2015, pp. 13–24. ISBN: 978-3-662-44874-8. DOI: 10.1007/978-3-662-44874-8_2. URL: https://doi.org/10.1007/978-3-662-44874-8_2 (visited on 10/01/2023).

[28] A.E. Eiben and J.E. Smith. *Introduction to Evolutionary Computing.* Natural Computing Series. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015. ISBN: 978-3-662-44873-1 978-3-662-44874-8. DOI: 10.1007/978-3-662-44874-8. URL: http://link.springer.com/10.1007/978-3-662-44874-8 (visited on 10/01/2023).

[29] Xuezhou Fan, Ke Li and Kay Chen Tan. "Surrogate Assisted Evolutionary Algorithm Based on Transfer Learning for Dynamic Expensive Multi-Objective Optimisation Problems". In: *2020 IEEE Congress on Evolutionary Computation (CEC).* 2020 IEEE Congress on Evolutionary Computation (CEC). July 2020, pp. 1–8. DOI: 10.1109/CEC48606.2020.9185522.

[30] M. Farina, K. Deb and P. Amato. "Dynamic Multiobjective Optimization Problems: Test Cases, Approximations, and Applications". In: *IEEE Transactions on Evolutionary Computation* 8.5 (Oct. 2004), pp. 425–442. ISSN: 1941-0026. DOI: 10.1109/TEVC.2004.831456.

[31] Marco Farina, Kalyan Deb and Paolo Amato. "Dynamic Multiobjective Optimization Problems: Test Cases, Approximations, and Applications". In: *Evolutionary Computation, IEEE Transactions on* 8 (1st Nov. 2004), pp. 425–442. DOI: 10.1109/TEVC.2004.831456.

[32] Henry Gee. *In Search of Deep Time: Beyond the Fossil Record to a New History of Life*. 1st edition. Ithaca, N.Y: Cornell University Press, 1st Feb. 2001. 272 pp. ISBN: 978-0-8014-8713-2.

[33] Chi-Keong Goh and Kay Tan. "A Coevolutionary Paradigm for Dynamic Multi-Objective Optimization". In: *Evolutionary Multi-Objective Optimization in Uncertain Environments* 186 (1st Jan. 2009). ISSN: 978-3-540-95975-5. DOI: 10.1007/978-3-540-95976-2_6.

[34] Shixiang Gu et al. "Continuous Deep Q-Learning with Model-based Acceleration". In: *Proceedings of The 33rd International Conference on Machine Learning*. International Conference on Machine Learning. PMLR, 11th June 2016, pp. 2829–2838. URL: https://proceedings.mlr.press/v48/gu16.html (visited on 11/01/2023).

[35] Andreia P. Guerreiro, Carlos M. Fonseca and Luís Paquete. "The Hypervolume Indicator: Computational Problems and Algorithms". In: *ACM Computing Surveys* 54.6 (27th July 2021), 119:1–119:42. ISSN: 0360-0300. DOI: 10.1145/3453474. URL: https://doi.org/10.1145/3453474 (visited on 11/01/2023).

[36] Tomasz Gwiazda. "Genetic Algorithms Reference Volume I". In: *TOMASZGWIAZDA E-BOOKS* (). URL: http://www.tomaszgwiazda.com/Genetic_algorithms_reference_first_40_pages.pdf.

[37] Peter E. Hart, Nils J. Nilsson and Bertram Raphael. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". In: *IEEE Transactions on Systems Science and Cybernetics* 4.2 (July 1968), pp. 100–107. ISSN: 2168-2887. DOI: 10.1109/TSSC.1968.300136.

[38] Ahmad Hassanat et al. "Choosing Mutation and Crossover Ratios for Genetic Algorithms—A Review with a New Dynamic Approach". In: *Information* 10.12 (12 Dec. 2019), p. 390. ISSN: 2078-2489. DOI: 10.3390/info10120390. URL: https://www.mdpi.com/2078-2489/10/12/390 (visited on 10/01/2023).

[39] P. B. R. Hazell and R. D. Norton. "Mathematical Programming for Economic Analysis in Agriculture." In: *Biometrics* 43.4 (Dec. 1987), p. 1032. ISSN: 0006341X. DOI: 10.2307/2531573. JSTOR: 2531573.

[40] Ying Huang et al. "A Fitness Landscape Ruggedness Multiobjective Differential Evolution Algorithm with a Reinforcement Learning Strategy". In: *Applied Soft Computing* 96 (1st Nov. 2020), p. 106693. ISSN: 1568-4946. DOI: 10.1016/j.asoc.2020.106693. URL: https://www.sciencedirect.com/science/article/pii/S1568494620306311 (visited on 10/01/2023).

[41] André Hugo et al. "Hydrogen Infrastructure Strategic Planning Using Multi-Objective Optimization". In: *International Journal of Hydrogen Energy* 30.15 (1st Dec. 2005), pp. 1523–1534. ISSN: 0360-3199. DOI: 10.1016/j.ijhydene.2005.04.017. URL: https://www.sciencedirect.com/science/article/pii/S0360319905001163 (visited on 09/01/2023).

[42] C. L. (Ching-Lai) Hwang. *Multiple Objective Decision Making, Methods and Applications : A State-of-the-Art Survey*. In collab. with Internet Archive. Berlin ; New York : Springer-Verlag, 1979. 372 pp. ISBN: 978-0-387-09111-2 978-3-540-09111-0. URL: http://archive.org/details/multipleobjectiv0000hwan (visited on 09/01/2023).

[43] Hisao Ishibuchi et al. “Modified Distance Calculation in Generational Distance and Inverted Generational Distance”. In: *Evolutionary Multi-Criterion Optimization*. Ed. by António Gaspar-Cunha, Carlos Henggeler Antunes and Carlos Coello Coello. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2015, pp. 110–125. ISBN: 978-3-319-15892-1. DOI: 10.1007/978-3-319-15892-1_8.

[44] Goshgar Ismayilov and Haluk Rahmi Topcuoglu. “Dynamic Multi-Objective Workflow Scheduling for Cloud Computing Based on Evolutionary Algorithms”. In: *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*. 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion). Dec. 2018, pp. 103–108. DOI: 10.1109/UCC-Companion.2018.00042.

[45] Min Jiang et al. “Transfer Learning-Based Dynamic Multiobjective Optimization Algorithms”. In: *IEEE Transactions on Evolutionary Computation* 22.4 (Aug. 2018), pp. 501–514. ISSN: 1941-0026. DOI: 10.1109/TEVC.2017.2771451.

