Exploring the Population Dynamics of Evolutionary Algorithms using Gene Heritage

DISSERTATION

zur Erlangung des akademischen Grades
Doktoringenieur (Dr.-Ing.)

angenommen durch die Fakultät für Informatik
der Otto-von-Guericke-Universität Magdeburg

von **Tobias Benecke, M.Sc.**
geb. am 28.03.1995 in Wittingen

Gutachterinnen/Gutachter
Prof. Dr-Ing. Sanaz Mostaghim
Prof. Dr. Gabriela Ochoa
Prof. Dr. Hisao Ishibuchi

Magdeburg, den 26.03.2026

Ehrenerklährung

Ich versichere hiermit, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; verwendete fremde und eigene Quellen sind als solche kenntlich gemacht. Insbesondere habe ich nicht die Hilfe eines kommerziellen Promotionsberaters in Anspruch genommen. Dritte haben von mir weder unmittelbar noch mittelbar geldwerte Leistungen für Arbeiten erhalten, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen.

Ich habe insbesondere nicht wissentlich:

- Ergebnisse erfunden oder widersprüchliche Ergebnisse verschwiegen,
- statistische Verfahren absichtlich missbraucht, um Daten in ungerechtfertigter Weise zu interpretieren,
- fremde Ergebnisse oder Veröffentlichungen plagiiert,
- fremde Forschungsergebnisse verzerrt wiedergegeben.

Mir ist bekannt, dass Verstöße gegen das Urheberrecht Unterlassungs- und Schadensersatzansprüche des Urhebers sowie eine strafrechtliche Ahndung durch die Strafverfolgungsbehörden begründen kann. Die Arbeit wurde bisher weder im Inland noch im Ausland in gleicher oder ähnlicher Form als Dissertation eingereicht und ist als Ganzes auch noch nicht veröffentlicht.

Magdeburg, den 13.11.2025

Tobias Benecke

Abstract

The idea of evolutionary algorithms (EAs) is derived from the principle of natural selection in biology. A population of solutions to an optimization problem gets recombined and mutated to create a new generation of offspring. The best survive to the next generation. The idea of this evolutionary loop is simple. However, the resulting search dynamics become very complex, as we usually evolve hundreds or thousands of solutions over many generations. Even though we understand each component in this process, the sheer amount of data quickly becomes incomprehensible for humans. Furthermore, these algorithms are often only tested and evaluated in terms of their performance. While this is an important aspect of the algorithm, we disregard its behavior in the search space. Essentially, only half of the search process is evaluated. This means that we know *that* one algorithm performed better than another, but we do not truly know *why*. This question is typically subject to speculation or educated guesses from the user. Subsequently, designing and tailoring EAs for new problems relies mostly on intuition. We believe a better and proven base of knowledge about the search dynamics can only be beneficial for the development and application of EAs. For this reason, we want to explore their population dynamics in this thesis.

The idea of evaluating the population dynamics of EAs is not new. However, there is a lack of overview of existing approaches, as, to the best of our knowledge, neither review works nor a shared vocabulary around this topic exists. Here, we provide a first starting point by collecting and classifying existing approaches based on the data they collect. We find that in recent years, works visualizing and evaluating the paths of a population through the search space have gained traction. Furthermore, genealogical evaluations, which are also inspired by the field of biology, are often used to evaluate parent-child relationships of the evolutionary process. However, we do only find a small number of works that focus on precisely tracking how each part of a solution was assembled. Here lies the focus of this dissertation.

This thesis proposes a new approach for research into the population dynamics of EAs on a gene-based level. For this, we develop the traceable evolutionary algorithm (T-EA), which allows us to track the heritage of each gene from the initial population throughout the evolutionary process. Furthermore, we propose metrics to evaluate the newly collected data. Most importantly, we can evaluate the influence of a past individual with the amount of genetic material present in the current population. Besides creating new tools to evaluate the population dynamics in EAs, we also want to create first knowledge on how these algorithms assemble their solution. For this, we conduct first exploratory tests to gain a better idea of how the T-EA can be used and get a first interesting insight into the population dynamics of EAs. Our goal is to identify influential individuals in the evolutionary process. For this, we design a system to create individuals of desired genome and fitness properties. Empirically testing both performance and tracing gene heritage in EAs, we can identify which properties, and therefore which individuals, are influential in the search process. We do these tests both for single-objective evolutionary algorithms (SOEAs) and multi-objective evolutionary algorithms (MOEAs). We find that optimizing more than one objective, while only changing the survival process of the evolutionary loop, creates large differences in the convergence behavior.

Zusammenfassung

Die Idee von evolutionären Algorithmen (EAs) leitet sich aus dem Prinzip der natürlichen Selektion ab. Eine Population von Lösungen für ein Optimierungsproblem wird über mehrere Generationen neu kombiniert und mutiert, um neue Lösungen als Nachkommen zu schaffen. Diese neuen Lösungen werden mit einer Fitnessfunktion bewertet. Die Besten werden für die nächste Generation ausgewählt und der Rest wird aussortiert. Die Idee dieser simulierten Evolution ist einfach. Wir können jeden einzelnen Schritt nachvollziehen und wissen genau, wie die neuen Lösungen erstellt werden. Der daraus resultierende Suchprozess wird jedoch schnell komplex und von Menschen nicht mehr nachvollziehbar, da im Laufe aller Generationen zu viele neue Lösungen entstehen. Außerdem werden EAs oft nur anhand der Qualität oder Performance der Lösungen verglichen, die sie gefunden haben. Ihr Verhalten im Suchraum wird oft nicht evaluiert. Damit wird ein wesentlicher Teil des Suchprozesses ignoriert. Dadurch können wir zwar sagen, dass ein Algorithmus bessere Lösungen für ein Problem findet als ein anderer, aber nicht *warum*. Diese Frage nach dem *Warum* wird oft nicht oder lediglich durch Vermutungen und Spekulationen beantwortet. Das bedeutet, dass neue Algorithmen oft nur durch Intuition der Anwender an neue Optimierungsprobleme angepasst werden. Ein besseres Verständnis des Suchverhaltens von EAs ist sowohl für ihre Entwicklung als auch für ihre Anwendung wichtig. Das Ziel dieser Dissertation ist daher, dieses Suchverhalten zu untersuchen.

Dabei ist dieses Forschungsgebiet schon Jahrzehnte alt. Es gibt allerdings bisher keine Reviews, und kein gemeinsames Vokabular zu diesem Thema. Diese Arbeit schafft daher einen ersten Überblick über existierende Ansätze, und sammelt bisherige Erkenntnisse. Wir finden unter anderem Arbeiten, welche den Suchprozess von Algorithmen im Suchraum visualisieren und vergleichbar machen. Oder Ansätze, die ähnlich wie Ahnenforschung in der Biologie Eltern-Kind-Beziehungen erfassen, und Stammbäume erstellen. Allerdings gibt es nur wenige Arbeiten auf der kleinsten Einheit dieses simulierten Evolutionsprozesses, dem Gen. Hier liegt der Schwerpunkt dieser Arbeit.

In dieser Dissertation haben wir einen neuen Algorithmus zum Tracken der Suchdynamik auf der Genebene entwickelt, den tracebaren evolutionären Algorithmus (T-EA). Mit diesem Werkzeug kann die Herkunft jedes Gens von der initialen Population durch den Evolutionsprozess verfolgt werden. Darüber hinaus schlagen wir Metriken zur Auswertung dieser Daten vor. Wir können den Einfluss einer initialen Lösung anhand der Menge des genetischen Materials in der aktuellen Generation bewerten. Mit dem T-EA möchten wir auch erste Untersuchungen der Suchdynamik von EAs erstellen. Dafür führen wir erste explorative Tests durch, um eine bessere Vorstellung davon zu bekommen, wie der T-EA eingesetzt werden kann, und um erste Einblicke in die Populationsdynamik zu gewinnen. Unser Ziel ist es, einflussreiche Individuen im Evolutionsprozess zu identifizieren. Dazu entwerfen wir ein System, um Lösungen mit den gewünschten Genom- und Fitnesseigenschaften zu erstellen. In empirischen Tests messen wir die Performance und tracken die Suchdynamik mithilfe des T-EA, um zu evaluieren, welche Eigenschaften, und damit welche Lösungen, wichtig für den Evolutionsprozess sind. Wir führen diese Tests sowohl für EAs mit einem einzelnen Suchkriterium, als auch für multikriterielle EAs durch. Hier werden die Unterschiede im Suchverhalten für Probleme mit mehr als einem Suchkriterium deutlich.

Contents

Ehrenerklährung iii

Abstract v

Zusammenfassung vii

Contents ix

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Research Goals and Objectives . . . 2
- 1.3 Structure of the Thesis . . . 4

2 Background 7
- 2.1 Optimization Problems . . . 7
- 2.2 Evolutionary Algorithms . . . 7
- 2.3 Multi-Objective Evolutionary Algorithms . . . 10
- 2.4 Benchmark Problems . . . 13
  - 2.4.1 Single-Objective Benchmarks . . . 14
  - 2.4.2 Multi-Objective Benchmarks . . . 14
- 2.5 Evaluating Evolutionary Algorithms . . . 16
  - 2.5.1 Single-Objective EAs . . . 16
  - 2.5.2 Multi-Objective EAs . . . 18
- 2.6 Summary . . . 21

I Algorithmic Advances 23

3 Related Works and State of the Art 25
- 3.1 The Credit Assignment Problem . . . 25
- 3.2 Categories of Population Dynamics . . . 26
- 3.3 Methods for Tracking Population Dynamics . . . 26
  - 3.3.1 Population-Level Approaches . . . 27
  - 3.3.2 Individual-Level Approaches . . . 29
  - 3.3.3 Gene-Level Approaches . . . 32
- 3.4 Population Dynamics in Algorithm Design . . . 33
- 3.5 Summary . . . 35

4 T-EA: The Traceable Evolutionary Algorithm 39
- 4.1 Concept . . . 39
  - 4.1.1 General Idea . . . 39
  - 4.1.2 Evaluation Metrics . . . 40
- 4.2 Trace-Vector Implementation . . . 41
  - 4.2.1 Encoding and Operators . . . 41
  - 4.2.2 Evaluation Metrics . . . 42
- 4.3 Trace-List Implementation . . . 43
  - 4.3.1 Encoding and Operators . . . 43
  - 4.3.2 Evaluation Metrics . . . 45

4.4 T-EA for Multi-Objective Optimization . . . 47
4.5 Summary . . . 47

5 Generating Test Individuals **49**
5.1 Generating Individuals for Single-Objective EAs . . . 49
5.1.1 Sphere Function . . . 50
5.1.2 Rastrigin and Rosenbrock . . . 51
5.2 Generating Individuals for Multi-Objective EAs . . . 52
5.3 Summary . . . 54

II Single-Objective Population Dynamics **55**

6 Related Work **57**
6.1 Population Initialization . . . 57
6.2 Population Dynamics . . . 58
6.2.1 Population Level . . . 58
6.2.2 Individual Level . . . 59
6.2.3 Gene Level . . . 62
6.3 Summary . . . 63

7 Influence of Operators on the Population Dynamics of SOEAs **67**
7.1 Motivation and Overview . . . 67
7.2 Test Setup . . . 68
7.3 Evaluation . . . 68
7.3.1 Single Configuration in Detail . . . 69
7.3.2 Operator Comparison . . . 74
7.4 Summary . . . 78

8 Influential Individuals in SOEAs **81**
8.1 Motivation and Overview . . . 81
8.2 Test Setup . . . 82
8.3 Evaluation . . . 83
8.3.1 Influence of Optimal Genetic Material . . . 83
8.3.2 Differences in Distance to x^{opt} . . . 89
8.4 Summary . . . 92

III Multi-Objective Population Dynamics **95**

9 Related Work **97**
9.1 Population Initialization . . . 97
9.2 Population Dynamics . . . 98
9.2.1 Population Level . . . 98
9.2.2 Individual Level . . . 99
9.2.3 Gene Level . . . 99
9.3 Summary . . . 100

10 Influence of Population Size on the Population Dynamics of MOEAs **101**
10.1 Motivation and Overview . . . 101
10.2 Test Setup . . . 102

10.3 Results . . . 103
10.3.1 Performance . . . 103
10.3.2 Population Dynamics . . . 104
10.4 Summary . . . 109

11 Influential Individuals in MOEAs **111**
11.1 Motivation and Overview . . . 111
11.2 Test Setup . . . 113
11.3 Evaluation . . . 114
11.3.1 Performance . . . 114
11.3.2 Population Dynamics . . . 116
11.3.3 Impact of the Single Seeds Across the PF . . . 119
11.3.4 Impact of Combined Seeds Across the PF . . . 120
11.4 Summary . . . 122

12 Conclusion **125**
12.1 Summary of Contributions . . . 125
12.2 Outlook and Future Work . . . 130

Bibliography **133**

Publications of the Author **145**

Acronyms **147**

Symbols **149**

List of Figures **153**

List of Tables **155**

IV Appendix **157**

A Code and Data **159**

B Supplementary Material for Chapter 8 **161**

1 Introduction

1.1 Motivation 1
1.2 Research Goals 2
1.3 Structure of the Thesis . . . 4

1.1 Motivation

Evolutionary algorithms (EAs) are metaheuristic optimization algorithms, inspired by biological evolution. They are established in their use for computationally expensive and/or NP-hard problems as an efficient alternative when an exhaustive search is not feasible. These algorithms have been applied across multiple fields and problems, examples of which include the medical sector [1, 2], the energy sector [3], factory layout planning [4], supply chain management [5, 6], agriculture [7], and vehicle routing [8, 9], just to highlight a few.

One strength of EAs is their modularity. They consist of a few components, each influencing the search process and allowing the user to tailor the algorithm to best fit the optimization problem at hand. A basic overview of these components and the process of an EA can be found in Figure 1.1. They evolve a population of solutions, of which parents are selected for recombination (crossover and mutation) to create new offspring. These are evaluated with one or multiple fitness functions that determine their quality. A selection scheme decides who passes to the next generation, and so the evolutionary circle continues until a stopping criterion is met. As can be seen, the general concept of these algorithms is easy to describe and easy to understand. The same holds for each component of the algorithm. We know exactly how the crossover and mutation operators, or the selection schemes work. However, these algorithms typically evolve hundreds of individuals over hundreds or thousands of generations, resulting in a very complex search dynamic, which becomes an incomprehensible black-box for human observers.

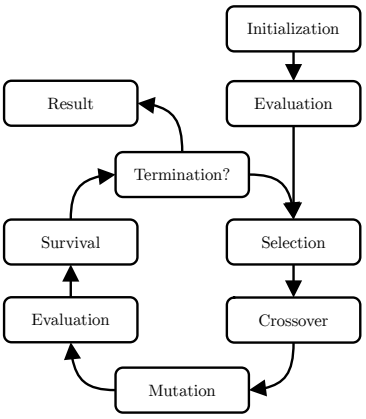


Figure 1.1: Basic process of an EA.

The field of explainable artificial intelligences (XAIs) is usually associated first with neuronal networks (NNs). Here, the main challenge is to answer *why* a NN has reached its output, given an input [10, 11].EAs are search algorithms featuring a different challenge. The output of the evolutionary process is usually interpretable. For example, in path-planning applications, we can understand and interpret the route found by the algorithms [8]. The unknown area here is the question of *how* the EA build these solutions, as the underlying search dynamics quickly become too large and complex to be comprehensible. One reason for this is the large number of solutions which are created over the course of the evolutionary process. Another is the modularity of these algorithms, which adds further complexity, as each component, or combination of components, affects the search dynamics. The problem to be solved, and its fitness landscape, also influences how EAs find their solutions. This is also true for the number of objectives. One strength of EAs is their ability to optimize multiple conflicting objectives at once, without the need to specify preferences pre-optimization. However, this leads to the algorithms converging towards multiple optimal solutions instead of one singular optimum, changing their search behavior.

In our evolutionary computation (EC) community, this question of *how* is often neglected. Usually, when evaluating and comparing algorithms, the focus lies on the performance. While we know that one algorithmic configuration performed better than another, the *why* is either unknown or subject to speculation. This critique on how we evaluate and compare EAs is decades old [12]. The same is true for research on how to evaluate the search dynamics of these algorithms [13]. Works in this area have also found inspiration in the field of biology. Here, the study of genealogy exists, tracing the descendants of individuals over generations. A well known example for this is ancestry trees for humans. While these techniques allow for greater insight into the search dynamics of EAs, they also suffer from some information loss, as it is not always possible to reconstruct which genetic features stem from which parent. This leads to a problem of credit assignment [14, 15].

In an older work on different types of EAs for parameter optimization, Bäck and Schwefel [16] note: "It is curious enough to see very different, sometimes contrasting design principles for evolutionary algorithms being emphasized by the different research communities. A clear goal of future research should be to identify the rational as well as not so rational reasons for this fact and to extract the general rules for designing components of new and maybe even better EAs." What they mean is different types of EAs emphasize different components and features, yet still all work well for optimization tasks. While this quote stems from 1993, it remains valid to this day. Research in the field of evaluating the search dynamics of EAs remains a niche, the application of such approaches is still rare, and competitive performance testing remains the norm for most works in our area. We believe this to be an important gap to be filled, as a better understanding of how and why some algorithmic configurations perform better than others is valuable both for the application of EAs, as well as for future algorithmic developments.

With this thesis, we aim to illuminate parts of this black-box that is the search process of EAs. More specifically, we are interested in better understanding how these algorithms combine the individuals of the initial population throughout their evolutionary process to reach their results. For this, new tools to track data in the evolutionary process are introduced, as well as a framework to test specific algorithmic configurations. The next section introduces these research goals in detail, specifying research questions on which these aims can be quantified.

1.2 Research Goals and Objectives

The goal of this dissertation is to gain a better understanding of the underlying search dynamics of EAs. We believe this to be an important research area, as a better understanding is vital for the future development and application of our algorithms. The search dynamics of EAs is a large area with many facets. Our specific objective is to assess how these algorithms use the individuals of the initial population to reach their results. For this, we need a tool to precisely track gene heritage without information loss, or problems of credit assignment. To be of significant use for our community, it needs to be scalable in the number of objectives and decision variables, and support all common representations and

operators used with EAs. The development of this tool, which we call the traceable evolutionary algorithm (T-EA), is one central goal of this dissertation.

Furthermore, we want to use the T-EA to create a solid base of understanding about the population dynamics in EAs, beyond the educated guesses derived from performance evaluations. For this, we need to design and test how individuals of different properties influence the population dynamics. Generating such individuals of desired properties, like a target fitness or specific genome composition, will be important to test basic hypotheses about how and what genetic material is used by the algorithms. This is summarized in the four research goals of this dissertation, providing the broad outline of this work:

- G 1 Classify existing approaches for analyzing the search dynamics of EAs.
- G 2 Create a tool to precisely track gene heritage without information loss.
- G 3 Create a system to evaluate individuals of specific fitness and genome configuration.
- G 4 Gain a knowledge base on the population dynamics of EAs.

For the first goal, we believe it is vital to gain an overview about existing techniques to evaluate the search dynamics of EAs, classifying them by their strengths and weaknesses, and to gain an overview of the knowledge gained over the past years:

- RQ 1 Which techniques exist to evaluate the search dynamics on EAs?
  - RQ 1.1 On which factors can we classify these approaches?
  - RQ 1.2 What knowledge about the search dynamics of EAs has been gained so far?

Reviewing the literature reveals a gap in this area: Currently, no technique exists to track the heritage information of the initial population for without loss of information for all types of algorithms and operators. This leads to a problem of credit assignment. As our work is specifically focused on this question, one central goal is developing a more precise approach which allows tracking genetic heritage in EAs. This approach should support all common representations and operators, and also be applicable to multi-objective evolutionary algorithms (MOEAs). Besides this, an important aspect is the evaluation of this data.

- RQ 2 How can we precisely track genetic heritage without information loss and precise credit assignment?
  - RQ 2.1 How can we trace heritage in EAs?
  - RQ 2.2 How to adapt the approach to all typical genome representations and operators of genetic algorithms?
  - RQ 2.3 How can we apply this to multi-objective optimization?
  - RQ 2.4 What methods can be used to calculate genetic influence for the wide variety of operators and algorithms?

To reach our overall goal towards a better understanding of the search dynamics on EAs, we need to design tests on how different individuals influence the evolutionary process. For this reason, the third research question is focused on this topic.

RQ 3 How can we create individuals of specific properties to test their influence on the evolutionary process?

RQ 3.1 How can we design a system to create individuals of desired fitness and genome targets for single-objective evolutionary algorithms (SOEAs)?

RQ 3.2 Can we create similar individuals for MOEAs?

Furthermore, we aim to create a knowledge base from which this research area can grow further. As a starting point, we want to identify which individuals in the initial population are influential in the search process of EAs. Due to their inherent differences, we split this between single- and multi-objective EAs.

The fourth research goal aims at identifying influential individuals in the search process of SOEAs. For this, we first want to test and discuss the capabilities of the T-EA. This is done with a first proof-of-concept evaluation on different operators commonly used for these algorithms. The main goal is tested by generating test individuals of specific properties, the creation of which is our third research question, and tracking and evaluating their influence throughout the evolutionary process.

RQ 4 Can we identify influential individuals in the search process of SOEAs?

RQ 4.1 How can we use the T-EA to evaluate the population dynamics of SOEAs?

RQ 4.2 How do the proposed impact metrics differ?

Finally, we also wand to identify influential individuals in the search process of MOEAs. For this, the differences in convergence to SOEAs is discussed. We also wand to explore the use of the T-EA on MOEAs. This is again done with a proof-of-concept evaluation, this time on the population size parameter.

RQ 5 Can we identify influential individuals in the search process of MOEAs?

RQ 5.1 How do the population dynamics of MOEAs differ to SOEAs?

RQ 5.2 How can the T-EA be used to evaluate the population dynamics of MOEAs?

1.3 Structure of the Thesis

Background

After this introduction, the scientific fundamentals for the thesis are laid out in Chapter 2. Terminology around optimization and evolutionary algorithms are discussed, and their most important concepts for this thesis are highlighted. The remainder of this thesis is split into three parts.

Part I - Algorithmic Advances

Part I is concerned with algorithms and methods for research into the search dynamics of EAs. This part features three chapters. First, Chapter 3 describes the related work and provides an overview of existing approaches in this research area (RQ 1). Following this, Chapter 4 introduces the traceable evolutionary algorithm (T-EA), which is the foundational heritage tracking approach developed for this thesis, directly

linked to Research Goal 2. The general gene tracing approach is presented, both in its early version (RQ 2.2) and its extension for all common crossover and mutation operators (RQ 2.3). Furthermore, the evaluation metrics around this heritage tracking process are described (RQ 2.4). The chapter is closed with an evaluation on the use of the approach with MOEAs (RQ 2.3). Finally, Chapter 5 presents a system to generate individuals for the evaluation, both for the single- and multi-objective EAs (RQs 3.1 and 3.2).