[46] Shouyong Jiang and Shengxiang Yang. “Evolutionary Dynamic Multiobjective Optimization: Benchmarks and Algorithm Comparisons”. In: *IEEE Transactions on Cybernetics* 47.1 (Jan. 2017), pp. 198–211. ISSN: 2168-2275. DOI: 10.1109/TCYB.2015.2510698.

[47] Shouyong Jiang, Juan Zou and Xin Yao. “Evolutionary Dynamic Multi-Objective Optimisation: A Survey”. In: *ACM Computing Surveys* 55 (28th Mar. 2022). DOI: 10.1145/3524495.

[48] Shouyong Jiang et al. “Benchmark Problems for CEC2018 Competition on Dynamic Multiobjective Optimisation”. In: (2018).

[49] Yaochu Jin and Bernhard Sendhoff. “Constructing Dynamic Optimization Test Problems Using the Multi-objective Optimization Concept”. In: *Applications of Evolutionary Computing*. Ed. by Günther R. Raidl et al. Red. by Takeo Kanade et al. Vol. 3005. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 525–536. ISBN: 978-3-540-21378-9 978-3-540-24653-4. DOI: 10.1007/978-3-540-24653-4_53. URL: http://link.springer.com/10.1007/978-3-540-24653-4_53 (visited on 23/06/2022).

[50] Sourabh Katoch, Sumit Singh Chauhan and Vijay Kumar. “A Review on Genetic Algorithm: Past, Present, and Future”. In: *Multimedia Tools and Applications* 80.5 (1st Feb. 2021), pp. 8091–8126. ISSN: 1573-7721. DOI: 10.1007/s11042-020-10139-6. URL: https://doi.org/10.1007/s11042-020-10139-6 (visited on 09/01/2023).

[51] Joshua D. Knowles, Richard A. Watson and David W. Corne. “Reducing Local Optima in Single-Objective Problems by Multi-objectivization”. In: *Evolutionary Multi-Criterion Optimization*. Ed. by Eckart Zitzler et al. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2001, pp. 269–283. ISBN: 978-3-540-44719-1. DOI: 10.1007/3-540-44719-9_19.

[52] Jens Kober, J. Bagnell and Jan Peters. “Reinforcement Learning in Robotics: A Survey”. In: *The International Journal of Robotics Research* 32 (1st Sept. 2013), pp. 1238–1274. ISSN: 978-3-642-27644-6. DOI: 10.1177/0278364913495721.

[53] Wee Tat Koo, Chi Keong Goh and Kay Chen Tan. “A Predictive Gradient Strategy for Multiobjective Evolutionary Algorithms in a Fast Changing Environment”. In: *Memetic Computing* 2.2 (2 1st June 2010), pp. 87–110. ISSN: 1865-9292. DOI: 10.1007/s12293-009-0026-7. URL: https://doi.org/10.1007/s12293-009-0026-7 (visited on 17/06/2022).

[54] Oliver Kramer. "Scikit-Learn". In: *Machine Learning for Evolution Strategies*. Ed. by Oliver Kramer. Studies in Big Data. Cham: Springer International Publishing, 2016, pp. 45–53. ISBN: 978-3-319-33383-0. DOI: 10.1007/978-3-319-33383-0_5. URL: https://doi.org/10.1007/978-3-319-33383-0_5 (visited on 11/01/2023).

[55] K Ming Leung. "Naive Bayesian Classifier". In: (2007). URL: https://cse.engineering.nyu.edu/~mleung/FRE7851/f07/naiveBayesianClassifier.pdf.

[56] Changhe Li, Shengxiang Yang and Sanyou Zeng. "Dynamic Multi-objective Optimization for Multi-objective Vehicle Routing Problem with Real-time Traffic Conditions". In: *Developments in Advanced Control and Intelligent Automation for Complex Systems*. Ed. by Min Wu, Witold Pedrycz and Luefeng Chen. Studies in Systems, Decision and Control. Cham: Springer International Publishing, 2021, pp. 289–307. ISBN: 978-3-030-62147-6. DOI: 10.1007/978-3-030-62147-6_11. URL: https://doi.org/10.1007/978-3-030-62147-6_11 (visited on 09/01/2023).

[57] J J Liang et al. "Problem Definitions and Evaluation Criteria for the CEC 2015 Competition on Learning-based Real-Parameter Single Objective Optimization". In: (2015).

[58] Zhengping Liang et al. "Hybrid of Memory and Prediction Strategies for Dynamic Multiobjective Optimization". In: *Information Sciences* 485 (1st June 2019), pp. 200–218. ISSN: 0020-0255. DOI: 10.1016/j.ins.2019.01.066. URL: https://www.sciencedirect.com/science/article/pii/S0020025519300787 (visited on 10/01/2023).

[59] Adam Lipowski and Dorota Lipowska. "Roulette-Wheel Selection via Stochastic Acceptance". In: *Physica A: Statistical Mechanics and its Applications* 391.6 (Mar. 2012), pp. 2193–2196. ISSN: 03784371. DOI: 10.1016/j.physa.2011.12.004. arXiv: 1109.3627 `[cond-mat, physics:physics]`. URL: http://arxiv.org/abs/1109.3627 (visited on 10/01/2023).

[60] Min Liu et al. "An Adaptive Diversity Introduction Method for Dynamic Evolutionary Multiobjective Optimization". In: Proceedings of the 2014 IEEE Congress on Evolutionary Computation, CEC 2014. 1st July 2014, pp. 3160–3167. DOI: 10.1109/CEC.2014.6900364.

[61] Yang Liu et al. "Generational Distance Indicator-Based Evolutionary Algorithm With an Improved Niching Method for Many-Objective Optimization Problems". In: *IEEE Access* 7 (2019), pp. 63881–63891. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2916634.

[62] Xuemin Ma et al. "Multiregional Co-Evolutionary Algorithm for Dynamic Multiobjective Optimization". In: *Information Sciences* 545 (4th Feb. 2021), pp. 1–24. ISSN: 0020-0255. DOI: 10.1016/j.ins.2020.07.009. URL: https://www.sciencedirect.com/science/article/pii/S0020025520306721 (visited on 10/01/2023).

[63] Joaquim Martins and Andrew Ning. *Engineering Design Optimization*. 1st Oct. 2021. ISBN: 978-1-108-83341-7. DOI: 10.1017/9781108980647.

[64] Ujjwal Maulik, Sanghamitra Bandyopadhyay and Anirban Mukhopadhyay. *Multiobjective Genetic Algorithms for Clustering: Applications in Data Mining and Bioinformatics*. Springer Science & Business Media, 1st Sept. 2011. 292 pp. ISBN: 978-3-642-16615-0. Google Books: bzMzUHyYEBQC.