Part II - SOEA Population Dynamics

After this, the following parts focus on the population dynamics in EAs. Due to the differences in their nature, this is done separately for single- and multi-objective EAs. Part II centres around SOEAs. First, Chapter 6 provides a literature overview of the current results found in this area (RQ 1.2). After this, a proof of concept on how the T-EA can be used is provided in Chapter 7 (RQ 4.1). Here, the effects of different crossover and mutation operators on the population dynamics are discussed, further showing the use of the T-EA and the proposed evaluation metrics (RQ 4.2). The final chapter of this part (Chapter 8) explores what makes an individual influential in the search process of EAs (RQ 4), closing the discussion on SOEAs in this thesis.

Part III - MOEA Population Dynamics

Finally, Part III focuses on the population dynamics of MOEAs. Similar to the previous part on SOEAs, we provide an overview of the findings for the population dynamics in the field of MOEAs in Chapter 9 (RQ 1.2). A first test on the effects of different population sizes on the population dynamics in Chapter 10 serves as a first proof-of-concept for the use of the T-EA for multi-objective optimization problems (MOPs) (RQ 5.1). Finally, Chapter 11 explores the idea of influential individuals in MOEAs (RQ 5), discussing the differences between single- and multi-objective EAs in the process (RQ 5.2).

Conclusion

Concluding this thesis, Chapter 12 provides an overview of the research contributions and findings in this work, followed by an outlook on future research directions in this area.

2 Background

This chapter presents an overview of the most important concepts and algorithms used in this thesis. First, the concepts of optimization problems and EAs are introduced, and the terminology around them is defined. This is followed by a discussion on multi-objective evolutionary algorithms (MOEAs), defining the most important concepts, gaining an overview of the most popular algorithms, and discussing the most important differences to their single-objective counterparts. Finally, this section is closed by presenting the most common ways of how EAs are evaluated, first by discussing benchmark problems, and briefly presenting typical performance evaluations.

2.1 Optimization Problems . 7
2.2 Evolutionary Algorithms 7
2.3 Multi-Objective EAs . . . 10
2.4 Benchmark Problems . . 13
2.4.1 SOEA Benchmarks 14
2.4.2 MOEA Benchmarks 14
2.5 Evaluating EAs 16
2.5.1 Single-Objective EAs . . . 16
2.5.2 Multi-Objective EAs . . . 18
2.6 Summary 21

2.1 Optimization Problems

In this work, optimization problems are formally defined as:

$$\text{minimize} \quad \vec{f}(\vec{x}) = (f_1(\vec{x}), \ldots, f_o(\vec{x})) \text{ Subject to } \vec{x} \in \mathcal{S}^{\mathfrak{s}} \tag{2.1}$$

Terminology

Here, $\mathcal{S}$ denotes the $\mathfrak{s}$-dimensional search space, also called decision space. In this work, solutions to an optimization problem are denoted as $\vec{x}$, with each element of this vector defining one decision variable. The encoding of the decision variables depends on the optimization problem used[1]. In this thesis, we will use problems with a continuous real-valued encoding of the search space. The objective functions $\vec{f}$ transform the search space representation into the o-dimensional objective space $\mathcal{O}$. In the context of optimization problems, $\vec{f}(\vec{x})$ is a vector of functions $f_i(\vec{x})$, with $1 \leq i \leq o$, assigning how well a solution $\vec{x}$ is performing for the given objective f_i. If the optimization problem only features one objective, $f(\vec{x})$ is also denoted without an index. To solve the optimization problem, the output of these objective functions is either to be maximized or to be minimized. Without loss of generality, we will assume minimization in this work. Finally, all optimization problems used are unconstrained.

2.2 Evolutionary Algorithms

Background on EAs

EAs are bio-inspired optimization algorithms, modeling the process of natural selection to solve optimization problems. They are metaheuristics, belonging to the field of EC, and can be described as a sort of intelligent search algorithm. These algorithms are used for optimization problems, which are too large (NP-hard) or too computationally complex to be solved exactly. While EAs aim for the global optimum, they usually find sufficiently close solutions in a reasonable amount of time. Multiple subcategories of EAs exist, such as genetic algorithms (GAs) [17], genetic

[1] In this thesis, $\vec{x}$ is always a vector, however, generally speaking, solutions can also be represented differently.

programming (GP) [18], evolutionary strategys (ESs) [19], evolutionary programming (EP) [20], or differential evolution (DE) [21]. They all share the same Darwinian principle of survival of the fittest, but differ in some areas, mainly in the representation of solutions, and how new solutions are created. At the end of this section, the differences to the other algorithm types is briefly discussed.

Terminology

EAs evolve a set of solutions to an optimization problem, which are denoted as $X = \{\vec{x}_1, \ldots, \vec{x}_n\}$. This set X is also called the *population*, with the number of individuals n also referred to as the *population size*. The solutions to these optimization problems are also referred to as *individuals*, and are represented in two spaces. The search space representation $\vec{x}_i = [x_{i,1}, \ldots, x_{i,s}]$ is referred to as the *genome* of an individual. In this thesis, the genome of an individual $\vec{x}_i$ is always a vector, with each of the $x_{i,j}$ decision variables also being called the *genes*. The objective space representation $\vec{f}(\vec{x}_i)$ is called the *fitness* of an individual.

The Evolutionary Process

The general evolutionary procedure of an EA is depicted in Figure 2.1. First, the algorithm is initialized by generating an initial population X^0. Before starting the evolutionary circle, the fitness of each individual is assessed through the fitness function(s). Over multiple generations, the current population is recombined and mutated to create a new population of offspring individuals. This process consists of these five general parts:

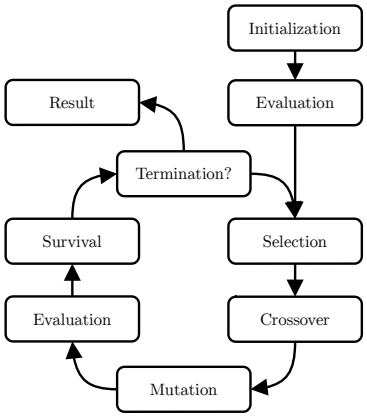


Figure 2.1: Basic process of an EA.

1. Selection: Selecting parent individuals from the current population X^i for offspring creation.
2. Recombination: Combining the genomes of the selected parents to create new offspring individuals.
3. Mutation: Introducing random changes to the genome of the offspring.
4. Evaluation: Assessing the fitness of the offspring individuals.
5. Survival: Selecting the surviving individuals from the parent and offspring populations.

Initialization

The initial population is an important part of an EA, as it is the base on which the evolutionary process builds upon. It is this genetic material, which is recombined over multiple generations, leading the algorithm to its result. While the mutation operator can introduce new genetic material through random changes, the initial population remains the foundation of the evolutionary process and plays a crucial role in its success. Most commonly, EAs are initialized randomly, meaning each decision variable independently gets a random value assigned to it. More elaborate strategies on how to sample the initial population, of course, also exist. A popular example for this is Latin-hypercube sampling (LHS) [22]. A more detailed discussion on initialization strategies in the context of population dynamics can be found in the later Section 6.1 for SOEAs, and Section 9.1 for MOEAs.

Mating Selection

In the first step after the initialization, individuals are selected for offspring creation. Approaches try to find a balance between promoting the good fitness individuals, without being too greedy and risking local convergence. A popular example of a mating selection approach is roulette wheel/fitness proportionate selection [17], in which the probability to be selected for offspring creation is directly proportional to the fitness of the individuals. This way, good fitness individuals have a higher chance

of being selected. Probably the most used operator in the literature of EAs is tournament selection[2]. For each individual needed for offspring creation, a tournament between randomly selected individuals from the parent population is held. The winner of the tournament is the individual with the best fitness, getting selected as the parent for mating. The *tournament size* determines the selective pressure towards better fitness individuals.

Crossover

In crossover, the genetic material of the previously selected individuals gets recombined. Figure 2.2 shows a practical example of a one-point crossover operation on two binary encoded individuals. The genome of both individuals gets cut at a random point, and their two halves get reassembled into two new individuals. Of course, many different strategies for crossover exist. Another popular example is uniform crossover (UX) [24], in which a random decision is made separately for every gene, determining if it should be swapped. For real-valued numerical problems, two gene values can also be combined to create a new one. The simplest example for this is a linear recombination crossover, where both parents are weighted. The most popular of these recombining operators is simulated binary crossover (SBX) [25].

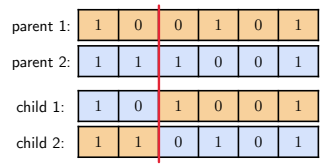


Figure 2.2: Example of one-point crossover.

Mutation

Mutation, in EAs, is usually applied to the offspring generated from the crossover operation. It introduces new genetic material by randomly changing values in the genome. Typically, mutation is applied using a *mutation probability*, determining for each gene if it should be mutated or not. This is an important parameter in EAs, as it determines how much new genetic material is introduced. The optimal probability is dependent on many factors, like the algorithm, operators used, or optimization problem at hand. The most usual rule of thumb used in literature is to mutate each offspring individual on average once, so to set the mutation probability to $\frac{1}{n}$. Of course, different mutation operators exist. Figure 2.3 shows an example for a bit-flip mutation on a binary encoded individual. The operator flips the selected gene to its opposite value. An equivalent for integer or real-value encoded individuals would be uniform mutation, setting the gene to a new random number. The most popular operator in literature is polynomial mutation (PM) [26], which alters a gene based on its original value using a probability distribution.

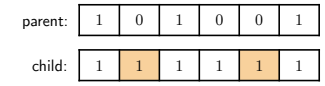


Figure 2.3: Example of bitflip mutation.

Survival Selection

After offspring creation, the individuals from the offspring and parent populations are selected for the next generation. Here lies the biggest difference in single- and multi-objective EAs. For SOEAs, two main strategies exist. We assume a parent population μ and an offspring population λ. Then the first strategy can simply be to select the best individuals both from both the parent and offspring populations. This is also called $(\mu+\lambda)$ EA. It keeps the best individuals, but might come at the disadvantage of being too greedy, losing genetic diversity, and converging locally. The other common strategy is to select the next generation only from the offspring population, referred to as a (μ, λ) EA. Here, it is also common to keep a number of *elite* (best) individuals from the parent's population, to not lose the best individuals found so far. Most MOEAs use the $(\mu + \lambda)$ approach. Here, the algorithm tries to find a diverse set of solutions for all objectives, meaning the population is not just the

[2] Tournament selection is attributed to an unpublished work by Wetzel [23], a detailed description of the operator can be found in [23].

genetic base for the next round of offspring creation. The next section will explain the challenges and approaches of MOEAs in more detail.

Termination

The termination criterion determines the end of the evolutionary process. In scientific literature, termination is usually bound by a set number of generations, or set number of *function evaluations*. The term "function evaluations" describes the number of fitness evaluations done by the algorithm. It is the standard for measuring the algorithm's computational cost, as this is typically the computationally most expensive part in the evolutionary loop, especially for real-world applications, which often require expensive simulations. Keeping the same number of function evaluations means evaluating the same number of solutions, and allowing for a fair performance comparison independent of the computational power of the hardware used to run the tests. For this reason, it is an important factor for a fair performance comparison of EAs [27]. Dynamic stopping criteria, such as stopping if the algorithm did not find better solutions over a period of time, also exist.

Types of EAs

There are many different types of EAs, as each component in the evolutionary loop can be adjusted to fit the problem at hand. Popular types include genetic algorithm (GA) [17], evolutionary strategy (ES) [19, 28], genetic programming (GP) [18], or differential evolution (DE) [21]. They all share a similar evolutionary principle but are aimed at different tasks, leading to some differences, mainly in the representation of solutions and offspring creation. These differences also lead to unique strengths and challenges for the algorithms, importantly for this thesis also in their population dynamics. GAs, for example, are typically represented as a binary encoded vector, mostly aimed at discrete or combinatorial problems. They generate new individuals mainly through recombination, after which mutation is applied. ES, on the other hand, are mainly aimed at numerical problems and are usually encoded as real-valued vectors. Their emphasis lies more on mutation and self-adaptation, but they also feature recombination in a second step. Solutions in GP are usually represented as a mathematical tree, aimed at various problems, such as symbolic regression, program synthesis, or classification. DE is also aimed at numerical problems, but uses differences between the individuals of a population in their offspring creation process. In this thesis, we focus on algorithms sharing the structure as described for GAs, but for real-valued problems. In the literature, these are often referred to as "EAs" (not as an umbrella term, but as a standalone algorithm type) and occasionally as "real-valued GAs". Throughout this work, we will use the term EAs to describe this class of algorithm.

Single- vs. Multi-Objective EAs

When optimizing problems with a single objective, EAs converge towards one single optimal solution. The quality of two individuals can directly be compared by their fitness value. However, for problems with multiple, conflicting objectives, there exists not one optimal solution, but a set of so-called Pareto-optimal solutions. The next section presents how EAs are adapted to solve MOPs.

2.3 Multi-Objective Evolutionary Algorithms

Real-world problems often feature multiple conflicting objectives. EAs are a popular choice of optimization algorithm for such problems. As a

population-based algorithm, they can search multiple areas of a search space simultaneously. This allows EAs to optimize multiple objectives at once, without the need to specify preferences or weigh objectives before the optimization process. While structurally very similar, the optimization of multiple objectives at the same time also presents distinct challenges compared to the single-objective case.

Pareto-dominance and Non-Dominated Solutions (NDS)

Every solution in a MOEA has multiple fitness values, so comparing and ordering solutions directly on their fitness quality is not possible. Ranking and comparing the quality of solutions against each other, however, is a crucial part of the evolutionary process. To compare the quality of solutions for multiple objectives, the concept of Pareto-dominance is introduced. We can define the dominance relationship between two individuals as:

Definition 2.3.1 (Pareto-dominance) *Given two solutions $\vec{x}_1, \vec{x}_2 \in S$ for an optimization problem $\vec{f}$, $\vec{x}_1$ is dominating $\vec{x}_2$ if and only if there exists one objective where $\vec{x}_1$ has a better (lower) fitness than $\vec{x}_2$ and all other objectives have at least equal or better fitness:*

$$\vec{x}_1 \prec \vec{x}_2 \Leftrightarrow \exists f_i \in \vec{f} : f_i(\vec{x}_1) < f_i(\vec{x}_2) \wedge \forall f_i \in \vec{f} : f_i(\vec{x}_1) \leq f_i(\vec{x}_2) \quad (2.2)$$

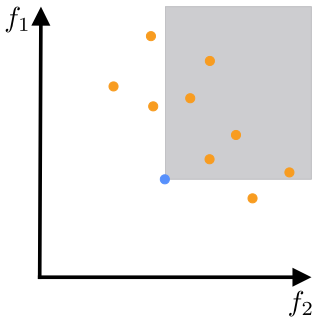


Figure 2.4: Visual example of Pareto-dominance.

Through the dominance relationship, we can compare the performance of two individuals for more than one objective. If $\vec{x}_1 \prec \vec{x}_2$, then $\vec{x}_1$ can be considered better, as it is better in one objective and at least equal in all others. A visual example of this can be found in Figure 2.4. All solutions in the area of the rectangle are dominated by the blue solution. If neither $\vec{x}_1 \prec \vec{x}_2$, nor $\vec{x}_2 \prec \vec{x}_1$, then both solutions are considered to be of equal quality.

Solutions which are not dominated by any other solution of the population are called non-dominated solutions (NDSs). These NDSs mark the best solutions found by the algorithm so far. They are all considered to be of equal quality, as improving the performance in one objective would decrease the performance in another objective. We formally define them as:

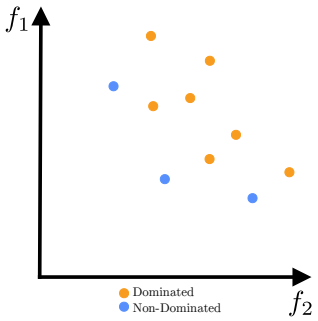


Figure 2.5: Visual example of non-dominated solution (NDS).

Definition 2.3.2 (Non-dominated Solutions) *In the context of EAs, we call a solution $\vec{x}_i \in X$ non-dominated, if there exists no other solution in the population X which dominates it:*

$$\nexists \vec{x}_j \in X : x_j \prec x_i \quad (2.3)$$

Pareto-Optimality

In MOEAs, there exist multiple optimal solutions for a given problem. These are also called Pareto-optimal, and are denoted as $\vec{x}_i^{opt}$. We can define Pareto-optimal solutions as solutions x_i for which no other solution x_j exists which dominates it:

Definition 2.3.3 (Pareto-optimal) *A solution $\vec{x}_i$ is called Pareto-optimal if there is no other solution $\vec{x}_j \in S$ in the search space which dominates it:*

$$\vec{x}_i =: \vec{x}_i^{opt} \Longleftrightarrow \nexists \vec{x}_j : \vec{x}_j \prec \vec{x}_i | \vec{x}_i, \vec{x}_j \in S \quad (2.4)$$

Pareto-front (PF) and Pareto-set (PS)

The goal of MOEAs is to find these Pareto-optimal solutions. These sets of optimal solutions are referred to as the Pareto-front (PF) in the objective space and the Pareto-set (PS) in the search space. Formally, they are defined as follows:

Definition 2.3.4 (Pareto-set) *The Pareto-set (PS) of an optimization problem is the set of all Pareto-optimal solutions for that problem in the search space:*

$$PS := \{\vec{x}_i^{opt} | \vec{x}_i^{opt} \text{ is pareto-optimal}\} \tag{2.5}$$

Definition 2.3.5 (Pareto-front) *The Pareto-front (PF) of an optimization problem is the corresponding objective-space representation of its Pareto-set (PS):*

$$PF := \{\vec{f}(\vec{x}_i^{opt}) | \vec{x}_i^{opt} \in PS\} \tag{2.6}$$

In practical terms, the PS is the set of all Pareto-optimal solutions for an optimization problem. The Pareto-front (PF) is the corresponding search space representation of the PS. This is visualized in Figure 2.6, showing the discussed concepts for an exemplary bi-objective optimization problem with two decision variables. On the left side, the PS is shown in the search space, and on the right, the corresponding PF in the objective space. The Pareto-optimal solutions lie directly in the PS/PF.

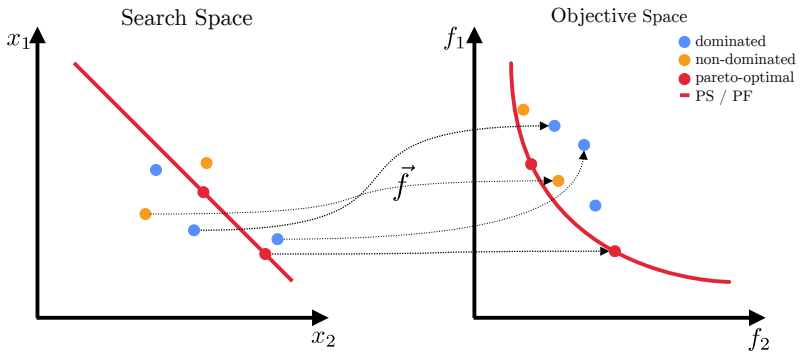


Figure 2.6: Exemplary PS and PF for a bi-objective optimization problem with two decision variables.

Differences to SOEAs

In principle, MOEAs use the same evolutionary loop and largely the same components as their single-objective counterpart. As mentioned in the previous section, the main difference lies in the survival strategy. A SOEA only converges to one singular optimum. This means it only needs to retain one best individual, while the rest of the surviving population can largely be seen as the genetic material that builds the next generation of offspring. A MOEA, on the other hand, aims to hold multiple best solutions in its population. This is reflected in the survival strategy. Most commonly, MOEAs are $(\mu + \lambda)$ algorithms, which can be categorized into three main types, namely *indicator*-based, *dominance*-based and *decomposition*-based. The main difference between these types is how they rank and select the solutions which survive to the next generation.

Indicator-Based MOEAs

Indicator-based MOEAs try to optimize the performance of a population as a whole, using performance indicators [29]. In short, performance indicators are metrics which try to translate the performance of a population into one singular, comparable value. More details on this can be found in the later Section 2.5.2. In the survival phase, individuals who contribute the least towards these performance indicators are removed.

Dominance-Based MOEAs

Dominance-based MOEAs select the individuals separately using, as the name implies, the principle of Pareto-dominance to rank the quality of solutions. Several algorithms exist in this category. We especially want to highlight the non-dominated sorting genetic algorithm II (NSGA-II) [30]. This algorithm ranks individuals based on how many other individuals dominate them. Then the new population is successively filled with NDS, once-dominated solutions, and so on, until the population size is reached. In case there are more individuals in one category than spots left in the new population, the *crowding distance* metric is used to select distant individuals, promoting the population's coverage of the PF. NSGA-II [30] is among the most used and most influential MOEAs of all time, and for this reason, a natural starting point for research into the population dynamics of MOEAs.

Decomposition-Based MOEAs

Finally, decomposition-based MOEAs decompose the objective space of a MOP into multiple single-objective sub-problems, usually through a set of reference vectors. Here, we want to highlight the multi-objective evolutionary algorithm based on decomposition (MOEA/D) [31], which, besides NSGA-II, is one of the most important and most used algorithms in our field. MOEA/Ds, uses a set of reference vectors to decompose the objective space. In the survival phase, the reference vectors get assigned individuals from the population, which perform best on the respective single-objective optimization problem (SOP) they represent. This way, the number of reference vectors also dictates the population size. Additionally, for mating selection, restrictions can be applied so that only individuals of neighboring reference vectors are allowed to mate. Over the years, many variants of MOEA/D have been proposed. The most common components to be altered are the decomposition method used, the weight vector generation method, and adjusting the evolutionary operators [32]. In this thesis, we focus on the initial version by Zhang and Li [31] as the starting point for population dynamics research.

2.4 Benchmark Problems

The goal of EAs is to find good solutions to real-world problems. These, however, are often computationally expensive, and extensive testing and algorithm tuning are unrealistic. Usually, algorithm designers rely on test problems to evaluate the performance and properties of their proposed algorithms. These problems are typically fast-to-compute mathematical functions. Often, they are combined into a set of problems, which we call a benchmark suite.

Desirable Properties of Benchmarks

Bartz-Beielstein et al. [33] formulate four desirable properties of benchmark suites. With no intended ranking, these are: First, they should be *diverse*, with a wide range of problems featuring different properties and characteristics. Second, they should be *representative* of difficulties and challenges found in the real world. Third, they should be *scalable and tunable*, for example, in the number of decision variables, number of objectives, level of dependence between decision variables, or multi-modality. Fourth, they should feature a *known optimum*, or at least a best-known performance.

2.4.1 Single-Objective Benchmarks

The bbob Benchmark Suite

A benchmarking set satisfying the desirable properties of Bartz-Beielstein et al. [33] is the bbob test suite [34]. It features 24 well-known test functions scalable in their search space dimensionality. These functions are sorted into four categories, namely separable functions, functions with low or moderate conditioning, unimodal functions with high conditioning, and multi-modal functions with weak global structure. All functions feature known optima, which are randomly shifted in the objective space, creating *function instances*. The purpose of this is to test the robustness of algorithms. The bbob functions are a widely respected and used benchmark set for SOEAs. An implementation can be found in the COCO framework [35]. In the following, we want to highlight three test functions, which are also found in the bbob benchmarks [34], that are used throughout this thesis.