[65] Mark Merezhnikov and Alexander Hvatov. "Multi-Objective Closed-Form Algebraic Expressions Discovery Approach Application to the Synthetic Time-Series Generation". In: *Procedia Computer Science* 193 (2021), pp. 285–294. ISSN: 18770509. DOI: 10.1016/j.procs.2021.10.029. URL: https://linkinghub.elsevier.com/retrieve/pii/S1877050921020706 (visited on 10/01/2023).

[66] Kaisa Miettinen. “A Posteriori Methods”. In: *Nonlinear Multiobjective Optimization.* Ed. by Kaisa Miettinen. International Series in Operations Research & Management Science. Boston, MA: Springer US, 1998, pp. 77–113. ISBN: 978-1-4615-5563-6. DOI: 10.1007/978-1-4615-5563-6_4. URL: https://doi.org/10.1007/978-1-4615-5563-6_4 (visited on 11/01/2023).

[67] Kaisa Miettinen. *Nonlinear Multiobjective Optimization.* Springer Science & Business Media, 1999. 324 pp. ISBN: 978-0-7923-8278-2. Google Books: ha_zLdNtXSMC.

[68] Brad L Miller and David E. Goldberg. “Genetic Algorithms, Tournament Selection, and the Effects of Noise”. In: *Complex Systems* 9.3 (1995).

[69] Thomas M. Moerland et al. “Model-Based Reinforcement Learning: A Survey”. In: *Foundations and Trends® in Machine Learning* 16.1 (3rd Jan. 2023), pp. 1–118. ISSN: 1935-8237, 1935-8245. DOI: 10.1561/2200000086. URL: https://www.nowpublishers.com/article/Details/MAL-086 (visited on 10/01/2023).

[70] Idel Montalvo et al. “Agent Swarm Optimization: A Platform to Solve Complex Optimization Problems”. In: Proceedings of the 7th International Conference on Engineering Computational Technology. 1st Jan. 2010. ISBN: 978-1-905088-41-6. DOI: 10.4203/ccp.94.19.

[71] A. Mostafaie et al. “Comparing Multi-Objective Optimization Techniques to Calibrate a Conceptual Hydrological Model Using in Situ Runoff and Daily GRACE Data”. In: *Computational Geosciences* 22.3 (1st June 2018), pp. 789–814. ISSN: 1573-1499. DOI: 10.1007/s10596-018-9726-8. URL: https://doi.org/10.1007/s10596-018-9726-8 (visited on 09/01/2023).

[72] Quyet Nguyen, Noel Teku and Tamal Bose. “Epsilon Greedy Strategy for Hyper Parameters Tuning of A Neural Network Equalizer”. In: *2021 12th International Symposium on Image and Signal Processing and Analysis (ISPA).* 2021 12th International Symposium on Image and Signal Processing and Analysis (ISPA). Sept. 2021, pp. 209–212. DOI: 10.1109/ISPA52656.2021.9552055.

[73] P. Pandian et al. “Secondary Population Implementation in Multi-Objective Evolutionary Algorithm for Scheduling of FMS”. In: *The International Journal of Advanced Manufacturing Technology* 57 (1st Dec. 2011). DOI: 10.1007/s00170-011-3359-6.

[74] Jack Parker-Holder et al. “Automated Reinforcement Learning (AutoRL): A Survey and Open Problems”. In: *Journal of Artificial Intelligence Research* 74 (1st June 2022), pp. 517–568. ISSN: 1076-9757. DOI: 10.1613/jair.1.13596. arXiv: 2201.03916 [cs]. URL: http://arxiv.org/abs/2201.03916 (visited on 09/01/2023).

[75] Zhou Peng, Jinhua Zheng and Juan Zou. “A Population Diversity Maintaining Strategy Based on Dynamic Environment Evolutionary Model for Dynamic Multiobjective Optimization”. In: *2014 IEEE Congress on Evolutionary Computation (CEC)* (2014). DOI: 10.1109/CEC.2014.6900268.

[76] Zhou Peng et al. “Novel Prediction and Memory Strategies for Dynamic Multiobjective Optimization”. In: *Soft Computing* 9.19 (19 2015), pp. 2633–2653. ISSN: 1432-7643, 1433-7479. DOI: 10.1007/s00500-014-1433-3. URL: https://www.infona.pl//resource/bwmeta1.element.springer-doi-10_1007-S00500-014-1433-3 (visited on 17/06/2022).

[77] Zhou Peng et al. “Novel Prediction and Memory Strategies for Dynamic Multiobjective Optimization”. In: *Soft Computing* 9.19 (2015), pp. 2633–2653. ISSN: 1432-7643, 1433-7479. DOI: 10.1007/s00500-014-1433-3. URL: https://www.infona.pl//resource/bwmeta1.element.springer-doi-10_1007-S00500-014-1433-3 (visited on 10/01/2023).

[78] Carlo Raquel and Xin Yao. “Dynamic Multi-objective Optimization: A Survey of the State-of-the-Art”. In: *Evolutionary Computation for Dynamic Optimization Problems.* Ed. by Shengxiang Yang and Xin Yao. Vol. 490. Studies in Computational Intelligence. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 85–106. ISBN: 978-3-642-38415-8 978-3-642-38416-5. DOI: 10.1007/978-3-642-38416-5_4. URL: http://link.springer.com/10.1007/978-3-642-38416-5_4 (visited on 14/10/2022).

[79] Hendrik Richter. “Dynamic Fitness Landscape Analysis”. In: *Evolutionary Computation for Dynamic Optimization Problems.* Ed. by Shengxiang Yang and Xin Yao. Vol. 490. Studies in Computational Intelligence. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 269–297. ISBN: 978-3-642-38415-8 978-3-642-38416-5. DOI: 10.1007/978-3-642-38416-5_11. URL: http://link.springer.com/10.1007/978-3-642-38416-5_11 (visited on 10/01/2023).

[80] Guido van Rossum and Fred L. Drake. *An Introduction to Python: Release 2.5.* 2. print. Bristol: Network Theory Limited, 2006. 155 pp. ISBN: 978-0-9541617-6-7.

[81] Manuela Ruiz-Montiel, Lawrence Mandow and José-Luis Pérez-de-la-Cruz. “A Temporal Difference Method for Multi-Objective Reinforcement Learning”. In: *Neurocomputing.* Multiobjective Reinforcement Learning: Theory and Applications 263 (8th Nov. 2017), pp. 15–25. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2016.10.100. URL: https://www.sciencedirect.com/science/article/pii/S0925231217310998 (visited on 09/01/2023).

[82] Shaaban Sahmoud and Haluk Rahmi Topcuoglu. “Exploiting Characterization of Dynamism for Enhancing Dynamic Multi-Objective Evolutionary Algorithms”. In: *Applied Soft Computing* 85.C (1st Dec. 2019). ISSN: 1568-4946. DOI: 10.1016/j.asoc.2019.105783. URL: https://doi.org/10.1016/j.asoc.2019.105783 (visited on 10/01/2023).

[83] Hussein Samma, Chee Peng Lim and Junita Mohamad Saleh. “A New Reinforcement Learning-based Memetic Particle Swarm Optimizer”. In: *Applied Soft Computing* 43.C (C 1st June 2016), pp. 276–297. ISSN: 1568-4946. DOI: 10.1016/j.asoc.2016.01.006. URL: https://doi.org/10.1016/j.asoc.2016.01.006 (visited on 17/06/2022).

[84] Mucahid Mustafa Saritas and Ali Yasar. “Performance Analysis of ANN and Naive Bayes Classification Algorithm for Data Classification”. In: *International Journal of Intelligent Systems and Applications in Engineering* 7.2 (2 30th June 2019), pp. 88–91. ISSN: 2147-6799. DOI: 10.18201//ijisae.2019252786. URL: https://ijisae.org/index.php/IJISAE/article/view/934 (visited on 10/01/2023).

[85] RUHUL SARKER, JOARDER KAMRUZZAMAN and CHARLES NEWTON. “EVOLUTIONARY OPTIMIZATION (EvOpt): A BRIEF REVIEW AND ANALYSIS”. In: *International Journal of Computational Intelligence and Applications* (20th Nov. 2011). DOI: 10.1142/S1469026803001051. URL: https://www.worldscientific.com/doi/abs/10.1142/S1469026803001051 (visited on 09/01/2023).

[86] Marc Schoenauer and Zbigniew Michalewicz. “Boundary Operators for Constrained Parameter Optimization Problems”. In: (25th May 1997).

[87] Oliver Schutze, Adriana Lara and Carlos A. Coello Coello. "On the Influence of the Number of Objectives on the Hardness of a Multiobjective Optimization Problem". In: *IEEE Transactions on Evolutionary Computation* 15.4 (Aug. 2011), pp. 444–455. ISSN: 1941-0026. DOI: 10.1109/TEVC.2010.2064321.

[88] Yansen Su et al. "A Multi-Objective Optimization Method for Identification of Module Biomarkers for Disease Diagnosis". In: *Methods*. Deep Networks and Network Representation in Bioinformatics 192 (1st Aug. 2021), pp. 35–45. ISSN: 1046-2023. DOI: 10.1016/j.ymeth.2020.09.001. URL: https://www.sciencedirect.com/science/article/pii/S1046202320301961 (visited on 09/01/2023).

[89] Sonja Surjanovic and Derek Bingham. *Ackley Function*. 2013. URL: https://www.sfu.ca/~ssurjano/ackley.html (visited on 09/01/2023).

[90] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning. Cambridge, Mass: MIT Press, 1998. 322 pp. ISBN: 978-0-262-19398-6.

[91] Mikkel Thorup. "Undirected Single-Source Shortest Paths with Positive Integer Weights in Linear Time". In: *Journal of the ACM* 46.3 (1st May 1999), pp. 362–394. ISSN: 0004-5411. DOI: 10.1145/316542.316548. URL: https://doi.org/10.1145/316542.316548 (visited on 09/01/2023).

[92] Yuan Tian et al. "Real-Time Model Calibration with Deep Reinforcement Learning". 9th June 2020. arXiv: 2006.04001 [cs, eess]. URL: http://arxiv.org/abs/2006.04001 (visited on 26/04/2022).

[93] Peter J. M. van Laarhoven and Emile H. L. Aarts. "Simulated Annealing". In: *Simulated Annealing: Theory and Applications*. Ed. by Peter J. M. van Laarhoven and Emile H. L. Aarts. Mathematics and Its Applications. Dordrecht: Springer Netherlands, 1987, pp. 7–15. ISBN: 978-94-015-7744-1. DOI: 10.1007/978-94-015-7744-1_2. URL: https://doi.org/10.1007/978-94-015-7744-1_2 (visited on 09/01/2023).

[94] Yu Wang and Bin Li. "Multi-Strategy Ensemble Evolutionary Algorithm for Dynamic Multi-Objective Optimization". In: *Memetic Computing* 2.1 (1st Mar. 2010), pp. 3–24. ISSN: 1865-9292. DOI: 10.1007/s12293-009-0012-0. URL: https://doi.org/10.1007/s12293-009-0012-0 (visited on 10/01/2023).

[95] Yuping Wang and Chuangyin Dang. "An Evolutionary Algorithm for Dynamic Multi-Objective Optimization". In: *Applied Mathematics and Computation*. Special Issue on Life System Modeling and Bio-Inspired Computing for LSMS 2007 205.1 (1st Nov. 2008), pp. 6–18. ISSN: 0096-3003. DOI: 10.1016/j.amc.2008.05.151. URL: https://www.sciencedirect.com/science/article/pii/S0096300308003135 (visited on 10/01/2023).

[96] Thomas Weise et al. "Why Is Optimization Difficult?" In: *Nature-Inspired Algorithms for Optimisation*. Ed. by Raymond Chiong. Red. by Janusz Kacprzyk. Vol. 193. Studies in Computational Intelligence. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 1–50. ISBN: 978-3-642-00266-3 978-3-642-00267-0. DOI: 10.1007/978-3-642-00267-0_1. URL: http://link.springer.com/10.1007/978-3-642-00267-0_1 (visited on 09/01/2023).

[97] Karl Weiss, Taghi M. Khoshgoftaar and DingDing Wang. "A Survey of Transfer Learning". In: *Journal of Big Data* 3.1 (28th May 2016), p. 9. ISSN: 2196-1115. DOI: 10.1186/s40537-016-0043-6. URL: https://doi.org/10.1186/s40537-016-0043-6 (visited on 10/01/2023).

[98] P. J. Withers. “Residual Stress and Its Role in Failure”. In: *Reports on Progress in Physics* 70.12 (Nov. 2007), p. 2211. ISSN: 0034-4885. DOI: 10.1088/0034-4885/70/12/R04. URL: https://dx.doi.org/10.1088/0034-4885/70/12/R04 (visited on 11/01/2023).