Sphere Function

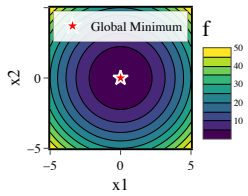


Figure 2.7: Fitness landscape of the Sphere function.

The Sphere function (f_1 in [34]) is probably the easiest and most understandable test function. It can be seen in Equation 2.7. It is separable and unimodal. For this thesis, it has another desirable property, as it is one of the few functions for which a genome can easily be calculated for a given target fitness. This allows us to create individuals with a specific fitness and genome (see Chapter 5), and test their properties (see Chapter 5). The fitness landscape can also be seen as a contour plot in Figure 2.7. The optimal fitness is known as $f^{opt}_{Sphere} = 0$ and $\vec{x}^{opt}_i = [0, \ldots, 0]$.

$$f_{Sphere}(\vec{x}_i) = \sum_{j=1}^{n} x_{i,j}^2 \tag{2.7}$$

Rastrigin Function

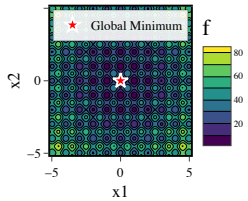


Figure 2.8: Fitness landscape of the Rastrigin function.

The Rastrigin function [36] is used as described in Equation 2.8. The test function is separable and highly multimodal, featuring multiple local minima. The parameter A controls the amplitude between the local optima. Commonly, $A = 10$ is chosen, which is also used throughout this thesis. A contour plot of the fitness landscape can be seen in Figure 2.8. The optimum is known at $f^{opt}_{Rastrigin} = 0$ and $\vec{x}^{opt}_i = [0, \ldots, 0]$.

$$f_{Rastrigin}(\vec{x}_i) = A \cdot \mathfrak{d} + \sum_{i=1}^{\mathfrak{d}} \left[x_{i,j}^2 - A \cos\left(2\pi x_{i,j}\right) \right], \; A = 10 \tag{2.8}$$

Rosenbrock Function

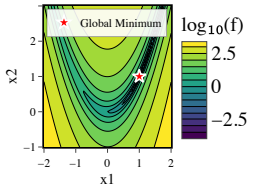


Figure 2.9: Fitness landscape of the Rosenbrock function.

Finally, the Rosenbrock function [37] is used as seen in Equation 2.9. In contrast to the Sphere and Rastrigin functions, it is not separable. It is also called the "banana function", as it features a banana-shaped valley, as can also be seen in the contour plot of its fitness landscape in Figure 2.9. The optimum is also known at $f^{opt}_{Rosenbrock} = 0$ and $\vec{x}^{opt}_i = [1, \ldots, 1]$.

$$f_{Rosenbrock}(\vec{x}_i) = \sum_{j=1}^{\mathfrak{d}-1} \left[100 \left(x_{i,j+1} - x_{i,j}^2 \right)^2 + \left(1 - x_{i,j} \right)^2 \right] \tag{2.9}$$

2.4.2 Multi-Objective Benchmarks

Early Benchmark Problems

The construction of MOPs as benchmarks is much harder, as they feature multiple conflicting objectives. Early benchmarks, like the ZDT [38] or

DTLZ [39] problems, are constructed around desirable properties of the PF in the objective space. Two examples of these properties are the shape of the PF to be linear, triangular or disconnected, or the curvature of the PF to be linear, convex or concave [27]. These problems are an important addition to the field of MOEAs. However, due to their construction technique, they are also criticised for unrealistic properties, like the regularity of their PF shapes, or their PS properties [40–45]. Despite their age and criticism, they are still frequently used, even in recent works.

"Easy" and "Complicated" PSs

Especially relevant in this thesis are PS properties. Li and Zhang [41] categorize benchmarks into ones with *easy* and ones with *complicated* PSs. As mentioned, older test problems in the field of MOEAs include the ZDT [38], DTLZ [39] or WFG [46] benchmark suites. Due to their construction around the desired PF, they feature what is also referred to as *positional* and *distance* genes [47, 48]. As their names imply, a distance gene controls only the distance of an individual to the PF, and usually converges to the same value [47]. Positional genes, on the other hand, only control the position of the individual on the PF [47]. As an example, in the PS of every three-dimensional DTLZ [39] test problem, all but two genes do not have a single optimal value:

$$0 \leq x_1^{opt}, x_2^{opt} \leq 1,\ x_3^{opt} = x_4^{opt} = \ldots = x_3^{opt} = 0.5 \tag{2.10}$$

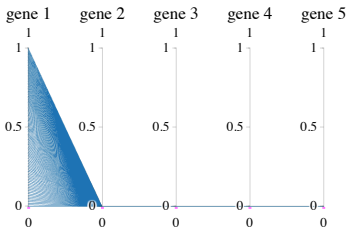


Figure 2.10: Genome variable distribution for 100 samples of the ZDT 1 PS.

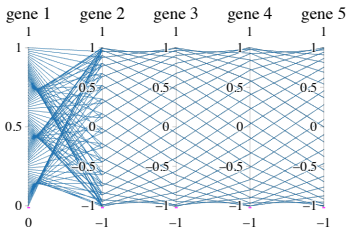


Figure 2.11: Genome variable distribution for 100 samples of the UF 1 PS.

The PS of both the ZDT [38] and WFG [46] benchmarks have similar features. A visual example of the genome variable distribution for the ZDT 1 problem can also be seen in Figure 2.10. For this reason, Li and Zhang [41] categorizes these problems as ones with *easy* PSs. It is long argued that real-world problems do not feature such properties and require the algorithm to find a range of values for each decision variable[40, 41]. Problems with *complex* PSs, on the other hand, do not feature these positional and distance genes. Here, a range of values can be optimal for every gene. Synthetic benchmarks with complex PSs have also been proposed, like the LZ [41] or UF [49] benchmarking suites. In both, all possible values of a gene can be optimal. The genome variable distribution of a sample of the PS of the UF 1 problem can be seen in Figure 2.11. The main challenge for these problems is to find the right combination of gene values instead of the right value of a gene. This differentiation becomes relevant when evaluating the population dynamics on these types of test problems, especially for studies on population initialization, where even very recent studies still use benchmarks with easy PSs.

The Difficulties of MOP Benchmarks

Designing multi-objective benchmarks proves difficult. All artificially created benchmark sets, like ZDT [38], DTLZ [39], WFG [46], including ones with complicated PSs, like the LZ [41], UF [49] problems, are criticised for featuring artificial characteristics, for example, to have their PS on the domain boundary, or feature a large share of separable functions [44]. More recent approaches fall into two categories: Real-world inspired test problems and the combination of single-objective functions to create multi-objective problems.

Real-World Inspired Benchmarks

Several novel benchmarks for MOEAs have been proposed over the last years. Examples include the multi-agent coordination problem (MACO) [50], modelling the challenges coordinating multiple agents through a narrow gap with the objectives of energy efficiency and safety. Or the circular supply-chain (CSC) problem [5, 51], modelling the challenges

of supply chain optimization towards a circular economy, considering ecological and economical objectives. The general aim of these real-world inspired benchmarks is to provide an easy-to-understand and fast-to-compute benchmark that still retains the actual challenges found in the real-world application. While they succeed as great options for specific use cases, they lack diversity for more general algorithmic testing. A suite of real-world benchmarks featuring a more diverse set of problems has been proposed by Tanabe and Ishibuchi [52].

Combining Single-Objective Functions

Brockhoff et al. [44] try to create a more general set of benchmarks for MOEAs by combining the single-objective bbob functions [34]. The rationale behind this was that real-world problems are also a combination of independent, or conflicting objectives [44]. So combining the well-known and accepted bbob functions should avoid artificial properties from constructing the problems around properties of the PF. This however comes with the drawback of unknown optima. Furthermore, it raises the question on which function combinations to use. The combination of all bbob functions results in quite a large set for bi-objective problems. Scaling to more objectives dramatically enlarges this number, resulting in an unrealistic amount of test functions. For this reason, Brockhoff et al. [44] hand-selects 55 test function combinations as the set of bbob-biobj benchmarks. Andova et al. [45] try to select a representative subset using exploratory landscape analysis (ELA) features [53], however this approach also features many open challenges. One of which is that even small changes in one objective can result in a vastly different fitness landscape.

2.5 Evaluating Evolutionary Algorithms

There are many ways to evaluate EAs. In this section, we will focus on the most common ways of how the performance of algorithms are compared. First, how SOEAs are usually evaluated is discussed. This is followed by the evaluations of MOEAs, where we discuss the performance indicators used to compare the output of these algorithms. While several techniques to evaluate the population dynamics also exist, their application remains scarce and is far off the "usual" way the majority of works evaluate EAs. These techniques will be discussed in the next Chapter 3.

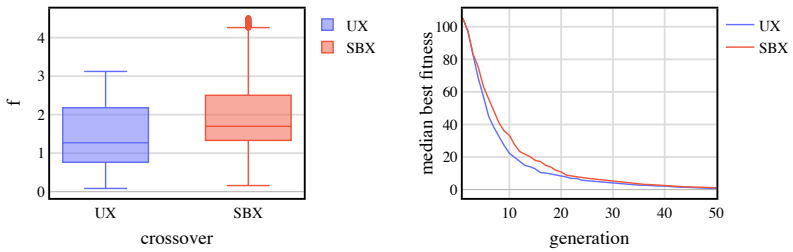
2.5.1 Single-Objective EAs

Metric

The evaluation of EAs can roughly be split into two categories: *Metric* approaches and *visualizations*. Evaluating SOEAs is usually done by the fitness of the currently best solution found by an algorithm. Only optimizing for a single objective usually means that there is a single best solution in a population, on which the performance of a population can be measured. The performance of two algorithms is usually compared by the best solution they were able to find.

Test Repeats

The evolutionary process of EAs involves randomness and is not deterministic. This means the algorithms can reach a different result each time. Comparing the performance between algorithms, therefore, requires repeating the tests multiple times. A commonly used rule of thumb is



(a) Exemplary boxplot for the final generation.

(b) Exemplary convergence plot over all generations.

Figure 2.12: Examples of typical plots to evaluate the performance of EAs. Two crossover operators are compared.

30 or 31 independent runs. Depending on the tested algorithms and distribution of the results, a larger number might be required for better robustness.

Visualizations

There are two most common visualizations when it comes to evaluating algorithmic performances. A small example is shown in Figure 2.12a and Figure 2.12b, where the performance of two different crossover operators for an EA are compared on the Rastrigin problem [36]. Boxplots are a common way to compare the performance of algorithmic configurations in a single generation. Figure 2.12a shows an example for this. One can compare which algorithm had a better median performance, as well as how far the solutions are spread. In the example, UX crossover shows both a lower spread, as well as a better median than SBX. The second most common visualization is plotting the algorithms performance over multiple generations. This is also referred to as a convergence plot. An example for this can be seen in Figure 2.12b, where the median of the best solutions found is plotted over multiple generations as a line plot. This allows us to investigate how fast the fitness of an algorithmic configuration improves, which is important for EAs, as we want to evaluate the performance over a multitude of stopping criteria. In the given example, UX shows a faster convergence, as well as a better overall performance on the Rastrigin problem. These are the most common plots used to evaluate algorithmic performance. Of course, more specific evaluations are also frequently used to gain insights into more specific cases. When evaluating a large number of algorithmic configurations, it is also common to see large tables showing the median fitness and standard deviation.

Limitations of Performance Testing

A lot of insights can be gained by comparing the performance of algorithms. Ultimately, we are interested in the best result, so it is very natural to compare the algorithms based on how well they are able to solve a given problem. However, this type of evaluation leaves a gap. While we can say that one algorithm performs better for a given problem, we cannot say *why*. This critique dates back decades, Hooker [12] already criticised this type of performance-based comparisons in the 1990s. While better guidelines and benchmarks have improved the field since then, little focus has been given to the question of *why*. McDermott, Georgiev, and Nicolau [54] show this for an exemplary case on a binary knapsack problem [55], where only evaluating the best found solution hides the population drifting into the infeasible areas of the search space. Also recently, Lezama et al. [3] argue the importance of a better explanation of how EAs reach their results for application domains, their focus lying on energy optimization. In this dissertation, we also support this view,

arguing for the importance of a better understanding of the population dynamics in EAs. As we will see in Part I of this thesis, works on evaluating the population dynamics do exist. However, their application still remains scarce, and the field still shows some gaps in their evaluation techniques.

2.5.2 Multi-Objective EAs

In general, MOEAs are evaluated the same way as SOEAs, comparing their performance over multiple test repeats. However, their performance cannot be directly compared using the fitness of the best solution, as these algorithms try to find a diverse set of tradeoff solutions. To enable performance comparisons, several metrics have been proposed, which are commonly referred to as *performance indicators*.

Performance Indicators

The goal of performance indicators is to compress the performance of a set of solutions found by a MOEA to a single metric, which can be compared like the best found fitness values in the single objective case. All performance indicators are calculated for the NDS found by an algorithm so far. We denote these solutions as $X^{NDS} = \{\vec{x}_1, \ldots, \vec{x}_\rho\}$, with $X^{NDS} \subseteq X$ and $\rho \leq n$. Over the years, many performance indicators have been developed. Here we focus on the four most popular ones: hypervolume (HV), generational distance (GD), inverted generational distance (IGD), and the inverted generational distance plus (IGD+).

Hypervolume (HV)

The hypervolume (HV) indicator was introduced by Zitzler and Thiele [56]. It is based on the combined area (or volume for $\mathfrak{z} > 2$) between each found solution in X^{NDS} and a selected reference point $\vec{r}$. A visual example for this is given in Figure 2.13. The HV is described in Equation 2.11. Here, Λ is defined as the Lebesgue measure. We further assume $[\vec{x}_i, \vec{r}]$ to be the volume limited by the solution $\vec{x}_i \in X^{NDS}$ on the one side, and the reference point $\vec{r}$ on the other side.

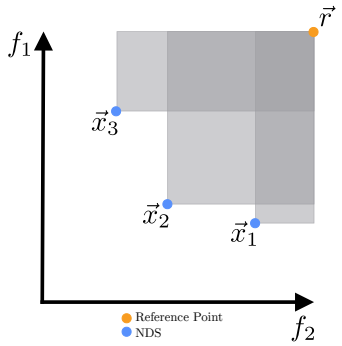


Figure 2.13: Visual example of HV.

$$HV(X^{NDS}, \vec{r}) = \Lambda\left(\bigcup_{x_i \in X^{NDS}} [\vec{x}_i, \vec{r}]\right) \tag{2.11}$$

The HV is the only known Pareto-compliant performance indicator [27, 57]. Furthermore, it is the only commonly used performance indicator not requiring knowledge about the true PF of the problem. However, it is computationally expensive and scales poorly with an increased number of objectives. Furthermore, the result is dependent on the choice of reference point [27, 58]. This choice of reference point also makes the comparison between the initially unconverged and finally converged population of a MOEA difficult. For more information on this, we refer the interested reader to [27, 58].

Generational Distance (GD)

The idea of the GD, proposed by Veldhuizen [59], is to calculate the distance from the NDS found by an algorithm to the true PF of the problem. In practice this is done with a set A of μ reference points approximating the true PF, denoted as $A = \{\vec{a}_1, \ldots, \vec{a}_\mu\}$, with $a_i \in PF$

and $1 \leq i \leq \mu$. The average Euclidean distance d from each solution x_i to the closest reference point is measured:

$$GD(X^{NDS}) = \frac{1}{\rho} \sum_{i=1}^{\rho} \min_{k=1}^{|A|} d(\vec{x}_i, \vec{a}_k) \tag{2.12}$$

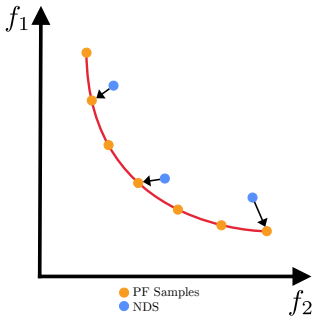


Figure 2.14: Visual example of GD.

The GD is an intuitive and fast-to-calculate performance metric. However, it requires the true PF of a problem, making it unsuitable for real-world problems where the optimal solutions are unknown. Furthermore, it only captures the convergence towards the PF, but not the spread across it. This means a good value can be reached even if all found solutions are concentrated in one part of the PF. Calculation of the indicator also depends on the selection of reference points, as Ishibuchi, Pang, and Shang mention in [27]. Finally, the GD indicator is not Pareto-compliant [57].

Inverted Generational Distance (IGD)

The inverted generational distance (IGD) is very similar to the GD, with the aim of not only capturing the distance from the NDS to the PF, but also the spread across it. This is done by reversing the distance calculation, measuring from the sample points to the nearest solutions in the NDS:

$$IGD(X^{NDS}) = \frac{1}{|A|} \sum_{k=1}^{|A|} \min_{i=1}^{\rho} d(\vec{x}_i, \vec{a}_k) \tag{2.13}$$

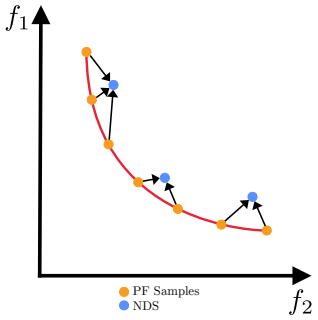


Figure 2.15: Visual example of IGD.

With this, the IGD measures both the distance to the PF as well as how well the population covers the whole front. However, similar to GD, the PF of the problem needs to be known. The indicator is also not Pareto-compliant [57] and is again dependent on the selection of reference points [27].

Inverted Generational Distance Plus (IGD+)

Finally, the IGD+ updates the IGD indicator by changing the distance measure, to address Pareto-compliance.

$$d^+(\vec{x}_i, \vec{a}_k) = \sqrt{\sum_{j=1}^{\mathfrak{z}} \left(\max\{x_{i,j} - a_{k,j}, 0\}\right)^2} \tag{2.14}$$

With this, IGD+ can be calculated as:

$$IGD+(X^{NDS}) = \frac{1}{|A|} \sum_{k=1}^{|A|} \min_{i=1}^{\rho} d^+(\vec{x}_i, \vec{a}_k) \tag{2.15}$$

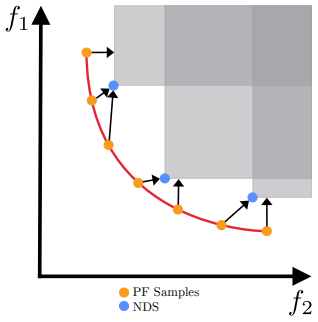
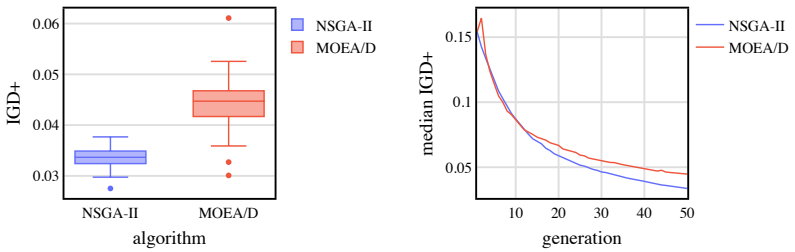


Figure 2.16: Visual example of IGD+.

A visual representation for the change in distance measure can be seen in Figure 2.16. IGD+ addresses most disadvantages of the GD. However, compared to HV, it is only weakly Pareto-compliant, and the calculation again requires knowing the true PF of a problem.

Summary of Performance Indicators

In summary, performance indicators allow compressing information about the convergence of MOEAs into a single metric. This allows a performance evaluation similar to SOEAs, and the comparison of a larger number of algorithms compared to visual comparison of the found solutions. However, each of the indicators features different advantages and disadvantages. These have to be considered when evaluating the performance with these indicators.



(a) Example of a boxplot.

(b) Example of a convergence plot.

Figure 2.17: Examples of typical plots to evaluate the performance of MOEAs. NSGA-II and MOEA/D are compared with the IGD+ performance indicator.

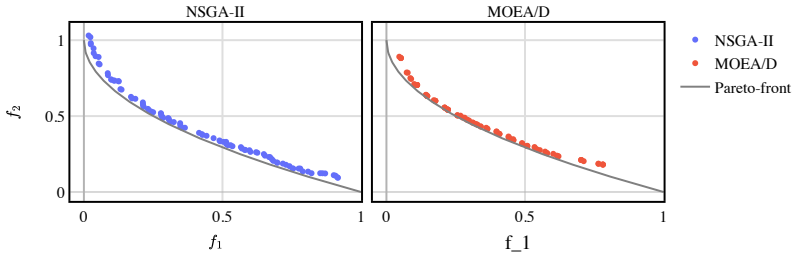


Figure 2.18: Example of a boxplot comparing the IGD+ of 31 test repeats for two algorithmic configurations.

Visualizations

Visualizing Performance Indicators

Essentially, the same visualizations as for SOEAs are commonly used. That is, boxplots for comparing the performance of algorithmic configurations in one generation, and convergence plots comparing their performance over multiple generations. Instead of the fitness of the best individual, the performance indicators are used. Figure 2.17a and Figure 2.17b show an example, where the performance in terms of the IGD+ [57] indicator on the LZ 1 [41] problem is compared between NSGA-II [30] and MOEA/D [31]. Essentially, both plots convey the same messages as in the single-objective case. However, it always has to be considered that the results of a performance indicator are dependent on its settings. For IGD+ this is the sample from the PF, for HV, this would be the choice of reference point. For a large number of algorithmic configurations, tables with mean or median values of a performance indicator, and their standard deviation, are also used. Of course, more detailed evaluations also exist, especially for problems with more than three objectives.

Visual Performance Comparison

In addition to the convergence- and boxplots of performance indicators, the search space of two- and three-objective tests can also be visualized directly. Figure 2.18 shows an example for this, again comparing NSGA-II and MOEA/D on the LZ 1 problem. These plots typically show the true PF, if it's known, and the fitness values of the population. This shows information which becomes hidden in the performance indicators, for example how well the PF is approximated in terms of closeness and coverage. In the example in Figure 2.18, NSGA-II covers the true PF better. This was also reflected in the IGD+ value. However, MOEA/D finds solutions a bit closer to the PF, especially in its center. While these plots contain more information about the convergence, they also take up a lot of space, and evaluating results for a larger number of algorithmic configurations becomes infeasible. For more than three objectives, more elaborate visualization techniques exist, for example through dimensionality reduction. A review of approaches for this was conducted by Tušar and Filipič [60].

Challenges of Performance Evaluation in MOEAs

Evaluating the performance of MOEAs is inherently more difficult than for just one single objective. This is both due to the difficulties with benchmark problems, as well as the need to compare sets of tradeoff solutions. While performance indicators solve this issue to a certain extent, they need to be set up properly, and can easily hide important information, which can only be found in visual comparisons of the NDS found. This section only scratches the surface of this topic. More detailed information about a fair performance comparison of MOEAs can be found in [27, 33]. These issues also amplify the previous critique on performance evaluations in SOEAs, namely that they show that one algorithm performes better, but not *why*. For this reason, this thesis argues for the need of evaluating the MOEAs beyond just their performance.

2.6 Summary

This chapter described the background of this work. We have discussed the foundational principles of optimization problems (Section 2.1), EAs (Section 2.2), and the basics of MOEAs (Section 2.3). Besides these fundamental concepts, the currently most common way of evaluating EAs is discussed in two parts. First, we discuss the use of benchmarking problems, some general design goals, and their limitations, both for a single- and multiple objectives. Second, the most usual ways of how EAs are evaluated, by their performance, were presented, and limitations of the approach were discussed. In these limitations lies the motivation of this work, trying to better understand the *why* and *how* EAs reach their solution.

Part I

Algorithmic Advances

Related Works and State of the Art

3

This chapter gives an overview of existing approaches to evaluate the population dynamics of EAs. First, we discuss some background and propose a categorization around the area of population dynamics in EAs. After this, the different methods for capturing and evaluating the population dynamics are presented, sorted by the defined categories. For a better overview, this chapter only focuses on the approaches, their capabilities, and limitations. The specific findings on population dynamics in EAs are separated into the respective parts of this thesis, this is Chapter 6 for SOEAs and in Chapter 9 for MOEAs.

3.1 The Credit Assignment Problem 25
3.2 Categories of Population Dynamics 26
3.3 Methods for Tracking Population Dynamics . . 26
3.3.1 Population Level 27
3.3.2 Individual Level 29
3.3.3 Gene Level 32
3.4 Population Dynamics in Algorithm Design 33
3.5 Summary 35

3.1 The Credit Assignment Problem

Credit Assignment in EAs

One inherent question in the field of population dynamics in EAs is what influence individuals from a past generation, or the used operators, have on the current population. In the literature, this is also referred to as *credit assignment* [14, 61]. However, this assignment of credit is not trivial. Whitacre, Sarker, and Pham describe this as the *credit assignment problem* [61], which features several open challenges.

The Difficulties in Credit Assignment

As humans, we inherit (roughly) half of our genetic material from each parent. However, in EAs this is not the case. Depending on the crossover operator used, one parent might contribute far more genetic material than another to create the new solution. This raises the question of how we should distribute credit among the parents. Most research on population dynamics in EAs either distributes credit equally for each parent [62, 63], selects a dominant parent to get the sole credit [14, 61, 64], or decides on a gene basis [13]. It is both non-logical to count each parent equally, or to only count the more dominant parent. Both approaches lose information, as it becomes unclear which genetic features stem from which parent. Furthermore, in the evolutionary process, there is no guarantee of genetic material from a distant grandparent still being retained in a current individual [13]. Should we still assign credit to a grandparent if none of its genetic material is retained in the offspring? On the one hand, it is part of the genealogical tree, on the other, no genetic material survived. Besides the genetic heritage of the parents, one could also consider the operators themselves to influence the search process [14, 61]. One could argue that it is not only the relevant genes, but also how effectively they are combined [61], or altered in mutation. It is questionable if we can fully attribute genetic heritage to the parents, or need to consider influences from the operators as well.

How to Assign Credit?

We argue that there exists no universal answer to these questions on credit assignment. Rather, it depends on the intended information to be gained from the evolutionary process. In this thesis, the aim is to evaluate the influence of individuals from the initial population throughout the evolutionary process. This needs to be considered in the design of the tools and evaluation techniques in this thesis.

Credit Assignment in Coevolution

We briefly want to acknowledge that in the field of EAs there exists another credit assignment problem for coevolutionary algorithms. Here, the question is the contribution of each sub-solution to the fitness of the combined result [65]. However, in this work, as this work is not concerned with coevolution, we only refer to the credit assignment problem for the heritage of individuals.

3.2 Categories of Population Dynamics

One challenge for population dynamics research in EAs is the amount of data to be processed. Existing approaches handle this by focusing on specific aspects of the evolutionary process. We propose a categorization into three levels, based on which aspect of a population is evaluated. In each level, different information and aspects of the convergence can be evaluated. These levels are: The *population level*, the *individual level*, and the *gene level*.

Population Level

The highest level category for this is the *population level*. The focus lies on the population itself, evolved by the EA. As this level views the population as a whole, the credit assignment between children and parents is no concern. The main challenges lie in the visualization of usually large dimensional spaces to extract humanly understandable information.

Individual Level

The *individual level* is the second category. In this level, the focus lies on the individuals in a population. The individual and its genome is always viewed as a whole. This level is most similar to the ancestral tracking we do as humans. Similarly to this, we also call these genealogical approaches. Depending on the research question at hand, the credit assignment can be an issue.

Gene Level

In the *gene level*, as the name suggests, the focus is the genes in the genome. Here we find specific gene tracking approaches. The heritage tracking approach developed in this thesis, the T-EA, also falls in this category. Tracking each gene separately resolves some credit assignment issues present in the previous level.

3.3 Methods for Tracking Population Dynamics

Even though the population dynamics research in EA is not among the most popular topics, several approaches have been developed over the past decades. To the best of our knowledge, no systematic review or overview exists for this area. Gaining an overview on existing works also proves difficult. Terms like "population dynamics" or "search dynamics" return few to no relevant works in this area. In this section, we want to provide an overview of the existing approaches in this area. This is done separately for each level of population dynamics. An overview of the related works in this area can be found in Table 3.2, in the summary of this chapter.

3.3.1 Population-Level Approaches

Dimensionality Reduction

De Lorenzo et al. [66] propose a visualization framework and assess the effects of different dimensionality reduction techniques for evaluating the population dynamics, with the main goal of preservation of population movement and exploration and exploitation tradeoff. They propose visualizations with two and three dimensions. The 2D visualizations show the individuals of a population in a dimensionality-reduced search space. They also show the possibility of adding fitness information, genealogical information, or the trajectory of the best individual through the search space. The 3D visualizations stack these previous 2D ones for each generation in the third dimension, showing the path of the population through the search space over the course of multiple generations. They tested different dimensionality reduction techniques, namely principal component analysis (PCA) [67], multidimensional scaling (MDS) [68], t-distributed stochastic neighbor embedding (t-SNE) [69], and uniform manifold approximation and projection (UMAP) [70] with a simple GA on a bit string and a continuous problem, both derived from the MaxOnes problem. De Lorenzo et al. conclude MDS to be the most suitable dimensionality reduction technique considering their focus.

Search trajectory networks (STNs)

Search trajectory networks (STNs) [71] are a recent addition of tools to evaluate the search dynamics of EAs. The goal was to visualize the path of a population through the search space in a graph-based way, showing frequently visited areas, locations where the algorithm gets stuck, or more generally how the search process of the population looks. This is done by tracking locations in the search space visited by an algorithm. These locations are represented as nodes. For continuous problems, the search space needs to be partitioned, as considering each unique solution as a search location is deemed infeasible [71]. To track these locations visited by the population, a representative individual is chosen. For SOPs this is done by selecting the best-known solution [71]. For MOPs, reference vectors can be used to track representative solutions [72]. This creates one STN for each reference vector, which is merged. The result, in both cases, is a network of the locations visited by the population of an EA. These networks can be evaluated in two ways [71]. First, STNs can be evaluated visually through their graph-based model. An example of this can be seen in Figure 3.1. The figure was created using STN Analytics [1], which is the most current. The figure was created using the exemplary data available at this website, comparing the search dynamics of a three algorithms. Second, graph metrics can be used to evaluate the data. Typical metrics include the total number of nodes, as the number of unique locations visited, the number of edges as search transitions between locations, the number of unique end locations for each algorithm, or the average path length [71].

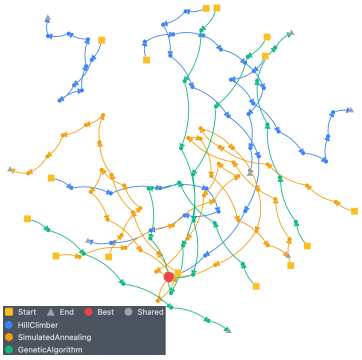


Figure 3.1: Example of a 2D search trajectory network (STN) visualization, created with the exemplary dataset at `https://www.stn-analytics.com/`.

Extensions of STNs

STNs are, besides genealogical evaluations (which will be covered in the next section) among the most applicable and most used tools to evaluate population dynamics in EAs. As mentioned, they were introduced by Ochoa, Malan, and Blum [71] for the use on discrete and continuous problems. In [73], they were extended for combinatorial problems with a binary encoding, the main difference being the search space partitioning. Narvaez-Teran, Ochoa, and Rodriguez-Tello [74] also extend STNs

[1] STN Analytics: `https://www.stn-analytics.com/`

for combinatorial optimization, this time for a permutation encoding. Furthermore, multidimensional scaling [75] is used to reduce the search space before partitioning. Sarti and Ochoa [76] extend STNs for the use with variably sized genomes, and more specifically focus on the neuroevolution of augmented typologies (NEAT) [77] system, which evolves the topology and weights of a NN at the same time. For this, they used a genotype mapping to build bytestreams out of the otherwise variably sized genomes in NEAT. This variant of STNs was later extended into neuroevolution trajectory networks (NTNs) [78]. These NTNs work similar to STNs, but are built on the behavior space of agent systems and not on the genome information in the search space. This behavior space is much smaller than the serialized genome space, which leads to easier evaluations [78]. As mentioned, Lavinas, Aranha, and Ochoa [72] adapted STNs for the use with MOEAs by decomposing the search space with reference vectors to select a few sets of representative individuals. These sets then are merged into one large STNs. One limitation of this initial approach for MOEAs was, that the network visualization did not support plotting the STNs of two algorithmic configurations at the same time. This was later addressed in [79]. STNs were also extended for the use with multi-objective combinatorial problems [80]. Hu, Ochoa, and Banzhaf [81] adapted and applied STNs to linear GP algorithms. Here, search trajectories are not only built on the search space, like in the original approach [71, 73], but also on the phenotype space, to better highlight fitness neutral transitions between individuals.

Search trajectory networks on the web (STNWeb)

The original implementation of STNs [71, 73] is based on R and available on GitHub[2]. To make the application and evaluation of the method more accessible, Chacon-Sartori, Blum, and Ochoa [82] created a web tool based on the original implementation in R. They also added some features, for example an automated search space partitioning or extending the limit of algorithms which can be compared simultaneously. A brief description of the software architecture of STNWeb can also be found in [83]. Chacón Sartori, Blum, and Ochoa [84] later extend STNWeb with the use of a new partitioning scheme using agglomerative clustering. With this, the authors want to improve the interpretability of plots, especially for problems of a large dimensionality. A brief introduction to this web application in its current form can be found here [85].

LLMs for Automated Algorithm Evaluation

While STNs are a useful tool for evaluating the population dynamics of EAs, they are also quite sophisticated and require a certain understanding and knowledge to be interpreted. Chacón Sartori, Blum, and Ochoa [86] try to lower the barrier to use STNs by incorporating large language models (LLMs), specifically GPT-4 [87], to aid in the evaluation process and lower the barrier of entry. For this, the LLM is given three tasks. First, to determine if there is a winner among the compared algorithms. Second, to suggest a better parameter setup when producing the STN graphs. And third, to produce an automatically generated summary with plots of the two previous tasks. The work also describes the prompt engineering to mitigate some drawbacks when using LLMs. The most important are the known struggle of LLMs with more complex mathematical reasoning [88, 89], and to avoid hallucinations[3] [90]. They conclude their work [86] that through their prompt engineering, the trustworthiness and

[2] STN repository on GitHub: https://github.com/gabro8a/STNs.

[3] In the context of LLMs hallucinations mean the generation of unrealistic or untrue data, which is problematic for real-world applications.

explainability of the LLMs can be improved. However, Chacón Sartori, Blum, and Ochoa also note users have to be aware and pay close attention to the generated results.

3.3.2 Individual-Level Approaches

Genealogy and Phylogeny

The main approach in the area of the individual level is tracking the relationships between the individuals of each generation, similar how we would build heritage trees of a family for humans. The EC community derives two terms from biology to describe this: *genealogy* and *phylogeny*. In the field of biology, phylogeny describes the heritage relationships between species, for example the relationship between humans, apes, and their common ancestors. Genealogy, on the other hand, describes the heritage relationship between two entities of the same species, for example human parent-child relationships [64]. While genealogy and phylogeny have different meanings in the context of biology, they are often used as synonyms in the realm of EAs. In practice, the same heritage data is tracked for each individual. For consistency, we will use the term genealogy in this thesis.

Tracking the Genealogy in EAs

Tracking the genealogy can be done by keeping a record of all occurring crossover and mutation operations over the course of an algorithmic run. A simple implementation of this can be found in the distributed evolutionary algorithms in python (DEAP) framework [91]. Recording all individuals and their relationships, however, can generate a lot of data, even for usual algorithmic configurations. To handle this better, Donatucci, Dramdahl, and McPhee [92] first propose to use a graph database (neoj4) to collect the genealogical data. They show the capabilities of this in a proof-of-concept evaluation for a standard sized GP configuration, mainly focusing on evaluation metrics like the number of contributing individuals, or the effectiveness of the crossover and mutation operators in terms of fitness improvement. While graph databases help with data storage and handling, the tracked genealogical information still produces a lot of data. In an evaluation by McPhee, Donatucci, and Helmuth [62], also using graph databases to save the complete genealogical history, the authors note that their evaluation of 200 test runs with a population size of for each of $n = 1000$ over 300 generations amounted to 49 GB of data. This was comparing two algorithmic configurations, each with 100 test repeats. Testing algorithms as suggested by Ishibuchi, Pang, and Shang [27], using different parameters and operators for each algorithmic configuration on a wide scale, remains a challenge.

Visualizing the Genealogy

The most intuitive way to evaluate genealogical data is to visualize the heritage trees in the form of a directed graph, similar to heritage trees for humans. An example of such a tree can be seen in Figure 3.2. It shows the genealogy of the best individual found for a sample run of the MaxOnes problem, and was generated using the built-in tool of the DEAP framework [91], with a population size of $n = 10$ over 10 generations. Each numbered node refers to one individual, with the connections indicating a parent-offspring relationship. Nodes with a single parent connection are created through mutations, while nodes with two parents stem from crossover operations. The main challenge for these visualizations is again the large amount of data. More sophisticated approaches are required to avoid cluttered plots when tracking the

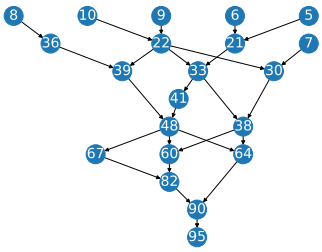


Figure 3.2: Genealogy tree of the DEAP framework [91].

genealogy of hundreds of individuals over hundreds. To the best of our knowledge, the first instance of a heritage tree being visualized in a scientific work was done by Burlacu et al. [93]. They visualize the genealogy of the best individual throughout the generations of an exemplary GP benchmark. Later in [94], Burlacu et al. introduce additional visualizations for the genealogical data, now also for the final generation of individuals, or the whole population. Additionally, they track and evaluate "genetic fragments", which in practice are subtrees, throughout the evolutionary process. While their evaluation shows promising results, the approach is limited in capacity. The graphs will get large and cluttered for large population sizes and many generations, and only one test run can be visualized at a time. This is a general challenge for these approaches, as evaluating the recommended wide range of algorithmic setups remains challenging. In a later book chapter, a proof of concept of how to use these evaluation tools for symbolic regression problems in GP is discussed in Affenzeller et al. [95]. In addition to the genealogical data, they also discuss the use of diversity metrics for the population. With the introduction of graph databases to track the genealogy of whole algorithmic runs, more elaborate evaluations on one test run became possible, and is the focus of [62, 63]. First, McPhee, Donatucci, and Helmuth [62] discuss and visualize the genealogy trees around interesting points in the evolutionary process. That is the points where the most common ancestors of the final population emerged, or the lineage of the best found solution. They do this with a comparison of lexical selection and tournament selection on a software synthesis problem. In their evaluation, they showed exemplary visualizations and discussions on why the algorithm behaved a certain way for one successful (optimal) individual, and one unsuccessful individual. The major difficulties in this area remain the amount of data and the size of the graphs generated for each generation [62]. Finally, McPhee et al. [63] extended their approach to visualize the heritage over all generations in a huge genealogy tree. Here, each individual was visualized as a rectangle, with the width representing the number of selections as a parent, the height the number of offspring in the graph, and the color by its total error (fitness). The graphs show patterns in the genealogical data and points of interest, where important individuals are discovered. In their proof of concept, they show the differences in these patterns between a successful (optimal) and unsuccessful individual. Besides visualizations, we also find several metrics to evaluate genealogical data, which are discussed in the following parts of this section.

Event Takeover Value (ETV)

Whitacre, Pham, and Sarker provide a first measure of the impact of an individual on the following populations in EAs in [14]. They propose the measure of event takeover values (ETVs), based on the genealogy of the individuals. In short, the ETV is calculated as the number of individuals in a current population with a genetic heritage to the measured individual. This means the maximum ETV possible for a measured individual is bound by the used population size ($\max(ETV) = n$). It is to note that the authors chose to only track the heritage for the most dominant parent in a crossover operation, which was assessed using the Euclidean distance. Whitacre, Pham, and Sarker [14] were self-critical about this choice and the credit assignment problem at hand. However, this needs to be considered when evaluating works on population dynamics using ETVs.

Exploration, Exploitation and Genealogy

Črepinšek, Mernik, and Liu [96] track the genealogy in binary encoded multi-objective GAs. In addition to the heritage, they also track how many bits in the genome changed from parent to offspring, using the Humming distance. Similar to [61], only the heritage of the dominant parent is tracked, measured by similarity (Hamming distance). Influence of the non-dominant parent is discarded. In addition to genealogical data, the parent-child similarity is also tracked. This data is used to measure if an individual was created through exploration or exploitation. If the child is similar to the parent, the algorithm is exploiting, and if it is different, it is exploring. The threshold for this is dependent on the problem and the similarity measure, and needs to be set by the user. Based on this threshold and the genealogical data of the heritage tree, they propose sixteen metrics to measure the population dynamics of a test run, seven for exploration, seven for exploitation and two additional. A proof-of-concept evaluation on the multi-objective 0/1 knapsack problem showed the potential of the approach. However, two drawbacks become apparent. First, only tracking the heritage of the dominant parent needs to be criticized in terms of credit assignment. Second is the need to set the threshold for exploration and exploitation manually. This threshold is dependent on the problem at hand, and can change the resulting metrics of population dynamics [96]. This is acknowledged in a later work [97], which calculates this threshold based on attraction basins in the fitness landscape.

Identifying Building Blocks of GP

In [98], Burlacu et al. propose a new method to identify building blocks in GP. They use genealogical data and develop an algorithm to trace specific genetic fragments throughout the evolutionary process. These are, in practical terms, subtrees of the individuals. This process was extended in a second work [99], tracking the frequency of these fragments. More frequently occurring fragments were classified as more influential in the search process. The work [99] also includes a first evaluation showing the capabilities of the approach on the poly-10 benchmark. In the same year, Burlacu, Affenzeller, and Kommenda [100] used the approach in a second study on population dynamics in GP. Here, they also introduced additional evaluation metrics. First, operator effectiveness measured the distance in fitness from the root parent to the offspring. Second, the average relative overlap as a similarity measure for the heritage trees of individuals. Finally, the contribution ratio, equivalent to the number of contributors in this work.

Ecological Theory in EAs

Dolson, Banzhaf, and Ofria [101] advocate the use of metrics from the biological field of ecological theory in EC. They use of genealogical data as a measure of diversity, and to evaluate which genetic lineages are created in the evolutionary process. The metric of *richness* is measured as the count of nodes from the current population to the most recent common ancestor. Furthermore, *divergence* is measured as the mean pairwise distance between the genomes in the population. Both metrics are derived from ecological theory, and provide a measure of diversity on the genealogy in EAs. In a proof-of-concept study on three GP problems, comparing the effects of different selection mechanisms, they could show genealogical diversity to be distinct from phenotypic diversity. Later, Dolson et al. [102] extended the approach. In addition to the metrics based on ecological theory, they also propose new *lineage metrics*. The lineage of an individual is reduced to a linear sequence of states on which these

metrics are computed. For crossover operations, they only consider one parent for the sequence, tracking the other parent as mutation influence, acknowledging that this process is losing information [102]. Four metrics are proposed to evaluate the lineage sequences of individuals: First, the lineage length as the number of states in the sequence. Second, the mutation accumulation, a set of measurements for changes through mutation, for example the number of mutations in a lineage, or the magnitude of change through mutation. Third, phenotypic volatility, as the rate at which the phenotype changes throughout a lineage. And fourthly, summary statistics to give a better overview of all lineages in the population at the same time. In addition to the metrics, the authors propose the use of a few visualizations, like lineage trees with a span of life component, overlaying the lineage of an individual on the fitness landscape of the problem, visualizing the state sequence, or using Muller plots.

Limitations of Genealogy

While many useful insights can be gained using genealogy trees, they also feature disadvantages. First, these ancestral trees can become very large for an algorithm running for hundreds of generations, involving tens of thousands of individuals. Multiple works mention this challenge in the literature [62–64, 102]. This is an obstacle for evaluations on a larger scale, as the amount of data collected becomes difficult to store, process, and visualize. This limits evaluations to certain statistical numbers, like number of ancestors [64]. Furthermore, even collecting this large amount of data is still subject to some information loss, leading to an issue of proper credit assignment. While mutation operations with a single parent could be reconstructed, for crossover operations we lose the information on which parts of the genome stem from which parents. This makes it impossible to reconstruct the amount of genetic material passed from the offspring to the parents.

3.3.3 Gene-Level Approaches

Gene Heritage in GP

Only few works which evaluate the population dynamics of an EA on a gene level exist. The first work on this topic is several years old. McPhee and Hopper [13] use heritage markers to track genetic diversity in GP. Each node of an individual gets assigned two IDs ($ID : memID$). Both are unique identifiers referencing the origin of a gene. The *ID* stores the origin of a gene in the initial population, unchanged by the evolutionary process. The *memID*, gets updated whenever the subtree of the node gets changes through crossover, as the node then is considered a new gene. Compared to genealogical approaches, this evaluation does not suffer from the problem of how to assign credit to each of the parent individuals, as the heritage of each gene is tracked separately. In their experiments, they find only 1 % of the nodes in the initial population survive to the final result. Genetic material of less than seven of the initial 500 individuals survived. Furthermore, in 90 % of their test runs, a single common ancestor for the whole final generation could be identified. Daida et al. [103] later repeated these experiments using a different algorithmic configuration, also extending the metrics used. In addition to evaluate which individuals from the initial population contribute to the result, they also looked at the frequency of occurrence of individuals.

Besides this, they employed Tufte style visualizations of the individuals found by the GP algorithm. They overall found similar results to [13].

3.4 Population Dynamics in Algorithm Design

Population dynamics data is not only used for evaluating algorithms. There also exist techniques which track and use information about the population dynamics in the algorithmic design. While this work is more focused on the evaluational side, we still want to highlight the related work in this area, as it is insightful to see which information is used to guide the search of an EA. An overview of all approaches can be found in Table 3.1. Interestingly, we did not find population level data to be used.

Ancestral Mating Restrictions

Affenzeller et al. [95] propose a GP algorithm with the aim of managing diversity, called genetic programming using best individuals of genealogies (GPBIG). Each individual from the initial population represents one lineage. Created offspring is only kept when it is the best of its respective lineage, meaning featuring a higher fitness than its parents. Furthermore, a genetic lineage is removed if it does not produce any better offspring for a number of generations. In a small evaluation against a canonical GP configuration, GPBIG showed promising first results.