[99] Yan Wu, Yaochu Jin and Xiaoxiong Liu. “A Directed Search Strategy for Evolutionary Dynamic Multiobjective Optimization”. In: *Soft Computing - A Fusion of Foundations, Methodologies and Applications* 19.11 (11 1st Nov. 2015), pp. 3221–3235. ISSN: 1432-7643. DOI: 10.1007/s00500-014-1477-4. URL: https://doi.org/10.1007/s00500-014-1477-4 (visited on 17/06/2022).

[100] Yulong Ye et al. “Knowledge Guided Bayesian Classification for Dynamic Multi-Objective Optimization”. In: *Knowledge-Based Systems* 250 (Aug. 2022), p. 109173. ISSN: 09507051. DOI: 10.1016/j.knosys.2022.109173. URL: https://linkinghub.elsevier.com/retrieve/pii/S0950705122005810 (visited on 12/10/2022).

[101] Sanyou Zeng et al. “A General Framework of Dynamic Constrained Multiobjective Evolutionary Algorithms for Constrained Optimization”. In: *IEEE Transactions on Cybernetics* 47.9 (Sept. 2017), pp. 2678–2688. ISSN: 2168-2275. DOI: 10.1109/TCYB.2017.2647742.

[102] Qingfu Zhang and Hui Li. “MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition”. In: *IEEE Transactions on Evolutionary Computation* 11.6 (Dec. 2007), pp. 712–731. ISSN: 1941-0026. DOI: 10.1109/TEVC.2007.892759.

[103] Xingyi Zhang, Ye Tian and Yaochu Jin. “A Knee Point-Driven Evolutionary Algorithm for Many-Objective Optimization”. In: *IEEE Transactions on Evolutionary Computation* 19.6 (Dec. 2015), pp. 761–776. ISSN: 1941-0026. DOI: 10.1109/TEVC.2014.2378512.

[104] Zhuhong Zhang. “Multiobjective Optimization Immune Algorithm in Dynamic Environments and Its Application to Greenhouse Control”. In: *Applied Soft Computing* 8.2 (1st Mar. 2008), pp. 959–971. ISSN: 1568-4946. DOI: 10.1016/j.asoc.2007.07.005. URL: https://www.sciencedirect.com/science/article/pii/S1568494607000841 (visited on 10/01/2023).

[105] Jinhua Zheng et al. “A Dynamic Multi-Objective Particle Swarm Optimization Algorithm Based on Adversarial Decomposition and Neighborhood Evolution”. In: *Swarm and Evolutionary Computation* 69 (1st Mar. 2022), p. 100987. ISSN: 2210-6502. DOI: 10.1016/j.swevo.2021.100987. URL: https://www.sciencedirect.com/science/article/pii/S2210650221001498 (visited on 10/01/2023).

[106] Aimin Zhou, Yaochu Jin and Qingfu Zhang. “A Population Prediction Strategy for Evolutionary Dynamic Multiobjective Optimization”. In: *IEEE transactions on cybernetics* 44 (26th Feb. 2013). DOI: 10.1109/TCYB.2013.2245892.

[107] Aimin Zhou et al. “Prediction-Based Population Re-initialization for Evolutionary Dynamic Multi-objective Optimization”. In: *Evolutionary Multi-Criterion Optimization*. Ed. by Shigeru Obayashi et al. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2007, pp. 832–846. ISBN: 978-3-540-70928-2. DOI: 10.1007/978-3-540-70928-2_62.

[108] Aimin Zhou et al. “Prediction-Based Population Re-initialization for Evolutionary Dynamic Multi-objective Optimization”. In: *Evolutionary Multi-Criterion Optimization*. Ed. by Shigeru Obayashi et al. Vol. 4403. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 832–846. ISBN: 978-3-540-70927-5 978-3-540-70928-2. DOI: 10.1007/978-3-540-70928-2_62. URL: http://link.springer.com/10.1007/978-3-540-70928-2_62 (visited on 11/01/2023).

[109] Fei Zou et al. “A Reinforcement Learning Approach for Dynamic Multi-Objective Optimization”. In: *Information Sciences* 546 (Feb. 2021), pp. 815–834. ISSN: 00200255. DOI: 10.1016/j.ins.2020.08.101. URL: https://linkinghub.elsevier.com/retrieve/pii/S0020025520308677 (visited on 14/10/2022).

A. Appendix

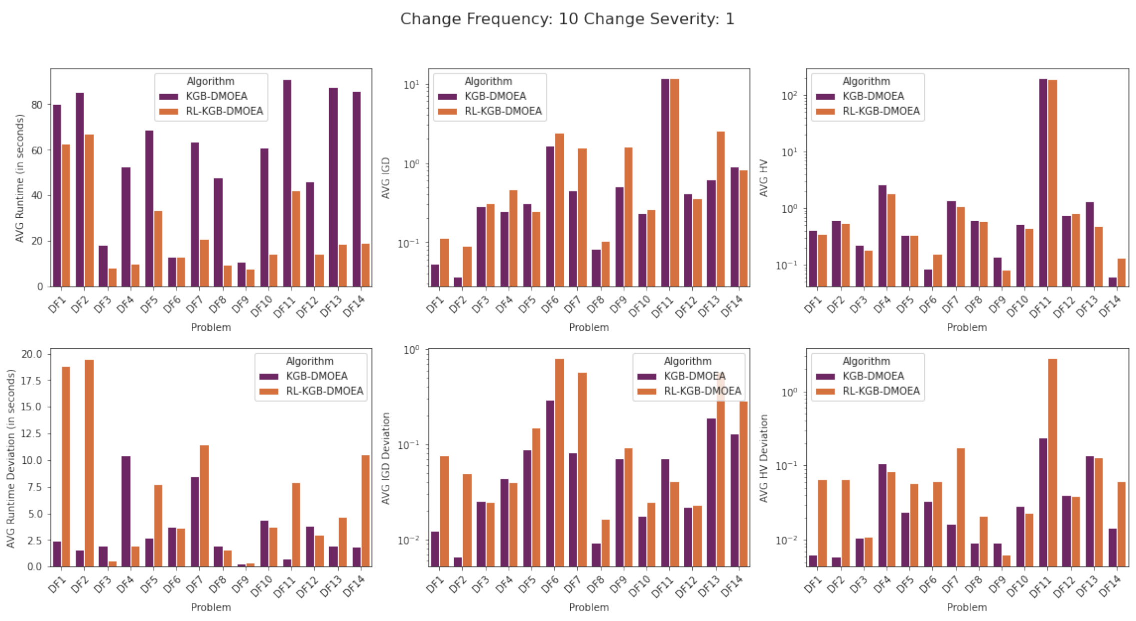


Figure A.1: Results of the KGB-DMOEA and RL-KGB-DMOEA on the DF1 Benchmark Suite, Change Frequency 10, Change Severity 1.

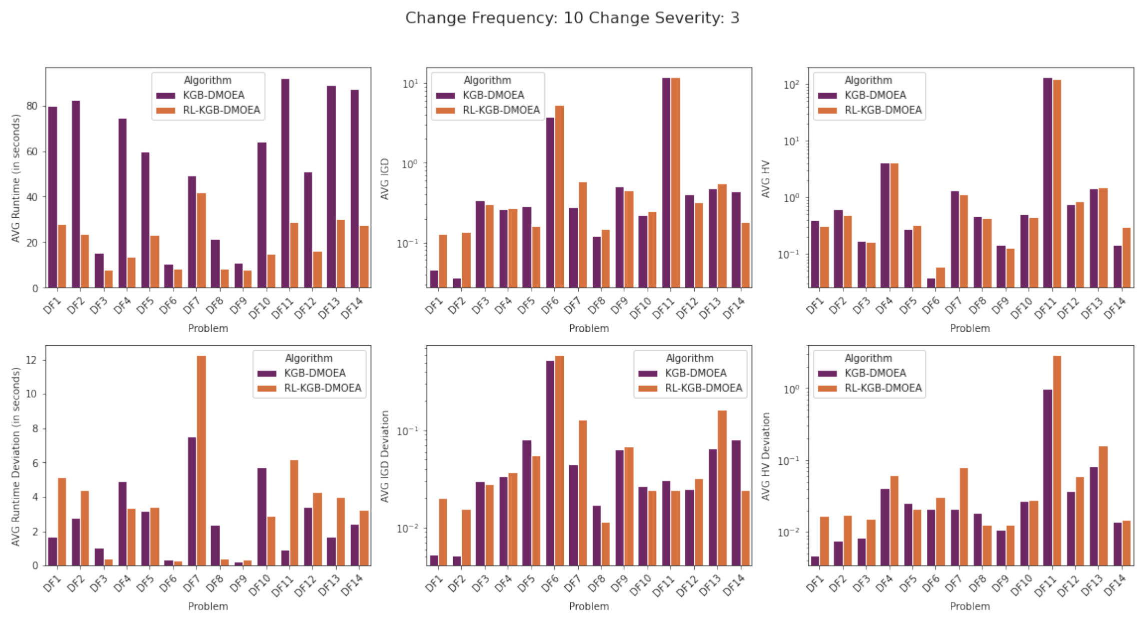


Figure A.2: Results of the KGB-DMOEA and RL-KGB-DMOEA on the DF1 Benchmark Suite, Change Frequency 10, Change Severity 3.

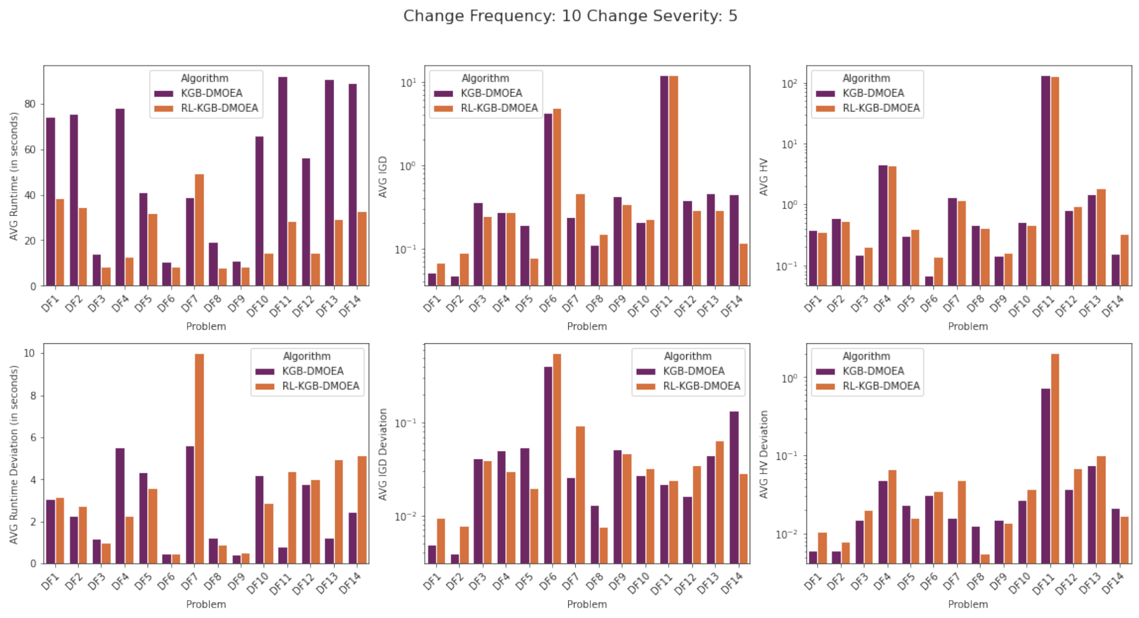


Figure A.3: Results of the KGB-DMOEA and RL-KGB-DMOEA on the DF1 Benchmark Suite, Change Frequency 10, Change Severity 5.