Control Approaches for Population Diversity

Zhu and Liu [104] use an adaptive control approach to manage the population diversity by adjusting crossover and mutation probabilities for a GA on a traveling salesman problem (TSP). Each individual in the initial population is marked with a unique ID (*memID*), which gets passed from parent to offspring in recombination. The number of unique IDs left is used alongside three other population related measures to determine the population diversity. The goal is comparable to [95, 105], trying to avoid a domination of one individual in the evolutionary process. Credit assignment is not done accurately though, as in the crossover operation, one parent passes its ID to one child, with the other child inheriting the ID of the other parent. This means each child gets credited with 100 % of influence from just one of the parents. While the approach might be sufficient for diversity management, it does not provide an accurate measure of credit assignment when evaluating the population dynamics. Liu et al. [106] also propose an adaptive control approach to manage an algorithm's exploration and exploitation based on the metrics proposed by Črepinšek, Mernik, and Liu [96]. The authors conclude that a balance between exploration and exploitation throughout the generations, and a higher selective pressure on exploitation[4] showed the best performance.

Genealogical Resampling

Kuber et al. [105] propose to use genealogical data to trigger resampling events. Similar to lineage selection [107], the aim was to increase diversity by decreasing the convergence of the whole population. They use the genealogical information as a measure of convergence, or similarity, for individuals. If a majority of the population is genealogically linked to one ancestor, they assume a loss in diversity and introduce new genetic material. In experiments on the Schwefel function they showed this approach to improve algorithmic performance.

[4] Meaning only the best individuals are used for exploitation

Fast Fitness Estimations for Software Synthesis

Evolving programs with GP can require evaluating solutions against a large number of training cases, which can get computationally expensive. To strike a balance between thoroughness in evaluation and computational cost, approaches try to estimate the quality of an individual [108], or use subsampling techniques to estimate solution qualities on a smaller number of tests [109, 110]. Genealogical data has been used for both approaches. In [108], two types of ancestry-informed fitness estimation are proposed. The assumption was that the fitness of an offspring will be similar to the fitness of its ancestors (type one), or general relatives (type two). Computational tests on four problems from the program synthesis benchmark suite were conducted. Results indicate that the proposed operators increase the population's diversity and search space exploration. Performance improvements, however, varied between the problems and the algorithmic configuration used. Lalejini et al. [109][5] propose two ancestry-informed subsampling mechanisms. They compare their approaches on ten software synthesis problems against traditional random subsampling. They find that their genealogy-informed approaches work well in cases where only a small subset (1 %) of test cases are used in evaluation. In other cases (10 % of test cases for evaluation), results were more mixed, and the proposed approaches did not outperform traditional random subsampling.

Lineage Selection

Burke et al. [107] propose a selection operator for crossover in GP, called *lineage selection*. The operator is based on tournament selection. However, only individuals with no common root parent are allowed to be combined. Similar to McPhee and Hopper [13], the heritage of each individual is tracked, but only for the root node. Individuals with the same root ancestor are not allowed to mate. This way, the authors hope to increase genetic diversity. In their test they conclude that the operator does succeed in this goal. However, algorithm performance did not improve across the board. While for deceptive problems, lineage selection could show performance improvements, performance on simpler fitness landscapes was decreased.

Implicit Mating Restrictions

Essam and McKay [111] also propose a selection approach with the goal of limiting the chance of a single individual dominating the whole population in GP. However, instead of precisely tracking the heritage of each individual, they use a *heritage tag*, which in practice is an integer vector. Each time, offspring is created in crossover, parts of this vector are exchanged at random, like uniform crossover. For mutation, parts of the vector get assigned with new values. When selecting individuals for recombination, a minimum dissimilarity between the heritage tags is required. By scaling the size of the vector and the required dissimilarity, they can restrict for how many generations offspring cannot reproduce with each other. Compared to precise heritage tracking, like in [107], a certain randomness is introduced, meaning sometimes offspring of the same individual can reproduce sooner, sometimes later. In a proof-of-concept evaluation, the proposed heritage tag approach produced comparable performance to a traditional tournament selection.

Historical Markers in NEAT

Finally, the NEAT framework [77], which aims at evolving the topology as well as the weights of a network at the same time, uses such markers under the name *innovation numbers*. These markers are similar to the ones employed by McPhee and Hopper [13], with each node getting

[5] An extended version of the work is available [110].

assigned a unique ID. However, these IDs are not used for population dynamics evaluation, but are used in crossover to find matching parts of the genome, avoiding a more costly topological analysis. These numbers are also just used to track the connecting genes, and not the node genes of the genome. This contrasts with the other types of gene tracking, including the one proposed in this thesis, in which all genes and all changes in the evolutionary process are traced to evaluate population dynamics.

Table 3.1: Literature overview of population history and dynamics used in the design of EAs.

Author(s)	Work	Year	Level	Description / Contribution	Algorithm	Objectives
Zhu and Liu	[104]	2004	Individual	Control approach for population diversity	GA	SOO
Liu et al.	[106]	2013	Individual	Controlling exploration and exploitation	GA	MOO
Affenzeller et al.	[95]	2014	Individual	Restricting crossover by genetic lineage	GP	SOO
Lalejini et al.	[108]	2024	Individual	Ancestry-informed fitness estimation	GP	SOO
Lalejini et al.	[109]	2024	Individual	Ancestry-informed subsampling	GP	SOO
Kuber et al.	[105]	2014	Individual	Extended version of [109]	GP	SOO
Burke et al.	[107]	2003	Gene	Lineage Selection	GP	SOO
Essam and McKay	[111]	2004	Gene	Lineage Selection	GP	SOO
Stanley and Miikkulainen	[77]	2002	Gene	NEAT system	GP	SOO

3.5 Summary

Chapter Overview

In this chapter, the related approaches to track the population dynamics in EAs are discussed. This is started by a brief discussion on the credit assignment problem in this area (Section 3.1). Furthermore, a classification in three levels (population, individual, gene) for approaches tracking the population dynamics is proposed in Section 3.2. The main part of this chapter is concerned with discussing the related approaches on tracking and evaluating population dynamics in EAs. To the best of our knowledge, there currently exists no overview on all approaches, although this area is already decades old. Reviewing the literature shows the potential of these approaches, as well as gaps for which no tools exist yet. An overview of the most important findings is also provided in the following parts of this summary, together with Table 3.2 which contains the discussed approaches. The final Section 3.4 briefly discusses the related works using population dynamics information in their algorithmic design. In the following paragraphs, we briefly summarize the most interesting findings of the literature review, and discuss the open challenges found in this field.

Population-Level Findings

Early works in the population level employ dimensionality reduction techniques to visualize the population in the search space [66]. Later Ochoa, Malan, and Blum [71] propose STNs, tracking, visualizing and evaluating the path of a population through the search space in a graph based way. This approach has been extended for the use of different algorithm and problem types, and are currently probably the most active research area in the field of population dynamics. Part of this success is probably also the fact that a relatively easy-to-use version is provided freely as a web service with STNWeb [82].

Individual-Level Findings

The individual level is synonymous with genealogical evaluations. These are intuitive to understand, as it is derived from heritage tracking in biology. Two main challenges exist here. First is the amount of data generated when tracking and evaluating the genealogy of whole algorithmic runs [62, 92]. Second, works here suffer from the issue of credit assignment. Some works track the heritage of both parents with the same weight [62, 63], some works only track the heritage of one parent [14, 15, 61]. No matter how they are implemented, both suffer from an inherent information loss. As McPhee and Hopper [13] point out, some individuals might be in the same lineage, but do not share any genetic material. Nevertheless, genealogical evaluations at the individual level are, besides STNs, among the most researched in our area. Increases in computational power and the introduction of graph databases allowed for the tracking and visualization of whole algorithmic runs [62, 63], which can show interesting points during the evolutionary process. Furthermore, several metrics, like divergence or richness derived from ecological theory [101, 102], or ETVs, have been developed for the use on genealogical data.

Gene-Level Findings

Only two works [13, 103] focus on the population dynamics on the gene level. The gene tracking mechanism employed is focused on GP, using two IDs to track the heritage from the parent and the root parent. While this is sufficient for GP, other types of algorithms, like GAs potentially require multiple parents to be tracked per gene, depending on the operator. Both works show the potential of evaluating the population dynamics more gradually on a gene-based level. First, they do not suffer from the problem of credit assignment. And second, they can explore parent-child relationships in more detail, for example by contrasting the root parent against the non-root parent in the crossover operations. However, both works are also quite old, and were never adapted for the use of different algorithm types, or with multiple objectives.

Population History in Algorithm Design

This chapter also discussed the use of heritage information in the design of EAs. Table 3.1 shows an overview of the existing works. Some studies find the use of heritage data in the algorithmic design to improve performance [95, 105]. However, the majority of works show no clear performance improvement, or mixed results depending on test problems and algorithmic setup [108–110]. While the ideas found in the works are very interesting, they do not seem to get much traction among the EA community so far.

Open Challenges

While several methods for evaluating the population dynamics of EAs exist already, we still find some challenges and gaps. First is the space and time needed for evaluating test data. Many of the discussed approaches are limited in their capacity, as evaluating the data is costly in terms of storage, time, or space in scientific publications. Testing a number of different algorithmic configurations and repeating tests is necessary to gain robust results. Second, we see a clear lack of approaches on the gene level. Two works exist in this space already. However, both are aimed at GP optimizing a single-objective. We believe this area to be key towards solving the issue of credit assignment, from which all genealogical approaches suffer. Third, and finally, we find most works to focus on single-objective optimization (SOO) and GP. This leaves a gap for other types of EAs, and multi-objective optimization (MOO). This thesis aims to address these gaps by developing a new approach to track genetic heritage in EAs, which is scalable, applicable to all commonly used operators and algorithms, and usable for MOPs.

Table 3.2: Literature overview of evaluation methods for population dynamics EAs excluding the contributions of this thesis.

Author(s)	Work	Year	Level	Description / Contribution	Algorithm	Objectives
McPhee and Hopper	[13]	1999	Gene	Number of ancestors in GP	GP	SOO
Daida et al.	[103]	2004	Gene	Extending [13] by introducing additional metrics	GP	SOO
Burlacu et al.	[98]	2015	Individual	Tracing genetic building blocks with genealogical data	GP	SOO
Burlacu, Kommenda, and Affenzeller	[99]	2015	Individual	Tracing genetic building blocks with genealogical data	GP	SOO
Burlacu et al.	[93]	2012	Individual	Visualizing genealogy	GP	SOO
Burlacu et al.	[94]	2013	Individual	Visualizing genealogy, update to [93]	GP	SOO
Donatucci, Dramdahl, and McPhee	[92]	2014	Individual	Tracking genealogy with graph databases	GP	SOO
McPhee, Donatucci, and Helmuth	[62]	2016	Individual	Visualizating genealogy trees, extending [92]	GP	SOO
McPhee et al.	[63]	2017	Individual	Visualizing genealogy trees, extending [62]	GP	SOO
Collier and Wineberg	[64]	2007	Individual	Number of ancestors metric	GA	SOO
Whitacre, Pham, and Sarker	[14]	2006	Individual	Introduction of ETVs	GA	SOO
De Rainville et al.	[91]	2012	Individual	Genealogy tracking in framework	EA	SOO+MOO
Affenzeller et al.	[95]	2014	Individual	Discussing population dynamics evaluation	GP	SOO
Zhu and Liu	[104]	2004	Individual	Lineage tracking with IDs	GA	SOO
Črepinšek, Mernik, and Liu	[96]	2011	Individual	Genealogy, exploration and exploitation	GA	MOO
Dolson, Banzhaf, and Ofria	[101]	2018	Individual	Genealogy, exploration and exploitation	GP	SOO
Dolson et al.	[102]	2020	Individual	Genealogy, exploration and exploitation	GP	SOO
De Lorenzo et al.	[66]	2019	Population	Dimensionality reduction of search space	GA	SOO
Ochoa, Malan, and Blum	[71]	2020	Population	STNs	GA	SOO
Ochoa, Malan, and Blum	[73]	2021	Population	STNs for combinatorial problems	DE, PSO	SOO
Narvaez-Teran, Ochoa, and Rodriguez-Tello	[74]	2021	Population	STNs for combinatorial problems	GA	SOO
Sarti and Ochoa	[76]	2021	Population	STNs for variable sized genomes / NEAT	GA	SOO
Sarti, Adair, and Ochoa	[78]	2022	Population	NTNs	GA	MOO
Lavinas, Aranha, and Ochoa	[72]	2022	Population	STNs for MOEAs	GA	MOO
Lavinas et al.	[79]	2022	Population	Extending STNs for MOEAs	GA	MOO
Ochoa et al.	[80]	2023	Population	Extending STNs for combinatorial MOEAs	GA	MOO
Chacon-Sartori, Blum, and Ochoa	[82]	2023	Population	Web tool for STNs	EA	SOO+MOO
Chacon-Sartori, Blum, and Ochoa	[83]	2023	Population	STNWeb software architecture	EA	SOO+MOO
Chacón Sartori, Blum, and Ochoa	[84]	2024	Population	Updated search space partitioning for STNWeb	EA	SOO+MOO
Chacón Sartori and Blum	[85]	2024	Population	Short introduction to STNWeb	EA	SOO+MOO
Chacón Sartori, Blum, and Ochoa	[86]	2024	Population	LLMs for automated evaluation of STNs	EA	SOO+MOO

T-EA: The Traceable Evolutionary Algorithm

4

This chapter presents the T-EA, an algorithm designed to track the gene heritage from the initial population throughout the evolutionary process of an EA. This is done by attaching historical markers to each gene. Over the past years, two implementations have been proposed. First, the *trace-vector* implementation, then the *trace-list* implementation. Both follow the same design concept, but feature different advantages and limitations.

For a better overview, this chapter first describes the general concept of the T-EA before diving into the details of both implementations. Finally, the application of the T-EA on MOPs is briefly discussed, and the chapter is closed with a short summary.

4.1 Concept 39
4.1.1 General Idea 39
4.1.2 Evaluation Metrics 40
4.2 Trace-Vector Implementation 41
4.2.1 Encoding and Operators . 41
4.2.2 Evaluation Metrics 42
4.3 Trace-List Implementation 43
4.3.1 Encoding and Operators . 43
4.3.2 Evaluation Metrics 45
4.4 T-EA for MOEAs 47
4.5 Summary 47

4.1 Concept

4.1.1 General Idea

Idea

The idea for the T-EA is to track the genetic material from the initial population throughout the evolutionary process. The concept was first introduced in [112, 113], and later extended for the use with more genetic operators in [114] and with MOEAs in [115]. Two implementations and representations of the algorithm have been developed: The *trace-vector* and the *trace-list*. Specifics and differences between the implementations will be presented in the following sections. However, they all follow the same general concept, which we briefly discuss here.

Historical Markers

To track genetic heritage in the evolutionary process, we attach *historical markers* to each gene. The exact representation of these markers differs between the implementations. As their names suggest, these markers can be a single traceID, or a list. They are evolved alongside the genome throughout the crossover operations, keeping its reference of origin. Mutations can also be tracked by assigning new markers. With this, the origin of each gene is kept and can be evaluated.

Crossover
Copying vs. Combining

For the heritage tracking process, we differentiate between genome *copying* and genome *combining* crossover operators. Copying operators simply copy the whole genome value of one parent to the offspring, like in n-point- or uniform crossover [24]. This means, the origin of that gene can be traced back to one parent only. Combining operators, as the name implies, combine the genes of both parents to a new one. The simplest example would be the average between two genes. In this case, one gene can have multiple parents, as both influence the new gene value in some way. A popular example for such an operator is SBX [25]. The differentiation between gene-copying and combining crossover is important when considering proper credit assignment for a genes heritage, as the two implementations presented later differ in their capability.

Mutation
Value Dependent vs. Value Independent

Similar to crossover, mutation operators can also be classified. We differentiate between *value-independent* and *value-dependent*. Value-independent operators change the original genome value without regard to the previous value. Uniform mutation or bit-flip mutation are good examples for this. Value-dependent mutations, on the other hand, change a gene based on its current value. Examples for this are the Gaussian mutation (GM) [116] or the popular PM [26]. Similar to the crossover, this differentiation is important for credit assignment and choice of implementation.

4.1.2 Evaluation Metrics

The historical markers allow us to track the heritage information throughout the evolutionary process. Alongside this, metrics to evaluate this data are proposed in this thesis. The exact formulation depends on the implementation used. Here, a short overview is given.

Contributors

First, individuals can be classified into *contributors* and *non contributors*. This classification was introduced in [115]. Practically, contributors are the individuals whose genetic material survived to the current generation. Non-contributors are the individuals for whom no historical markers were found in the evaluated population. The metric is inspired by Collier and Wineberg [64], who count the number of ancestors from the initial population which survived, using genealogical data. However, it has to be noted that it is possible for individuals to have an ancestral relationship with an individual from the initial population, without actually containing any genetic material from it anymore [13]. As we use gene-level data, we count only the individuals who actually contribute genetic material to the current population.

Impact Metrics

Further, we propose four metrics to measure the influence one individual from the initial population has on the current population [112–114]. In this work, we call this the *impact* of an individual, which is created as follows:

Counting-based impact (I_C) This is the most straightforward way, simply counting the occurrence of a historical marker in the current population. In practical terms, this metric returns the percentage of how many genes in the current population are related to a measured individual in the initial population.

Fitness-based impact (I_F) In EAs, good fitness individuals usually have a higher chance to survive to the next generation. This is reflected in this metric, scaling the occurrence of each historical marker with the fitness of the individual.

Entropy-based impact (I_E) The idea for this metric is to scale the occurrence of historical markers with the diversity in a decision variable. If, across the population, there are a lot of different values in one decision, it is likely not converged yet. It has importance for the future evolutionary process. If a decision variable has a low diversity, it is likely converged and may not have much influence in the following generations.

Fitness and Entropy-based impact ($I_{F \cdot E}$) This essentially is a combination of the two previous metrics, scaling both with the fitness of the individual and the entropy of the decision variable of the gene.

4.2 Trace-Vector Implementation

4.2.1 Encoding and Operators

This first iteration of the T-EA was proposed in [112, 113]. A historical marker is attached to each gene. We call this marker *traceID*, which points back to its genes origin in the initial population. A visual example of this implementation can be seen in the Figure 4.1. We denote an individual i with its genome $\vec{x}_i$, and a *trace-vector* $\overrightarrow{tv}_i$ of the same size n. The trace information of a whole population is denoted as $TV = \left\{\overrightarrow{tv}_1, \ldots, \overrightarrow{tv}_n\right\}$, similar to the genomes of all individuals in a population X. In the initial population, the first individual is initialized with the traceID "1", the second with traceID "2" and so on. The highest given traceID is always equal to the population size n. Figure 4.2 shows a visual example for this. Throughout the evolutionary process, these trace vectors are evolved alongside the genome, keeping the heritage information of each gene. Although developed independently, this implementation is similar to the gene tracing approach for GP by McPhee and Hopper [13]. Both approaches, however, differ in two ways. First, we only track the heritage of the direct parent with one traceID, as there is no root parent in GAs. Second, we initialize each individual with the same traceID, as operators in GAs usually do not swap values between decision variables, while McPhee and Hopper have to assign unique IDs to each gene. This way, we do not need to separately match each traceID back to the parent, simplifying the evaluation later.

Encoding

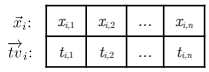


Figure 4.1: TraceID implementation of T-EA.

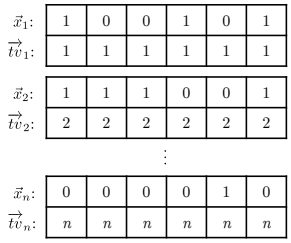


Figure 4.2: Initialization of the T-EA.

In crossover, the trace vectors for the offspring individuals are combined alongside the genome. Figure 4.3 shows an example for a one-point crossover operation of two binary encoded individuals. The trace-vector is split at the same point as the genome. This way, the heritage information is preserved and can be reconstructed, even if both genomes have the same value in one decision variable, like in the first or fifth genes in the example of Figure 4.3. Using this approach is an intuitive and fast-to-compute way of heritage tracking. However, it has to be noted that using a single traceID allows us to keep the heritage information of only one parent for each gene. This means this implementation only supports copying crossover operators, as combining two gene values would result in more than one parent influencing the gene.

Crossover

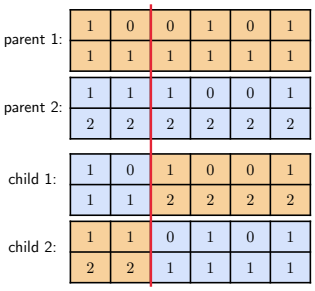


Figure 4.3: Crossover example for trace vectors.

Mutations can be traced by introducing new and currently unassigned traceIDs. In practical terms, descending negative numbers are used to differentiate between genes from the initial population, and between each occurring mutation. The first mutation gets the number -1, the second -2, and so on. An exemplary mutation operation can be seen in Figure 4.4. The traceID of the old gene is replaced with the new mutationID, indicating the changed genome value. Again, the limitations of the trace implementation need to be mentioned here, as only value-independent mutations can be accurately traced.

Mutation

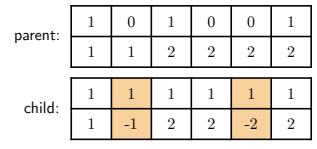


Figure 4.4: Mutation example for trace vectors.

Using one traceID for each gene is the simplest form of heritage tracking. This simplicity leads to a problem with proper credit assignment when considering crossover operators which combine gene values, or mutation operators, which are dependent on the old gene value. However, this implementation is not very resource intensive, as it only requires a second

Pros and Cons

integer vector for each individual, and is faster to compute compared to the following trace-list implementation, only linearly increasing runtime as the trace vectors are crossed and mutated alongside the genome.

4.2.2 Evaluation Metrics

Alongside the heritage tracking process, we propose evaluation metrics to assess the population dynamics of EAs using the trace data of the T-EA. First, contributing individuals are defined, which were introduced in [115]. Furthermore, four distinct metrics to measure the impact of initial population individuals on the following generations are presented, as proposed in the author's previous work [112, 113].

Contributors

Contributors ($\mathfrak{C}$) are defined as the individuals whose genetic material is found in the current population [115]. For the trace vectors, they can be defined as a set of traceIDs found in a population:

$$\mathfrak{C}(TV) = \{t | t \in TV\} \tag{4.1}$$

Based on that, we can further define the *number of contributing individuals* in a population as a measure of how many individuals from the initial population had some genetic material survive as $|\mathfrak{C}(TV)|$.

Counting-Based Impact (I_C)

The most intuitive way to measure the impact of a traceID k on a population is to count how many genes are influenced by it. We therefore call this metric the counting impact (I_C). For a traceID k, its I_c can be calculated as seen in Equation 4.2. The number of genes with a heritage to the traceID k are summed with an indicator function and normalized with the population size n and the genome size $\mathfrak{z}$. In practical terms, the I_C returns the amount of k in a given population as a percentage.

$$I_C(TV, k) = \frac{1}{n \cdot \mathfrak{z}} \cdot \sum_{i=1}^{n} \sum_{j=1}^{\mathfrak{z}} \mathbf{1}\left(TV_{i,j} = k\right) \tag{4.2}$$

Fitness-Based Impact (I_F)

The idea for the I_F is to take the quality of individuals, measured by the fitness function $f(\vec{x}_i)$, into account. In the counting-based impact I_C, the heritage information of all genes has the same weight. However, in EAs, good fitness individuals usually have a higher chance of survival, and therefore a higher chance to pass their genetic material to the next generation. We want to reflect this in I_F by scaling the occurrence of each traceID with the fitness of the individual. We do this using the distance in fitness to the worst (highest fitness) individual in the population, as seen in Equation 4.3. Genes in individuals with a low fitness, therefore, have a high fd. With this, we can scale the occurrence of each traceID k, as in Equation 4.4. We again normalize the metric to get a percentual value.

$$fd\left(\vec{x}_i, X\right) = |\max_{\vec{x} \in X}\left(f(\vec{x})\right) - f\left(\vec{x}_i\right)| \tag{4.3}$$

$$I_F(TV, X, k) = \frac{\sum_{i=1}^{n} \sum_{j=1}^{\mathfrak{z}} \mathbf{1}\left(TV_{i,j} = k\right) \cdot \left(1 + fd\left(\vec{x}_i, X\right)\right)}{\mathfrak{z} \cdot \sum_{i=1}^{n} 1 + fd\left(\vec{x}_i, X\right)} \tag{4.4}$$

Entropy-Based Impact (I_E)

Third, we define I_E. The idea is to increase the importance of decision variables, which are not converged yet, still featuring different values across the population. The Shannon-entropy [117] is used as the measure of diversity, as specified in Equation 4.7. For this, the probability of a traceID k appearing in a decision variable j is defined as:

$$p\left(TV_{i,j_{i=1}^{n}}, k\right) = \frac{1}{n}\sum_{i=1}^{n}\mathbf{1}\left(TV_{i,j} = k\right) \tag{4.5}$$

Further, we define $\mathfrak{TL}\left(TV_{i,j_{i=1}^{n}}\right)$ as the set of distinct traceIDs of the jth decision variable in the trace vectors TV of the population:

$$\mathfrak{TL}\left(TV_{i,j_{i=1}^{n}}\right) = \left\{t | t \in TV_{i,j_{i=1}^{n}}\right\} \tag{4.6}$$

With this, we can define H based on the Shannon-entropy [117]:

$$H\left(TV_{i,j_{i=1}^{n}}\right) = -\sum_{k\in\mathfrak{TL}\left(TV_{i,j_{i=1}^{n}}\right)} p\left(TV_{i,j_{i=1}^{n}}, k\right)\cdot\log_2\left(p\left(TV_{i,j_{i=1}^{n}}, k\right)\right) \tag{4.7}$$

H is low when the genes of one decision variable contain the same value, and high if they contain different values. With this, we can scale the impact as seen in Equation 4.8. To avoid division by zero, 1 is added.

$$I_E(TV, k) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{\delta}\mathbf{1}\left(TV_{i,j} = k\right)\cdot\left(1 + H\left(TV_{\cdot,j}\right)\right)}{n\cdot\sum_{j=1}^{\delta}1 + H\left(TV_{\cdot,j}\right)} \tag{4.8}$$

Fitness and Entropy-Based Impact ($I_{F\cdot E}$)

Finally, as the fourth influence metric, both the scaling factors of fitness (Equation 4.3) and entropy (Equation 4.7) are combined into (). This way, both genes in good fitness individuals, and genes for undecided decision variables, are weighted higher.

$$I_{F\cdot E}(TV, X, k) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{\delta}\mathbf{1}\left(TV_{i,j} = k\right)\cdot\left(1 + fd\left(\vec{x}_i, X\right)\right)\cdot\left(1 + H\left(TV_{\cdot,j}\right)\right)}{\sum_{i=1}^{n}\sum_{j=1}^{\delta}\left(1 + fd\left(\vec{x}_i, X\right)\right)\cdot\sum_{i=1}^{n}\sum_{j=1}^{\delta}\left(1 + H\left(TV_{\cdot,j}\right)\right)} \tag{4.9}$$

4.3 Trace-List Implementation

4.3.1 Encoding and Operators

Encoding

The *trace-list* implementation of the T-EA was first proposed in [114]. Similar to the trace-vector implementation, a trace component is attached to each gene. The traceIDs of the previous implementation get replaced with a *trace-list* ($tl = (tt_1, \ldots, tt_n)$), containing the heritage information. This way, the origin of multiple parents influencing one gene can be stored in the form of *trace tuples* ($tt = (t, w)$). The first element of the tuple is the traceID ($tt_1 = t$), and the second a weight $tt_2 = \omega$. The traceID t again points back to an individual in the initial generation. The weight ω determines the influence percentage from the traceID on the gene. The total influence factors of all trace tuples in a trace-list always sum to one ($\sum_{tt\in tl} tt_2 = 1$). This representation allows tracking multiple parents for

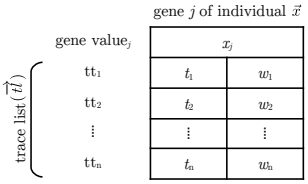


Figure 4.5: Trace-list encoding.

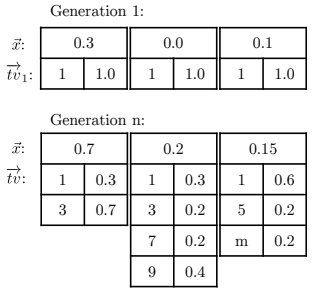


Figure 4.6: Example of trace-list encoded individuals in generation 1 and n.

one gene [114]. The trace lists for all genes in an individual i are stored in a vector $\overrightarrow{tl}_i = [tl_1, \ldots, tl_s]$. Similar to the trace vectors, we denote the trace information for a population of individuals as $TL = [\overrightarrow{tl}_1, \ldots, \overrightarrow{tl}_n]$. A visual example for the new encoding can be seen in the figure to the side (Figure 4.5). In the initialization process, all genes of an individual are initialized with only one trace tuple in the trace-list, containing the traceID corresponding to its individual (e.g. *ind. 1* is initialized with the trace tuple $(1, 1.0)$, *ind. 2* is initialized with the trace tuple $(2, 1.0)$ and so on). Figure 4.6 shows an example of an individual in the initial generation, as well as one individual later in the evolutionary process [114]. In the later generations, multiple traceIDs influence a gene with different weights. This way, the heritage information can be precisely tracked.

Crossover

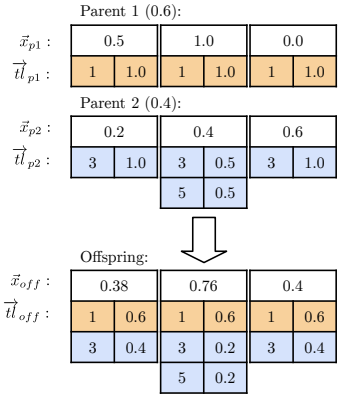


Figure 4.7: Trace-list example crossover.

In the trace-vector implementation [112, 113], the traceIDs could simply be swapped alongside the genes. This approach is still sufficient for gene-copying crossover operators, like UX [24]. For gene-combining crossover operators, a recombination step is needed to combine the trace lists alongside the gene values for the offspring. To compute this offspring trace-list (tl_{off}), the trace lists of its parents (tl_{p1}, tl_{p2}) need to be combined. Combining crossover operators like SBX [25] usually do not combine two gene values linearly, meaning the trace lists need to be weighted. This trace-list weight of parent 1 (tlw_{p1}) and parent 2 (tlw_{p2}) on the offspring is calculated using the percentual distance of the new gene value x_{off} to the parents values x_{p1} and x_{p1}, as seen in Equation 4.10. This way, the influence of a parent with a gene value closer to the offspring gene is greater than the influence of a parent with a more distant value [114]. This is done for each gene separately.

$$
\begin{aligned}
tlw_{p1} &= 1 - \frac{\left|x_{off} - x_{p1}\right|}{\left|x_{off} - x_{p1}\right| + \left|x_{off} - x_{p2}\right|} \\
tlw_{p2} &= 1 - \frac{\left|x_{off} - x_{p2}\right|}{\left|x_{off} - x_{p1}\right| + \left|x_{off} - x_{p2}\right|}
\end{aligned}
\tag{4.10}
$$

$$
\begin{aligned}
tl_{off} = \{&(t, w \cdot tlw_{p1}) | (t, w) \in tl_{p1}\} \\
&\cup \{(t, w \cdot tlw_{p2}) | (t, w) \in tl_{p2}\}
\end{aligned}
\tag{4.11}
$$

Equation 4.11 calculates the new trace-list for the offspring genes. In practical terms, if the parents both feature trace tuples with the same traceID, they are combined to avoid excessively long lists. An example of a linear recombination crossover can be seen in Figure 4.7. The gene values are linearly combined, with the weights of 60% for parent 1 and 40% for parent 2. Similar to the gene values, the trace tuples in the trace lists are combined, scaled with the weights $tlw_{p1} = 0.6$ for parent 1 and $tlw_{p2} = 0.4$ for parent 2 [114]. If gene values are not combined but copied, like in UX, the new value will always be equal to one of the parents.

Mutation

For the mutation operation, we need to differentiate between value-independent and value-dependent mutation operators. Using value-dependent mutation operators, like GM [116] or PM [26], the trace-list also need to be updated. Similar to the crossover operation, the weight of the mutation tlw_m is assessed using the parents gene value x_p and the offspring gene value x_{off} [114] (Equation 4.12). The rest of the trace-list gets weighted with tlw_r.

$$tlw_m = \frac{|x_{off} - p|}{|p| + |x_{off} - p|} \qquad (4.12)$$
$$tlw_p = 1 - tlw_m$$

With these weights, the trace-list of the offspring individual for the mth mutation can be built as follows:

$$tl_{off} = \{(t, w \cdot tlw_p) : (t, w) \in tl_p\} \cup \{(-m, tlw_m)\} \qquad (4.13)$$

Similar to the trace-vector implementation, mutated genes get a new mutationID assigned, with in practical terms, are descending negative numbers. Figure 4.8 shows an exemplary mutation operation for a value-dependent mutation. Here, genes 2 and 3 are mutated, with the trace lists of the offspring (tl_{off}) still partially retaining the gene origin from the initial population. Value-independent mutation operators, on the other hand, do not need this update step. Here, the trace-list of the offspring can simply be replaced with a new one containing only the mutationID (Figure 4.8).

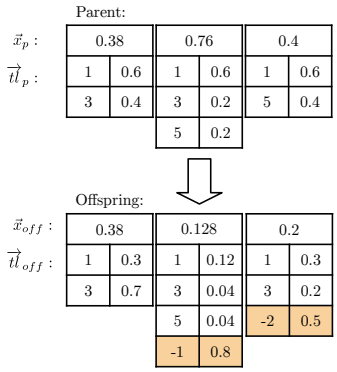


Figure 4.8: Trace-list example mutation.

Pros and Cons

This second encoding addresses the credit assignment for gene-combining crossover and value-dependent mutation. It allows for a precise heritage tracking for the most common representations and operators in GAs. This benefit comes with two drawbacks. First, it requires more memory, as more than one traceID needs to be kept per gene, which needs to be considered for very large population sizes. Second, the computational cost is higher, as we need to compute the weights and combine the trace lists, which is more costly than simply copying the heritage information. For this reason, we recommend only using this implementation when necessary. Finally, we acknowledge a limitation for variably sized genomes and permutation-based combinatorial problems. While the approach generally still works, we will lose the information about the position of the gene in the genome of the initial individual.

4.3.2 Evaluation Metrics

Contributors

The contributors $\mathfrak{C}$ are defined similarly to the previous implementation, as can be seen in Equation 4.14. If a traceID is on at least one trace-list of the population, it is considered to be contributing. The number of contributors is again defined as $|\mathfrak{C}(TL)|$.

$$\mathfrak{C}(TL) = \{t | (t, w) \in tl \text{ for at least one } tl \text{ in } TL\} \qquad (4.14)$$

Counting-Based Impact (I_C)

For the impact metrics, we define $\mathcal{W}(tl, k)$ as a function returning the amount of influence ω a traceID k has in the trace-list tl of an individual:

$$\mathcal{W}(tl, k) = \sum_{(t,w) \in tl} w \cdot \mathbf{1}(t = k) \qquad (4.15)$$

With this, the counting-based impact for a traceID k on a population can be calculated by summing the weights of all trace tuples with the traceID

$t = k$:

$$I_C(TL, k) = \frac{1}{n \cdot \mathfrak{z}} \cdot \sum_{i=1}^{n} \sum_{j=1}^{\mathfrak{z}} \mathcal{W}(TL_{i,j}, k) \tag{4.16}$$

Fitness-Based Impact (I_F)

The fitness-based impact metric I_F can equivalently be adjusted, as seen in Equation 4.17. We still use the fitness distance fd (Equation 4.3) as a measure of quality compared to the current population. The metric is again normalized.

$$I_F(TL, X, k) = \frac{\sum_{i=1}^{n} \sum_{j=1}^{\mathfrak{z}} \mathcal{W}(TL_{i,j}, k) \cdot \left(1 + fd\left(\vec{x}_i, X\right)\right)}{\mathfrak{z} \cdot \sum_{i=1}^{n} 1 + fd\left(\vec{x}_i, X\right)} \tag{4.17}$$

Entropy-Based Impact (I_E)

The idea of the entropy-based impact metric is to increase the influence of yet undecided decision variables. This is again done using the Shannon-entropy [117]. Previously, two genes had the same heritage if they both had the same traceID. Now, two genes are considered to have the same heritage if they feature the same elements in their trace-list. For this, we define:

$$\begin{gathered} tl_1 = tl_2 \\ \Longleftrightarrow \\ (\forall(t,w) \in tl_1 \exists(t,w) \in tl_2) \wedge (\forall(t,w) \in tl_2 \exists(t,w) \in tl_1) \end{gathered} \tag{4.18}$$

With this, the probability p of a trace-list tl appearing in a decision variable is:

$$p\left(TL_{i,j_{i=1}^{n}}, tl\right) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}\left(TL_{i,j} = tl\right) \tag{4.19}$$

Further, the set of distinct trace lists in the jth decision variable can be defined as:

$$\mathfrak{TL}\left(TL_{i,j_{i=1}^{n}}\right) = \left\{tl | tl \in TL_{i,j_{i=1}^{n}}\right\} \tag{4.20}$$

With this, the Shannon-entropy [117] for trace lists can be defined as:

$$H\left(TL_{i,j_{i=1}^{n}}\right) = -\sum_{tl \in \mathfrak{TL}\left(TL_{i,j_{i=1}^{n}}\right)} p\left(TL_{i,j_{i=1}^{n}}, tl\right) \cdot \log_2\left(p\left(TL_{i,j_{i=1}^{n}}, tl\right)\right) \tag{4.21}$$

And the entropy-based impact for the trace-list representation can be calculated as:

$$I_E(TL, k) = \frac{\sum_{i=1}^{n} \sum_{j=1}^{\mathfrak{z}} \mathcal{W}(TL_{i,j}, k) \cdot \left(1 + H\left(TL_{\cdot,j}\right)\right)}{n \cdot \sum_{j=1}^{\mathfrak{z}} 1 + H\left(TL_{\cdot,j}\right)} \tag{4.22}$$

Fitness and Entropy-Based Impact ($I_{F \cdot E}$)

Finally, the fitness and entropy-based impact can be calculated using both the fitness distance (Equation 4.3) and the entropy (Equation 4.21):

$$I_{F \cdot E}(TL, X, k) = \frac{\sum_{i=1}^{n} \sum_{j=1}^{\mathfrak{z}} \mathcal{W}(TL_{i,j}, k) \cdot \left(1 + fd\left(\vec{x}_i, X\right)\right) \cdot \left(1 + H\left(TL_{\cdot,j}\right)\right)}{\sum_{i=1}^{n} \sum_{j=1}^{\mathfrak{z}} \left(1 + fd\left(\vec{x}_i, X\right)\right) \cdot \sum_{i=1}^{n} \sum_{j=1}^{\mathfrak{z}} \left(1 + H\left(TL_{\cdot,j}\right)\right)} \tag{4.23}$$

4.4 T-EA for Multi-Objective Optimization

T-EA for MOEAs

The implementation of the T-EA is independent of the number of objectives. It can be used for MOEAs without modifications, as was first done in [115]. Furthermore, the I_C, contributors, and the number of contributors can be calculated without adjustment.

Limitations

However, two limitations apply, first for the metrics that are designed around the fitness, and second the metrics designed around the entropy. For the fitness-based metrics (I_F, $I_{F\cdot E}$), the application is limited as they are based on using a single fitness value. Solutions could be to calculate separate values for each objective, or to combine the objectives into a single value, for example, with weights. Furthermore, the entropy-based metrics (I_E, $I_{F\cdot E}$) are designed around the concept that the algorithm converges towards one single optimum, with each decision variable featuring one optimal value. As MOEAs try to find a diverse set of optimal solutions, the idea of these metrics is defeated. While it would be possible to use these impact metrics for the use of MOEAs, early tests using a single-objective (see Chapter 7) did not find a strong difference between the four proposed metrics. For this reason, the I_F, I_E and $I_{F\cdot E}$ were not updated for the use with MOEAs in this thesis.

4.5 Summary

Chapter Overview

In this chapter, the traceable evolutionary algorithm (T-EA) is presented. The algorithm allows to track the heritage of each gene throughout the evolutionary process. Two versions were proposed. Alongside this, evaluation metrics for the tracked heritage data were proposed. Finally, the use of the T-EA for MOPs was discussed.

T-EA Implementations

Two versions of the T-EA were proposed, the *trace-vector* and the *trace-list* implementation. The trace-vector version was the first implementation. It is intuitive to understand and fast to compute. However, it lacks support for genome-combining crossover and value-dependent mutation operators. Both of which are often used for real-valued vector representations. This was addressed with the trace-list implementation. Here, the single traceID is replaced with a list containing the heritage information of each gene. Through this list, the influence of multiple parents can be tracked precisely.

Evaluation Metrics

The main evaluation metrics used for the T-EA were the four ways to calculate the impact of an individual. In the I_C, the amount of genetic material, or influence in the trace lists, is summed. This way, we get an intuitive metric of the influence of an individual from the initial population. For the I_F, the idea was to scale the occurrence of genetic material in good fitness individuals higher, as they have a higher chance of survival in the evolutionary process. Similarly, the I_E biases towards decision variables that feature a diverse set of values and are not converged towards one singular value. Finally, the $I_{F\cdot E}$ scales both with the fitness and diversity. As an additional metric, we define contributors as individuals whose genetic material can be found in the current generation.

T-EA for MOPs

The T-EA can be used for MOPs without any modifications. The only limitations are found for impact metrics. Here, we only use the I_C metric, as the scaling factors cannot be directly applied for MOEAs.

5 Generating Test Individuals

One goal of this thesis is to understand which individuals, or genes, are influential in the search process of EAs (RQ 4 and RQ 5). This requires two things. First is the heritage tracking process of the T-EA, which was described in the last Chapter 4. The second is to set up the algorithm with individuals of specific properties, to then track and evaluate their influence. This chapter describes, how these individuals are generated, first for SOEAs, and second for MOEA.

5.1 Generating Seeds for SOEAs 49
5.1.1 Sphere Function 50
5.1.2 Rastrigin and Rosenbrock 51
5.2 Generating Seeds for MOEAs 52
5.3 Summary 54

5.1 Generating Individuals for Single-Objective EAs

Motivation

Individuals in EAs exist in two spaces, their genome $\vec{x}$ in the search space, and their fitness $f(\vec{x}_i)$ in the objective space. Usually, it is assumed that it is the good fitness individuals who lead the algorithms to their final result. This is reflected in the mating selection, as we typically bias better quality solutions to be selected more often, for example in the roulette wheel selection [17], or the tournament selection [23]. Little regard is given to the search space representation. However, one can easily imagine two individuals with the same fitness, but different genomes. The question arises if one of them is better at leading the algorithm towards the optimum than the other, due to their genetic material, or if both would perform the same, as they are measured with the same quality. In this thesis, we investigate two aspects of the genome: First is composition of the genome in terms of the *gene quality*, and second, the *closeness to the optimum* in the search space.

Gene Quality

For optimization problems with a single global optimum in the search space, the quality of a gene can be compared by measuring its distance to the corresponding optimal value. This is true for most SOPs used to benchmark EAs, for example, for all but one test function in the bbob [34] suite. Again, imagine two individuals $\vec{x}_1$ and $\vec{x}_2$, both with the same fitness $f(\vec{x}_1) = f(\vec{x}_2)$, but different genomes. One individual $\vec{x}_1$ might have a genome with a few very good (close to optimal) genes, but a few genes that are still far off their optimal value. The second individual $\vec{x}_2$ features an overall more mediocre genome, with no "good" but also no "bad" genes. A practical example for the sphere function would be $\vec{x}_1 = [2, 0]$. Another individual might have a genome in which genes are neither far off, nor close to their optimum, for example $\vec{x}_2 = [\sqrt{2}, \sqrt{2}]$. To test the differences of both individuals, we will generate ones with different amounts of optimal genes, setting the rest of the genome accordingly.

Closeness to the Optimum

Another scenario is two individuals with the same fitness, and the same number of optimal and not optimal genes, but still having a different distance to the optimum. Even with only one gene in the genome, there can be different genomes with the same fitness. Imagine two individuals $\vec{x}_1 = [\pm 3.05]$ and $\vec{x}_2 = [\pm 0.24]$, both again have the same fitness, but one

is much closer to the optimal value $x^{opt} = 0$ than the other[1]. We would assume, that $\vec{x}_2$ being closer to the optimum would also be better at leading the algorithm towards it. To test this, we generate all individuals both "close" and "far" from the optimum.

Fitness Targets

Finally, the fitness of an individual also influences how often an individual is selected. For this reason, we want to generate individuals with different fitness targets. Since these seeds are placed into an otherwise randomly generated population, their fitness targets depend on the population's fitness distribution. We will generate them as the "best" fitness individual, with half the fitness of the otherwise best individual in the population, as well as the fitness of the lower quartile, the median, and the upper quartile of the population. In this way, we can test the influence of a good, mediocre, and comparably bad fitness seed individual.

Configurations

To summarize, we will create the following eight seed individuals:

- Fitness targets ("best", "good", "medium", "bad")
- Percentage of Optimal Genes (0 %, 25 %, 50 %, 75 %)
- Distance to the optimum ("far", "close")

This leads to 16 different seeds to be generated. Tests will be done with three test problems: The Sphere function, the Rastrigin function [36] and the Rosenbrock function [37]. For the Sphere function, the seeds can be computed mathematically. This is not possible for the other test functions, for which the seeds need to be approximated. For this reason, the next section will first discuss how the seeds for the Sphere function are computed, followed by an explanation on how seeds for the other functions are created.

5.1.1 Sphere Function

This section is largely based on the author's publication [118].