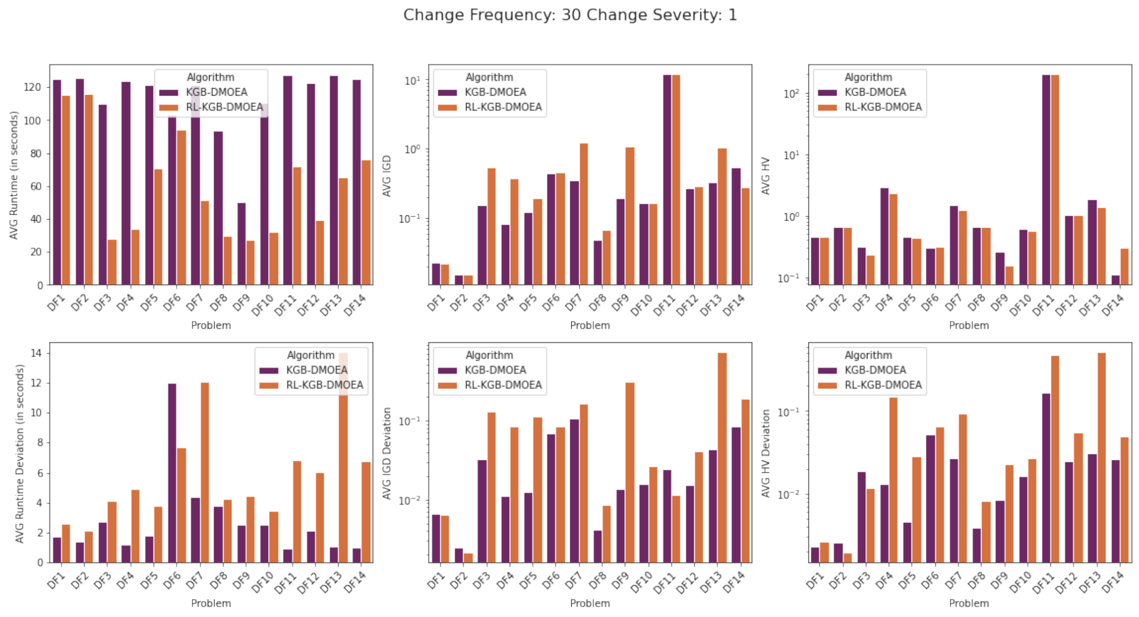


Figure A.4: Results of the KGB-DMOEA and RL-KGB-DMOEA on the DF1 Benchmark Suite, Change Frequency 30, Change Severity 1.

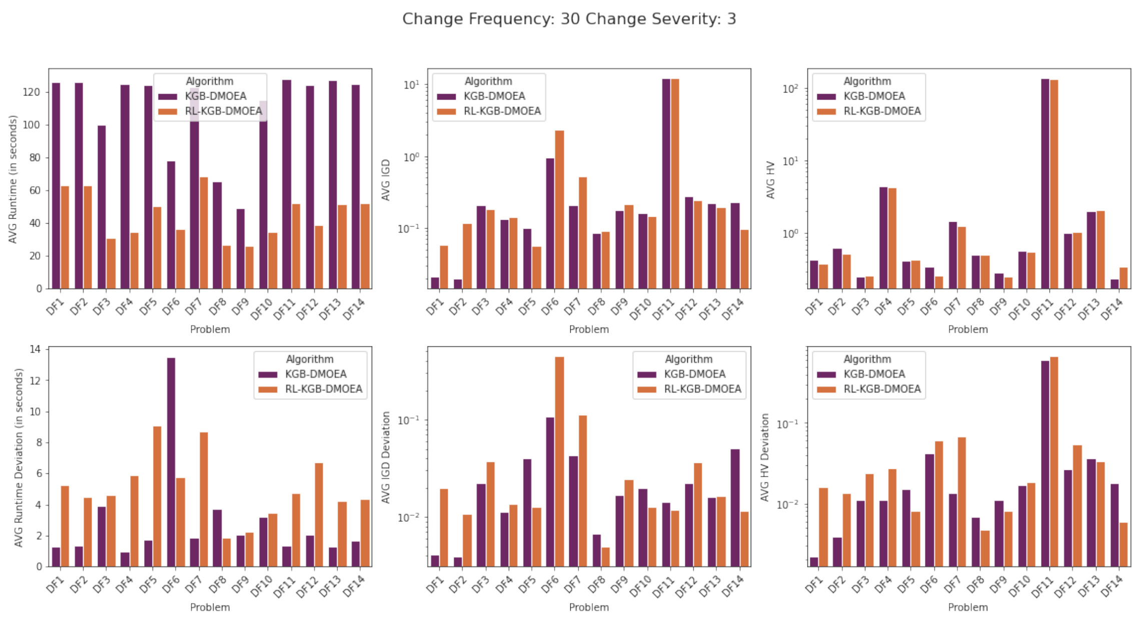


Figure A.5: Results of the KGB-DMOEA and RL-KGB-DMOEA on the DF1 Benchmark Suite, Change Frequency 30, Change Severity 3.

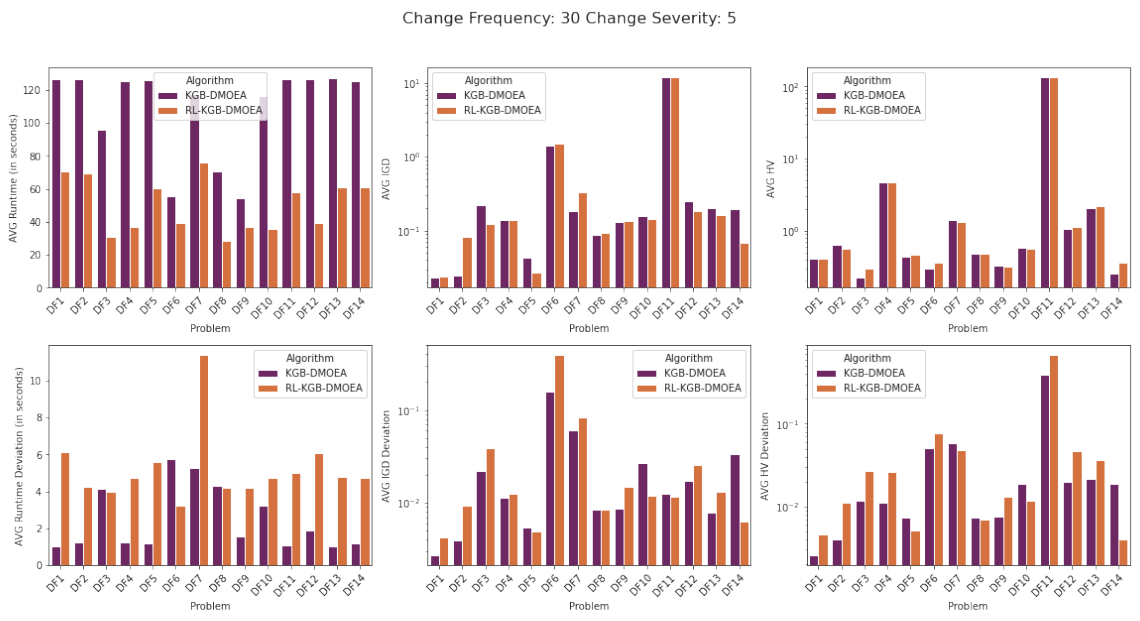


Figure A.6: Results of the KGB-DMOEA and RL-KGB-DMOEA on the DF1 Benchmark Suite, Change Frequency 30, Change Severity 5.