Sphere function Properties

To investigate the effects of desired individuals, the benchmarking function needs to be solved in reverse, calculating the genome values for a given fitness. In this work, this is only possible for the Sphere function (recall Equation 2.7 in Section 2.4.1). It is one of the simplest benchmark functions, not featuring any local optima. It is also the only function where two individuals with the same fitness will always have the same distance to the optimum in the search space, meaning we cannot generate "close" and "far" individuals.

Simplifications

We need to simplify the problem in two ways:

1. All non-optimal genes will be set to the same value.
2. The non-optimal genes will be placed at the beginning of the genome.

The first simplification is made as there are a potential infinite number of possible values for a genome with the size of $d \geq 2$ given the target fitness f^{target}. For this reason, we select one not optimal value for all genes. Furthermore, we want to create individuals of different genome compositions, including both individuals featuring optimal genetic material but a few poor genes, and conversely individuals with all mediocre genes.

[1] This example can also be found visualized in the next pages in Figure 5.1.

Selecting only one not-optimal value results in these kinds of individuals. The second simplification is made as there are still $\binom{d}{\theta} = \binom{d}{\tau}$ different permutations of optimal and non-optimal genome values, assuming θ optimal and τ not optimal genes in a genome. This is true both for the Sphere and the Rastrigin function.

Calculating the Genome

The optimum of the sphere function is known at $f^{opt}_{Sphere} = 0$ and $x^{opt}_{Sphere} = [0, 0, ..., 0]$ This means the number of optimal genes in a genome can be altered by setting them to zero. Assuming a genome size of n, of which θ genes are optimal, the value of the τ remaining genes for a given target fitness f^{target} can be calculated systematically with Equation 5.1.

$$x_i = \sqrt{\frac{f^{target}}{\tau}}, 1 \leq i \leq \tau \tag{5.1}$$

The individual of a given target fitness can then be constructed by calculating the first τ genes with Equation 5.1 and setting the remaining θ genes to zero. As an example, the value of the not optimal gene for an individual with the target fitness of $f^{target} = 2$ and two not optimal genes ($r = 2$) would be $\sqrt{2/2} = 1$, resulting in the genome $x = [1, 1, 0, 0]$. The same example with three optimal genes would result in the genome $x = [\sqrt{2}, 0, 0, 0]$.

More Specifics of the Sphere Function

The Sphere function is one of the easiest single-objective benchmark. It also features a few properties which should be mentioned here. It is an unimodal function without local optima. This also means that individuals who have the same fitness, will always have the same distance to the optimum. For this reason, we cannot generate "close" and "far" individuals for the sphere function. Second is that the quality of the τ remaining genes is directly proportional to the amount of optimal genes in the genome. This means that individuals who have a high number of optimal genes feature a few genes that are very far away from their optimal value, dragging down the fitness of the otherwise good individual. This might be desirable for crossover operators which copy the genome values directly from the parent to the offspring, as the already optimal value is kept. If no genes are optimal, the distance of each gene to their respective optimal value is lower. In theory, this might be more desirable for crossover operators that combine genome values instead of copying them. However, the evaluation in Chapter 8 shows a different result.

5.1.2 Rastrigin and Rosenbrock

The majority of test functions do not allow the genome to be calculated in reverse, like for the Sphere function. To run tests with more complex test functions, we propose a second way, by formulating the construction of the genome as an optimization problem.

Simplifications

We again use the two simplifications made to set all non-optimal genes to the same value, and to put the optimal genetic material always at the back of the genome. This is first to simplify the problem by restricting the number of possible solutions. Second, for the Rosenbrock function it is necessary to specify the used permutation beforehand, as the values of neighboring genes affect each other.

Formulating the Optimization Problem

Generating the non-optimal gene value is an optimization problem itself, and done for each problem and seed configuration separately. With the two simplifications made, we try to find a value $x^{\neg opt}$ so that we can construct a genome of an individual i with τ not optimal genes and θ optimal genes as $\vec{x}_i = [x_{i,1}^{\neg opt}, \ldots, x_{i,\tau}^{\neg opt}, x_{i,\tau+1}^{opt}, \ldots, x_{i,\theta}^{opt}]$. For a given fitness target f^{target}, we can formulate this as an optimization problem to construct $\vec{x}$ as:

$$f_{\vec{x}}\left(x^{\neg opt}\right) = \left| f^{target} - f\left(\left[x_{i,1}^{\neg opt}, \ldots, x_{i,\tau}^{\neg opt}, x_{i,\tau+1}^{opt}, \ldots, x_{i,\theta}^{opt}\right]\right)\right| \quad (5.2)$$

To optimize this, the scipy.optimize.fsolve() function [119] is used, which internally uses a variation of Powell's hybrid method. The start points are created as 100 samples equally distributed in the bounds of the problem. This results in 100 approximated gene values. As these genes are not exactly fitting f^{target}, every solution with a distance greater than $1 \cdot 10^{-9}$ is discarded, only leaving very close solutions. From the remaining solutions, two are now selected based on their distance to the optimum in the search space. The solution with the smallest Euclidean distance to the optimum is selected as the "close" seed, and the solution with the largest distance as the "far" seed.

Example

With this methodology, seeds with nearly identical fitness but different distances to $\vec{x}^{opt}$ can be generated. It is important to note that the number of different gene values obtained by this method, as well as the differences in the distance to the search space optimum, highly depend on the benchmarking function used. Furthermore, for some fitness targets, we cannot create multiple solutions. An example for the Rastrigin function can be seen in Figure 5.1. The figure shows the fitness landscape of the Rastrigin [36] function with one decision variable (n= 1), with the blue line shows the fitness landscape and the red line the target fitness of 10. Suppose we want to generate an individual with the $f_{rastrigin}^{target}(x_i) = 10$, represented by the gray line. The possible values for $\vec{x}_i$ are at the intersection between the line and the fitness landscape. This leaves 14 possible gene values, with $\vec{x}_i \approx \pm 3.05$ being the furthest and $\vec{x}_i \approx \pm 0.24$ the closest to the optimum. Assuming $d = 4$ and $p_{opt} = 75\%$ the resulting genome for the "far" seed would be $\vec{x}_i = [3.05, 0, 0, 0]$. The same seed configuration with $p_{opt} = 50\%$ would result in $\vec{x}_i = [2.06, 2.06, 0, 0]$.

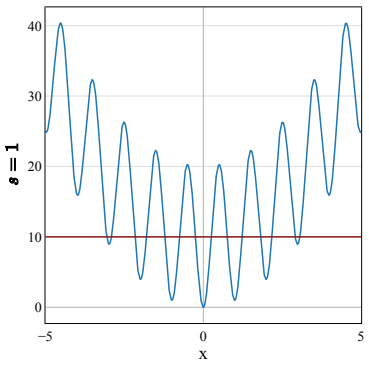


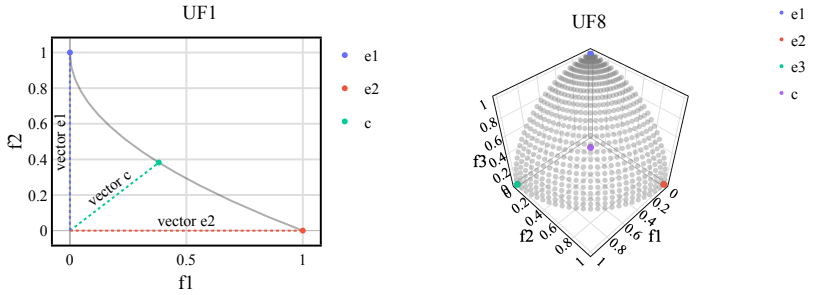
Figure 5.1: Fitness landscape of Rastrigin with s= 1 and an $f^{target} = 10$.

5.2 Generating Individuals for Multi-Objective EAs

This section is largely based on the author's publication [120].

Motivation

For MOPs, we compare the search behavior of specific individuals for different types of benchmarking problems. Specifically, we focus on benchmarks with *easy* and *complicated* PSs [41]. Recalling Section 2.4.2, Li and Zhang [41] introduced this classification. They criticize problems featuring decision variables which converge only to one value for all individuals as ones with an easy PS. This is true for earlier benchmarking



(a) Seeds UF1.

(b) Seeds UF8.

Figure 5.2: Examples seed locations on the PF for two- and three-objective problems.

problems, like ZDT [38], DTLZ [39], or WFG [46]. Problems without this property are deemed to have a "complicated" PS. In this thesis, we test whether an algorithm's convergence behavior differs between these two types of problems. Furthermore, it is evaluated if there is a difference in seeding good known solutions, but from different parts of the PF.

Defining Good Individuals in MOEAs

For MOEAs it is very difficult to define which individuals, or even which genes, are good. For the single-objective case, there exists only one optimum $\vec{x}^{opt}$, so the quality of two genes can be compared by measuring their distances to the optimal value. For MOPs, we find a set of optimal values (the PF/PS), and multiple values can be optimal, meaning a direct comparison is not possible.

Limitations for Seeding MOEAs

Without this classification, we cannot differentiate between optimal and not optimal genes. The only solutions, which are known to feature optimal, and therefore good, genetic material, are Pareto-optimal solutions. For this reason, we chose to use individuals directly from the PF. While this is unrealistic for real-world applications, it serves as a good example for our purpose of evaluating the search dynamics of MOEAs.

Generating MOEA Seeds

In order to systematically analyse the impact of seeded individuals, we concentrate on test problems with known PFs and PSs. This approach has already been done in the past, in research works on seeding MOEAs. Similar to [121] and [122], we select an extreme solution for each objective, and one center solution in the mid-point of the PF, as seed individuals, i.e., two ($e1$ and $e2$) or three ($e1$, $e2$ and $e3$) extreme seeds for the two- and three-objective problems respectively, plus one center seed (c). The seed individuals are computed at the intersection point between a reference vector and the PF as shown in Figure 5.2a and 5.2b.

Reference Vectors

These vectors can be defined by the nadir and ideal points of the PF. The ideal point is defined as a vector containing the best possible value for each objective ($ideal^{UF1} = [0, 0]$ for UF1 test problem in Figure 5.2a). The nadir point is defined as a vector with the worst possible values found for each objective on the PF ($nadir^{UF1} = [1, 1]$ for the UF1). With these two points, we can define the reference vectors for two-objective problems:

$$ref\_vector_{e1} = (nadir, [ideal_0, nadir_1])$$
$$ref\_vector_{e2} = (nadir, [nadir_0, ideal_1])$$

and three-objective problems:

$$ref\_vector_{e1} = (nadir, [ideal_0, nadir_1, nadir_2])$$
$$ref\_vector_{e2} = (nadir, [nadir_0, ideal_1, nadir_2])$$
$$ref\_vector_{e3} = (nadir, [nadir_0, nadir_1, ideal_2])$$

where $ref\_vector_c = (nadir, ideal)$. The final seed individuals are chosen from a sample of N points ($N = 1005$) as the closest to the respective reference vector.

5.3 Summary

Chapter Overview

In this chapter, two methods for generating seed individuals of specific properties were discussed, one for single- and one for multi-objective EAs. Generating such individuals will be important for testing and evaluating which individuals are influential in the search process.

Generating Individuals for SOEAs

For SOEAs, we aim to test individuals of specific properties. For this, three aspects are altered: The fitness, the genome, and the distance to $\vec{x}^{opt}$. Fitness is an important factor in survival. For this reason, the fitness targets are derived from the population these generated individuals are seeded into. We use the fitness targets of the seed as the best individual, and as the fitnesses of the upper quartile, median, and lower quartile. For the genome, we vary the amount of good (optimal) genes in four targets: 0 %, 25 %, 50 %, and 75 % of optimal genes. This way, we create individuals that have an overall mediocre genome (p^{opt}= 0%), and genomes with some very bad, but also a lot of very good genes (p^{opt}= 75%). Finally, we generate individuals of the same fitness and with the same amount of optimal genes, but different distances to the optimum. Here, we select individuals as close and as far away as possible from $\vec{x}^{opt}$. For some test problems, these seed individuals can be calculated, given some simplifications. For other test problems, generating these seed individuals was formulated as an optimization problem in itself.

Generating Individuals for MOEAs

Generating similar individuals for MOEAs proved difficult, mainly as it is not possible to measure the quality of the genome, or a single gene, in the same way as for SOEAs. This definition is further complicated by different features of PSs, which may also lead to large differences in the population dynamics. For this reason, it is not possible to generate and test individuals the same way as for SOEAs. Therefore, to research towards influential individuals in MOEAs, we selected the only individuals that we know contain optimal genetic material. These are individuals from the PF of the problem. Similar to previous works in this area [121, 122], we select individuals from different locations on the PF. We select edge solutions, optimal in one objective, and centre solutions, as well as a combination of seed locations.

Part II

Single-Objective Population Dynamics

Related Work 6

This chapter explores the related works of population dynamics in SOEAs. An overview of existing tools and approaches for this was provided in Section 3.3. First, population initialization techniques are briefly discussed, as they can be seen as a top-down approach to generate good initial solutions. Following this, the results from evaluating the population dynamics of SOEAs are presented. This is again separated into findings on the population level, the individual level, and the gene level. A short summary of all findings is provided in the final section of this chapter.

6.1 Population Initialization 57
6.2 Population Dynamics . . 58
6.2.1 Population Level 58
6.2.2 Individual Level 59
6.2.3 Gene Level 62
6.3 Summary 63

6.1 Population Initialization

Works on finding a good initialization strategy for EAs can be considered related, as they also try to find good genetic material to start the evolutionary process. Here, a short overview on this topic is given, as it can be seen as the top-down equivalent to our research for population dynamics. Two subcategories are highlighted: First is using more sophisticated strategies to *sample* the search space. Second is *seeding* solutions into the initial population of the algorithm.

Sampling

Sampling, in the context of EAs, refers to the process of generating a set of solutions in the search space of a given problem. Different techniques and strategies for this exist. Uniform random sampling can be considered the default initialization strategy, used in most works. Each gene in the genome gets assigned an independent random value as the starting point. However, for large dimensional search spaces, such solutions tend to be generated at the boundary of the search space. For large genome sizes, it is highly likely that at least one decision variable gets assigned a value close to its upper or lower bounds [123]. Other approaches to sample the initial population with a better distribution than uniform sampling exist and have been applied to EAs. This includes, for example, the also very popular LHS [22], gap-search sampling (GSS) [124], opposition-based learning (OBL) [125], or quasi-random numbers such as Sobol sampling [126]. Reviews and performance comparisons for different sampling techniques have also been conducted [127–130]. While one sampling strategy might perform better in some cases, no single approach consistently outperforms the others over all test problems, or for certain algorithms. Furthermore, these studies do not answer *why* certain techniques perform better on certain problems or algorithms.

Seeding

Seeding refers to inserting a few known good solutions into the initial population. Several strategies for seeding the initial population exist. One is inserting expert knowledge, which is especially relevant for real-world problems. An example of this can be found in [131], where expert knowledge is seeded into the initial population of GP algorithms, comparing different seeding techniques. Another approach is to generate solutions using a different optimization technique. Saavedra-Moreno et al. [132],

for example, use a greedy algorithm to create initial solutions for optimizing wind turbine positioning in wind farms. For problems where random sampling often creates infeasible solutions, seeding can be used to insert feasible individuals. Kazemzadeh Azad [133], for example, use this approach in the optimization of steel truss structures. All of these examples show performance using their respective approaches. However, it is difficult to derive more general conclusions about the search behavior of EAs. This is for two reasons. First is that seeding is often applied to more specific areas, with a lack of widespread evaluations over multiple problems. Second, most works only evaluate the performance improvements and not the underlying population dynamics when seeding the algorithms, providing only speculations about why the performance was improved.

The Answer is in the Search Space

The initial population has been the focus of many studies in the past. While good progress has been made in the field, the question of *why* certain sampling or seeding techniques work better is typically neglected, or speculated. The lack of a clear guideline on which initialization technique to use for a given problem or algorithm configuration reflects that we don't truly understand what a good initial population consists of. We believe evaluating the dynamics in the search space to hold valuable information on why certain solutions, or parts of solutions, support the evolutionary process.

6.2 Population Dynamics

6.2.1 Population Level

STNs for NEAT

In the population level, STNs are the most used tool for population dynamics evaluation. Sarti and Ochoa [76] evaluate the NEAT [77] system, which simultaneously evolves the topology and weights of NNs, using STN. For this, they repeat the benchmarks used in the original NEAT paper [77], using the XOR and double-pole balancing problems. They tested two algorithmic configurations, one with and one without crossover. In contrast to the original work, they found the algorithm without crossover to perform better. In the STNs of both algorithms, it could be shown that crossover did increase the diversity of the population. However, this did not lead to a better performance, as the search trajectories of the algorithm without crossover were shorter and more direct.

STNs for Combinatorial Optimization

Two works use STNs to evaluate the search process of combinatorial problems. First, [73] evaluates the population dynamics of five continuous and five problem instances of the combinatorial p-median problem. Three algorithms were used, iterative local search, particle swarm optimization (PSO), and a version of DE. The case studies on the continuous and combinatorial problems could show differences in search dynamics of the three algorithms used, namely in showing frequently visited areas of the search space, or general differences in how the algorithms traversed the search space. Second, [74] research the dynamics of the cyclic bandwidth sum problem, which is permutation encoded. The performance and search characteristics of a memetic algorithm and a dynamic multi-armed bandit memetic algorithm are compared. Evaluations could show

that the multi-armed bandit version did have a higher success rate than the standard version. The search trajectories of the algorithm revealed that it was more direct in the search for local/global optimum, and better at avoiding deceptive areas compared to the standard memetic algorithm.

Dimensionality Reduction Techniques

Recently, Mégane et al. [2] predicted the blood glucose levels of diabetes patients, comparing four different variants of structured grammatical evolution. Besides performance findings, the populations of the four variants were visualized in the search space using t-SNE. It could be shown that the float-based representation performed significantly better than the standard variant. The float-based representation also explored the search space more thoroughly, finding the best solutions in a different area than the standard variant. The paper only shows this for a brief evaluation for one patient. However, it clearly proves the potential for comparing the results between different patients in more detail, towards a more in-depth understanding of why certain predictions work well for some people, but not for others.

6.2.2 Individual Level

ETVs and Power Laws

Genealogical evaluations on the individual level are among the oldest and most used techniques. ETVs were proposed as a measure of impact of an individual on its following generations [14]. Several studies on this measure have been done to evaluate the population dynamics of EAs [14, 15, 61, 134, 135]. Or more specifically, if the probability of an individual to impact the following generations fits a power law distribution or not. In [61], Whitacre, Sarker, and Pham conduct tests for this on a wide variety of algorithmic configurations, testing a number of different crossover, mutation, and selection operators on 13 different SOPs. Besides the traditional parametric EA, they also used a spatially constrained mating mechanism, where only individuals in the same neighborhood, defined by a radius, can reproduce. Their tests showed two main results. First, the probability of an individuals impact on an EA was found to fit a power law distribution. This finding was largely insensitive to changes between the test problems and the algorithm. From this, they concluded that only a small number of individuals will have significant impact on future population dynamics. Secondly, only two algorithmic settings were found to change this power law distribution. Namely, introducing new individuals without any historical links to the previous population, and the use of specially constrained mating. However, a more recent study replicating ETV tests with different benchmark problems and a different algorithmic configuration did not find the power law distribution to be a general rule [134]. Martins and Nascimento studied these power law dynamics for the TSP in [135], finding that the power laws found by [61] hold for this combinatorial problem. They used the q-exponential distributions, a generalized form of the power law distribution, arguing that previous found deviations can be modelled using this approach. They also showed that the use of elitism significantly influences the power law scaling factors and deviations observed. While the works show differences in their result, one overall pattern emerges. The discussed works [61, 134, 135] imply that most individuals in traditional parametric

GAs have a negligible impact on the following populations. This is similar to the finding from McPhee and Hopper for GP [13].

Number of Ancestors

Collier and Wineberg [64] investigate the number of ancestors in GAs and EAs using genealogy trees, which in their work they call lineage trees. They researched the number of ancestors on the best individual found for three test problems: MaxOnes, Rastrigin [36], and Whiteney's F8F2 [136]. This number of ancestors was found to be similar for all three test problems. This was throughout the generations, with first a region of exponential growth in the early generations, followed by a region of slowing (diminishing) growth, finally settling in a region of stability in the later generations [64]. Around 35 % of individuals from the initial population showed an ancestral relationship for all three test problems [64]. Besides the observation, they also propose three models to estimate the number of ancestors. Their best model considers an exponential growth, with inbreeding and duplicate solutions as limiting factors. However, it has to be noted that in our experiments, we do not find such rules, as will be discussed in Chapter 10. While the work can be considered to be an interesting starting point in evaluating heritage information in GAs, we need to set their results into context. First, there is the question of true heritage, as a genealogical heritage does not guarantee any genetic material from the parent to be found in the offspring [13]. Furthermore, they only experiment with a fixed population size of $n = 200$, a fixed genome size for each of the test problems, using only a single (unspecified) crossover and mutation operator and tournament selection of size two. None of these parameters were considered in their original model. Increasing the selective pressure in the tournament size, or changing algorithmic operators, will have an impact on the number of ancestors [114, 115]. Bäck and Schwefel [16] note that different algorithms emphasize on different features for a successful evolutionary process. With the wide spectrum of possible algorithmic configurations in EAs it remains questionable if concrete rules about the population dynamics can be found.

Evaluating the Genealogy of whole Algorithmic Runs

As mentioned in Section 3.3.2, one challenge when tracking the genealogy of EAs is the amount of data generated. One emerging way to track the heritage of complete test runs is the use of graph databases. Donatucci, Dramdahl, and McPhee [92] pioneer this and conduct a proof-of-concept evaluation on a canonical GP implementation approximating $sin(x)$. Their data demonstrated three things. First, the heritage of the final generation can be traced to only a few individuals in the evolutionary process. Second, the effectiveness of the mutation operator[1] dropped significantly in the early generations. Third, they could show the significance of the root parents[2] lineage. Later, McPhee, Donatucci, and Helmuth [62] also evaluate and compare the differences between lexicase and tournament selection for a GP algorithm on the replace-space-with-newline problem, an introductory programming benchmark [137]. Their evaluation showed that for lexixcase selection, most of the final population could be traced back to only one parent (934 of 1000 individuals of an exemplary testrun). Tournament selection was found to have a much lower selective pressure and more individuals contributing to the final result. However, the authors also note the challenge of visualizing and evaluating the

[1] Effectiveness in terms of producing better fitness offspring than the parents.
[2] The root parent is the parent passing the root node to the offspring.

large amount of heritage data over all the independent test runs. For this reason, evaluations were conducted on exemplary testruns and only one test problem.