	avg_time	avg_time	avg_time_dev	avg_time_dev	avg_igd	avg_igd	avg_igd_dev	avg_igd_dev	avg_hv	avg_hv	avg_hv_dev	avg_hv_dev
Algorithm	KGB-DMOEA	RL-KGB-DMOEA	KGB-DMOEA	RL-KGB-DMOEA	KGB-DMOEA	RL-KGB-DMOEA	KGB-DMOEA	RL-KGB-DMOEA	KGB-DMOEA	RL-KGB-DMOEA	KGB-DMOEA	RL-KGB-DMOEA
Problem												
DF1	79.9413	28.2073	1.6747	5.1606	0.0463	0.1281	0.0053	0.0204	0.3973	0.3137	0.0047	0.0166
DF2	82.4799	23.7468	2.7762	4.412	0.036	0.1357	0.0052	0.0155	0.6129	0.4779	0.0077	0.0172
DF3	15.5363	8.0146	1.0658	0.4174	0.3445	0.3006	0.0304	0.0278	0.1697	0.1641	0.0085	0.0152
DF4	74.3758	13.7372	4.9083	3.3793	0.2639	0.2679	0.0342	0.0369	4.1197	4.0697	0.0409	0.0621
DF5	59.8172	23.3078	3.2106	3.4183	0.2897	0.1624	0.0809	0.0561	0.2751	0.3275	0.0257	0.0207
DF6	10.6011	8.37	0.3556	0.2836	3.8404	5.4112	0.5253	0.5979	0.0377	0.0606	0.0211	0.0308
DF7	49.3231	41.8494	7.5572	12.2489	0.2761	0.5903	0.0448	0.129	1.341	1.1447	0.0206	0.0788
DF8	21.3226	8.3031	2.3536	0.3936	0.1217	0.1465	0.0171	0.0114	0.471	0.4337	0.0184	0.0128
DF9	10.9237	7.7915	0.2619	0.3331	0.5023	0.4593	0.0646	0.0693	0.1423	0.1274	0.0108	0.0128
DF10	64.2857	14.9028	5.7428	2.8902	0.2182	0.2455	0.0269	0.0244	0.5072	0.4484	0.0266	0.0283
DF11	92.1514	28.7486	0.9231	6.1814	11.8379	11.8803	0.0307	0.0241	133.3321	125.5758	0.9837	2.9037
DF12	50.9934	16.1173	3.443	4.2875	0.4028	0.322	0.025	0.0325	0.7437	0.8713	0.0367	0.059
DF13	89.1311	30.0986	1.6951	3.9831	0.4847	0.5575	0.0661	0.1626	1.4565	1.5088	0.0813	0.1585
DF14	87.2977	27.4468	2.4174	3.2666	0.446	0.1819	0.081	0.0242	0.1418	0.3001	0.0139	0.0147

Table A.1: Real Valued Results of the KGB-DMOEA and RL-KGB-DMOEA on the DF Benchmark Suite, Change Frequency 10, Change Severity 3.

	avg_time	avg_time	avg_time_dev	avg_time_dev	avg_igd	avg_igd	avg_igd_dev	avg_igd_dev	avg_hv	avg_hv	avg_hv_dev	avg_hv_dev
Algorithm	KGB-DMOEA	RL-KGB-DMOEA	KGB-DMOEA	RL-KGB-DMOEA	KGB-DMOEA	RL-KGB-DMOEA	KGB-DMOEA	RL-KGB-DMOEA	KGB-DMOEA	RL-KGB-DMOEA	KGB-DMOEA	RL-KGB-DMOEA
Problem												
DF1	126.3098	63.0185	1.2805	5.2802	0.021	0.0596	0.0041	0.0196	0.4271	0.384	0.0022	0.0161
DF2	126.2706	63.2745	1.341	4.5083	0.0199	0.1176	0.0039	0.0109	0.6392	0.5175	0.0039	0.0134
DF3	99.9274	30.7814	3.9127	4.5836	0.2086	0.1864	0.022	0.0378	0.2493	0.2609	0.0112	0.024
DF4	124.7952	34.8705	0.9491	5.9073	0.1341	0.1435	0.0114	0.0135	4.354	4.3277	0.011	0.0272
DF5	124.4325	50.5969	1.7143	9.0839	0.0992	0.0563	0.0405	0.0127	0.4182	0.4357	0.0153	0.0081
DF6	78.2512	36.6447	13.5259	5.7754	0.9408	2.3101	0.1067	0.449	0.347	0.2618	0.0419	0.0612
DF7	123.2231	68.8879	1.8753	8.7065	0.2078	0.524	0.0428	0.1132	1.4555	1.2473	0.0137	0.0673
DF8	65.3736	27.0724	3.6999	1.87	0.0852	0.0914	0.0067	0.005	0.5122	0.506	0.0069	0.0047
DF9	49.1621	26.27	2.0831	2.27	0.1799	0.2174	0.0168	0.0243	0.2912	0.251	0.011	0.0081
DF10	115.3036	34.8461	3.198	3.481	0.1641	0.1487	0.0199	0.0127	0.5736	0.5518	0.017	0.0183
DF11	128.2561	52.4623	1.3527	4.7194	11.9822	11.9674	0.0143	0.0119	135.3979	133.2049	0.6001	0.6772
DF12	124.2949	38.7903	2.0376	6.7401	0.2774	0.2437	0.0221	0.0363	1.0147	1.0428	0.0267	0.0548
DF13	127.2257	51.5264	1.2614	4.2407	0.2199	0.197	0.0162	0.0165	2.0236	2.1006	0.0363	0.0332
DF14	125.1936	51.9673	1.67	4.3626	0.2322	0.0984	0.0509	0.0115	0.2368	0.3447	0.0181	0.006

Table A.2: Real Valued Results of the KGB-DMOEA and RL-KGB-DMOEA on the DF Benchmark Suite, Change Frequency 30, Change Severity 3.

UNIVERSITÄT TRIER

Eidesstattliche Erklärung / Plagiatprüfung

Name: ____________________ Vorname: ____________________ Matr.-Nr. ______________

Adresse

____

Mail: ___________________________ Studienfach/-fächer: _________________________________

Hiermit erkläre ich an Eides statt, dass ich die vorliegende, an diese Erklärung angefügte/n/s

☐ Seminararbeit	☐ Bachelorarbeit	☐ Bericht des Studienprojekts*
☐ Masterarbeit	☐ Paper im Forschungspraktikum*	☐ Portfolio-Dokument*

* von jedem Gruppenmitglied auszufüllen und zu unterschreiben

Titel der Arbeit:

selbst angefertigt und alle benutzten Hilfsmittel in der Arbeit angegeben habe.

Ich habe die beigefügte Arbeit noch nicht zum Erwerb eines anderen Leistungsnachweises eingereicht.

Mit der Plagiatprüfung meiner Arbeit durch ein internetbasiertes Softwareprogramm erkläre ich mich einverstanden.

Trier, ____________________________ ____

(Unterschrift)