Genetic Building Blocks in GP

Burlacu, Affenzeller, and Kommenda evaluate the effectiveness of operators in GP [100]. They used genealogical data and traced genetic fragments (subtrees of solutions) with the approach developed in [98, 99]. A canonical GP and one with an updated offspring selection was used on the poly-10 and Vladislav-8 benchmarks. Their experiments showed the best solutions to be a result of only a few critical crossover and mutation operations, as most operations did not lead to a fitness improvement. Furthermore, they find that most of the final population is built off only a few individuals from the initial population. In a more recent work using the same evaluation techniques as in [98, 100], Burlacu, Yang, and Affenzeller [138] extend the previous tests, comparing the population dynamics of three GP algorithms. A canonical GP, a GP with an age-layered population structures, and a GP with an updated offspring selection are tested on five benchmark problems. Using genealogical data and analysis of traced genetic fragments, they could show differences in the population dynamics between the three algorithms. Besides the algorithmic configurations, different test problems also lead to differences in search dynamics. Similar to previously discussed works, they also find just a small percentage of individuals (1.7 % - 8 %) contributed genetic material to the final generation.

Diversity Control with Lineage IDs

Zhu and Liu [104] track the lineage of the initial population by attaching IDs to individuals in a GA. The goal was a control approach to maintain population diversity by adjusting recombination probabilities. For this, they conduct tests of the population dynamics in GAs for the TSP problem. They measure the number of contributors as the number of unique heritage IDs, the standard deviation of the population fitness and a diversity measure for the population in the search space and objective space. Their results also suggest only a few individuals to contribute to the final population. However, as mentioned previously in Section 3.3.2, the ancestral IDs in the recombination step are inaccurate in terms of credit assignment. They pass the ID of one parent to one child, and the ID of the other parent to the other child, meaning only the influence of one parent is traced. This could mean the number of relevant parents is underrepresented. Furthermore, they only repeat each test configuration 10 times. So while their results indicate similar findings to other works discussed here, they need to be viewed in the context of the test setup.

Controlling Exploration and Exploitation in DE

Liu et al. [106] propose a parameter control approach to control exploration and exploitation in DE using the metrics developed in [96], which are based on genealogical information. Alongside the genealogy of the population, the distance between the parent and offspring individual is also tracked. Again, we want to note that in this approach only the dominant parent is tracked, with the influence of the other parent in crossover being discarded. If parent and child are similar, the children are deemed to stem from exploitation. If they are more distant, the children are deemed to stem from exploration. Using this distinction and the genealogical data, they create metrics to quantify exploration and exploitation in EAs in [96]. Using these metrics, they develop a parameter control approach to manage the population dynamics in DE for six of the Sovova's mass transfer model problems, a real-world problem

from the chemical industry. They test five different parameter control strategies against each other, but do not compare against a canonical DE algorithm. Performance and population dynamics with the exploration and exploitation metrics of [96] are evaluated. They conclude that balance exploration and exploitation throughout the generations, and a higher selective pressure on exploitation[3] showed the best performance. In terms of algorithmic design this makes sense, as it should be desirable to frequently explore new regions of the search space, focusing exploitation on the best individuals found so far.

Genealogy as a Diversity Measure

Dolson, Banzhaf, and Ofria [101] also investigate the idea of using genealogical information as a diversity measure for EAs. They derived two metrics from ecological theory in biology: First, *richness*, measuring the shortest path between two individuals in the genealogy tree. Second, *divergence*, measuring the distance to all other individuals in the current population. In initial experiments, it could be shown that these diversity measures produce different results to the more conventional diversity measures on the genomes of fitness in the population. Later, Hernandez, Lalejini, and Dolson explore this idea further in [139]. They use the same metrics [101], with the goal of comparing population diversity of five different selection mechanisms. Tests were run using four sets of benchmark problems, two for GP and two for GAs. They also find genealogical diversity to produce different results to the conventional genome-based diversity measures. Furthermore, they note that in some computational experiments, genealogical diversity was a stronger driver for the populations success in terms of performance. These results were, however, dependent on the problem and selection operators used. A third work by Shahbandegan et al. [140] conducted similar tests, this time using three selection schemes on five benchmark problems. Overall, the obtained results are similar. This time, the authors find that results are more mixed with multi-modal problems, featuring multiple global optima.

6.2.3 Gene Level

Gene Heritage Findings in GP

The largest body of literature for evaluating the population dynamics of an EA exists in the field of GP. McPhee and Hopper [13] track the heritage of individuals using IDs, similarly to the T-EA. With their approach, they evaluate the population dynamics of a steady-state GP algorithm on three benchmarking problems. In their experiments, they find only a very small number (1 %) of the nodes in the initial population to survive to the final result. Genetic material of less than seven of initially 500 individuals survived. Furthermore, in 90 % of their test runs, a single common ancestor for the whole final generation could be identified. Later, Daida et al. [103] use a similar approach to evaluate a more traditional GP configuration. Overall, they reached a similar conclusion to [13]. The number of survivors and contributing individuals was found to be a bit higher, but still only a small percentage of the initial population. Additionally, they note the influence of the selection mechanism and problem difficulty on the population dynamics in GP.

[3] Meaning only the best individuals are used for exploitation.

The Initial Population in GP

In a second work, Daida [141] ask the question of what makes an initial population good in GP. For this, they conduct computational experiments on the binomial-3 problem, which they tuned with one "easy" and one "hard" configuration. Additionally, they compared two selection mechanisms, tournament selection, and fitness proportionate (/roulette wheel) selection. Their results show far fewer individuals survive for tournament selection than for fitness proportionate selection. Even though this indicates a lower overall diversity, they actually find tournament selection to perform better. This contrasts with later works, claiming genealogical diversity to be a predictor for success in GP [139, 140], arguably just under certain circumstances. Daida [141] concludes that it is highly ranked (meaning good fitness) individuals in the initial population which pass their genetic material to the final population. The theory is that it depends on a supply of good genetic material in these individuals for the algorithm to be successful.

6.3 Summary

This section briefly summarizes the overall findings for population dynamics in SOEAs. Table 6.1 presents an overview of this literature, at which level of population dynamics they are aimed at, and for which type of EAs it was applied to.

Number of Surviving Individuals

For GP, several studies have found that only a few individuals from the initial population contribute to the final result, this is both using gene heritage tracking [13, 103] and genealogical data [62, 63, 92, 94, 100, 138]. Moreover, it is often argued that the root parent has a significant role in the crossover of GP [13, 92]. Similar findings were also made for GAs and EAs. A few studies show only a small percentage of individuals from the initial population to contribute to the result [14, 15, 61, 104, 105, 134, 135]. However, we want to note that these studies are done on genealogical data, so the credit assignment problem has to be considered.

What Influences the Population Dynamics?

Unsurprisingly, different algorithms, like PSO and DE, have been shown to result in different search dynamics [2, 73, 74]. However, changes in operators were also shown to produce differences in the search dynamics. This has been shown for different mating selection schemes [98, 99, 101, 139, 140], offspring selection [100, 138], or using or not using crossover [76]. Several studies also point out that the fitness landscape of different test problems also affect the population dynamics. This is both for GP [101, 103, 139, 140] and for GAs [101, 140].

Diversity

Some papers claim the loss of diversity to be a problem [13], some try to increase genealogical diversity deliberately in their algorithmic design [109, 110], while others find a lower genealogical diversity to perform better [141]. Some papers claim genealogical diversity to be a good predictor of algorithmic performance under certain circumstances [139, 140].

Research Gap

Several studies on the population dynamics of SOEAs already exist, as can be seen in Table 6.1. However, we can identify several open areas in this field. First, most approaches are aimed at the individual level, which inherently suffers from the credit assignment problem. We see only a few works focused on the gene level [13, 103, 141], all of which are focused

on GP. In general, there are only very few general findings across the literature. It is difficult to compare the mostly specific case studies of each work. In this thesis, we aim specifically at the gap of gene-level population dynamics in GAs.

Table 6.1: Literature overview of papers evaluating population dynamics of SOEAs, excluding the contributions of this thesis.

Author(s)	**Work**	**Year**	**Level**	**Description / Contribution**	**Algorithm**
McPhee and Hopper	[13]	1999	Gene	Evaluating number of contributors	GP
Burlacu et al.	[94]	2013	Individual	Visualizing genealogical data	GP
Collier and Wineberg	[64]	2007	Individual	Counting umber of ancestors	GA+EA
Whitacre, Sarker, and Pham	[61]	2009	Individual	Power law dynamics of ETVs	EA
Martins and Nascimento	[135]	2022	Individual	genealogy, metric (ETVs)	GA
Fernandes et al.	[134]	2019	Individual	Power law dynamics of ETVs	GA
Burlacu, Affenzeller, and Kommenda	[100]	2015	Individual	Genealogical data and genetic fragments	GP
Burlacu, Yang, and Affenzeller	[138]	2024	Individual	Genealogical data and genetic fragments	GP
Donatucci, Dramdahl, and McPhee	[92]	2014	Individual	Visualizing genealogical data	GP
McPhee, Donatucci, and Helmuth	[62]	2016	Individual	Visualizing genealogical data	GP
Zhu and Liu	[104]	2004	Individual	Tracking individuals with ancestral IDs	GA
Liu et al.	[106]	2013	Individual	Quantifying exploration and exploitation with genealogy	DE
Dolson, Banzhaf, and Ofria	[101]	2018	Individual	ecological theory in EAs	GP+EA
Dolson et al.	[102]	2020	Individual	extension to [101]	GP+EA
Hernandez, Lalejini, and Dolson	[139]	2022	Individual	Measuring genealogical diversity	GP+GA+EA
Shahbandegan et al.	[140]	2022	Individual	Measuring genealogical diversity	GP
Daida et al.	[103]	2004	Gene	Repeating tests of [13]	GP
Daida	[141]	2005	Gene	Investigating selection and surviving genetic material	GP
Narvaez-Teran, Ochoa, and Rodriguez-Tello	[74]	2021	Population	STNs for combinatorial problems	memetic GA
Mégane et al.	[2]	2025	Population	Search space visualization with t-SNE	GP
Whitacre, Sarker, and Pham	[61]	2009	Individual Level	ETVs and power laws	EA
Whitacre	[15]	2009	Individual Level	ETVs and power laws	EA
Martins and Nascimento	[135]	2022	Individual Level	genealogy, metric (ETVs)	GA
Fernandes et al.	[134]	2019	Individual Level	genealogy, metric (ETVs)	GA

Influence of Operators on the Population Dynamics of SOEAs

7

This chapter is largely based on the author's publication [114].

In this chapter, we provide an initial overview of the capabilities and potential knowledge gain from the T-EA in a first proof of concept. For this, we evaluate the differences in population dynamics when using different crossover and mutation operators.

7.1 Motivation and Overview 67
7.2 Test Setup 68
7.3 Evaluation 68
7.3.1 Single Configuration . . . 69
7.3.2 Operator Comparison . . 74
7.4 Summary 78

7.1 Motivation and Overview

Motivation

EAs are solving complex optimization problems inspired by biological evolution [142]. A set of individuals, which represent solutions to the given problem, is used as the starting point for a search using mutation, recombination, and selection to generate new individuals for a number of generations until a stopping criterion is met. While we understand each of these steps in the evolutionary process, the generation of tens of thousands of offspring individuals over the course of many generations quickly becomes incomprehensible for humans. Related works on this topic were discussed in the previous Chapter 6. Here, we already saw interesting results. Genealogical evaluations, for example, showed that most of the initial population in GAs could be tracked back to only a few individuals in the initial population [14, 15, 61, 104, 105, 134, 135]. However, genealogical evaluations often suffer from a problem of credit assignment, and gene-level approaches have previously not been proposed for GAs. Better understanding the search process of EAs on a gene level is the foundational motivation of this thesis. Because of this, the T-EA was presented as a tool to track gene heritage throughout the evolutionary process, allowing for the evaluation of the gene-level population dynamics. To gain a first understanding about the gene-level population dynamics of EAs, we test the effects of different crossover and mutation operators as a proof of concept.

Gene Copying vs. Gene Combining Crossover

In Chapter 4, we differentiated between gene-copying and gene-combining crossover operators. In short, gene-copying crossover operators, like UX, copy the full genome value from one parent directly to the offspring. Here, only one parent individual can exist for each gene. In gene-combining crossover, on the other hand, the gene values of the parents will be combined. For example, through linear recombination with the linear crossover (LX) operator. This way, two or more parents can influence an offspring's gene. This differentiation is not only relevant when designing a heritage tracking approach, but also in how the algorithms can use the parents' genomes to create their offspring. In this chapter, we want to investigate the differences between these two classes of operators.

Value Dependent vs. Value Independent Mutation

Similar to crossover, there also exist two classes of mutation operators. Value-independent mutation changes the gene values randomly without regard to the previous value. Value-dependent mutation, on the other hand, changes a gene based on its previous value. Here, we designed the T-EA to retain the parents influence. For this first evaluation of heritage

data in this thesis, we also want to compare the differences found for these two classes of mutation operators.

Chapter Overview

This chapter offers a first investigation into the gene-level population dynamics of GAs, using the heritage data of the T-EA. For this, we evaluate the effects found for different crossover and mutation operators. That is gene-combining and gene-copying crossovers, as well as value-dependent and value-independent mutation. The next section describes the test setup in more detail. The following evaluation is split into three parts. First, a single test configuration is evaluated in more detail. Second, the differences in population dynamics between the operators are discussed. Third, the differences between the four proposed impact metrics are discussed. The chapter closes with a summary of the findings and an outlook on future research directions.

7.2 Test Setup

Test Problem

The Rosenbrock problem [37] is used as specified in Equation 2.9 (see Page 14). It was chosen as a proof of concept, as it is not separable and features local optima. A genome size of $\mathfrak{d} = 10$ is chosen. Its optimum is known at $f^{opt}_{Rosenbrock} = 0$ and $\vec{x}^{opt}_i = [1, \ldots, 1]$.

Initial Populations

To keep the results for the impact metrics comparable, each test has to be run with the same initial population. For this, we select three initial populations from a sample of 1000 randomly generated ones, sorted by their median fitness. The populations with the corresponding *worst* (population 1), *median* (population 2), and *best* (population 3) median fitness were chosen. The fitness values for all individuals of these three populations can be found in Figure 7.1.

Algorithmic Setup

Each test configuration was initialized with 20 individuals and a genome size of 10. Tournament selection with a tournament size of 2 is used for breeding selection. The tests were each run for 75 generations, each repeated 31 times. We use two crossover operators. First, UX represents the gene-copying crossover operator, and LX [143] the gene-combining crossover. Similarly, two mutation operators are chosen. Uniform mutation (UM) for value-independent mutation, and GM [116] as a value-dependent operator. Both are configured with a mutation probability of 1%. The GA implementation of the pymoo framework version 0.6.1.5 [144] is used to run the tests. This results in four algorithmic setups.

Evaluation Metrics

The performance of the algorithms is not a big concern for our evaluation. Nevertheless, the best fitness of each individual is tracked. For evaluating the population dynamics, the T-EAs is used, with the four impact metrics I_C, I_F, I_E, and $I_{F \cdot E}$.

7.3 Evaluation

The evaluation is split into two parts. First, a single test run is evaluated in more detail to gain an overview of how the T-EA can be used for more specific evaluations. Second, we evaluate the four test setups on a wider scale to find the differences in convergence behavior between the two crossover and two mutation operators used.

7.3.1 Single Configuration in Detail

As a first investigation into the heritage data of the T-EA, the algorithmic configuration using UX and GM were chosen. The comparison between all the used operators is done in the next section. Here, the focus lies on demonstrating the capabilities of our heritage tracing approach. First, we explore the impact metrics found in the final generation. This is followed by investigating the population dynamics over multiple generations on one exemplary run.

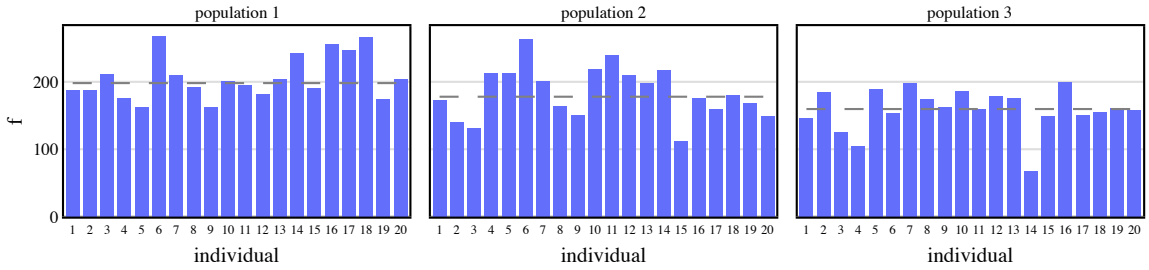


Figure 7.1: Fitness of the three initial populations. The median fitness is marked with a dashed line.

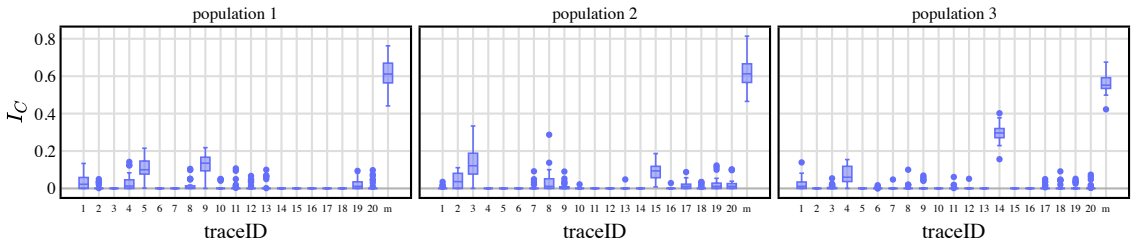


Figure 7.2: I_C for the three different initial populations using UX and GM.

Final Generation of One Test Configuration

Beginning the evaluation on the population dynamics, we explore the results for the different impact metrics found in the final generation of the algorithm. As a reference for the quality of the initial population, Figure 7.1 shows the fitness values of the solutions in a bar chart. The grey line shows the median fitness of the population. The impact metrics can be found in Figure 7.2 for the I_C, Figure 7.3 for the I_F, Figure 7.4 for the I_E, and Figure 7.5 for the $I_{F\cdot E}$. In the following, the results for each population are discussed.

Results for Population 1

⇒ Four impact measures produce similar results.
⇒ Good fitness individuals from the initial population are influential, but a direct correlation does not seem to exist.

Impact data for the worst initial population shows the mutation operator to feature the largest influence. However, this cannot be attributed to the quality of the initial population, as this is true for all three populations used. Data for the I_C in Figure 7.3 shows that individuals 15 and 9 influence the final result the most from all initial individuals. They also have the best fitness in the initial population (Figure 7.1). Furthermore, the traceIDs 1, 4, 8 and 19 also frequently survive the evolutionary process. All of them also correspond to individuals with a fitness better than the

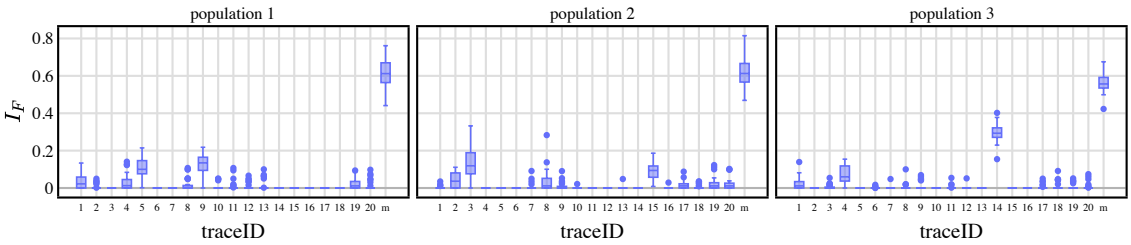


Figure 7.3: I_F for the three different initial populations using UX and GM.

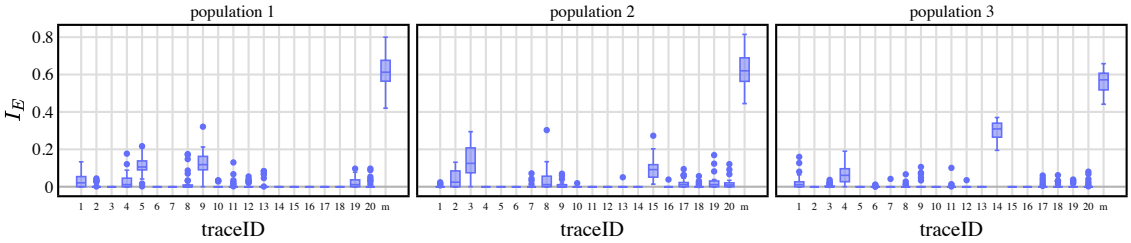


Figure 7.4: I_E for the three different initial populations using UX and GM.

median of the initial population. This might suggest that good fitness individuals are influential in the evolutionary process. However, we also find cases where the influence does not correlate with the initial fitness values. For example, the traceIDs 1 and 2 feature a very similar initial fitness. While traceID 1 shows considerable impact in the I_C metric, traceID 2 does not, only surviving the evolutionary process in some edge cases. This raises the question of why these two individuals, who feature a comparable fitness, have a different impact on the search process. Comparing the four impact metrics for population 1 in Figure 7.2, Figure 7.3, Figure 7.4, and Figure 7.5 shows them to be quite similar. Little differences can be found, especially comparing entropy-scaled and non-entropy-scaled metrics.

Results for Population 2

⇒ Similar results as for population 1.
⇒ Four impact measures again produce nearly identical results.
⇒ TraceIDs with a good initial fitness are influential, but a direct correlation does not seem to exist.

For population 2, we find similar effects to those for population 1. Again, the results for the four impact metrics are roughly equivalent. We also find the mutation operator to have the overall largest influence on the final result. Interestingly, in the I_C metric (Figure 7.2), the influence from mutation is nearly identical between population 1 and 2, even though population 2 has a better median fitness, and features fitter individuals in its initial population (Figure 7.1). This is an interesting observation, as one would assume a worse initial population might need more help from the mutation operator to reach the final result. But we do not find this to be the case. Taking a look at the individuals with the highest influence, we again see the initially best fitness individuals of population 2 (15, 3, 2, 20, 9) to feature the highest impact. And we again find it difficult to truly correlate the initial fitness with a large influence. Here, for example, individual 15 has the best fitness in the initial population, but only features the second-largest influence. Compared to population 1, this time, a few more individuals from the initial population contributed

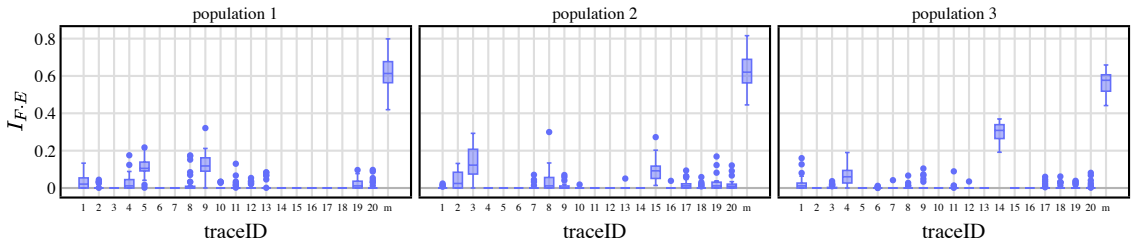


Figure 7.5: $I_{F\cdot E}$ for the three different initial populations using UX and GM.

genetic material.

Results for Population 3

⇒ Results for population 3 are again very similar, suggesting the fitness quality of the initial population is not a major influence on the population dynamics.

Population 3 again shows overall the same effects as found for the other two populations. Again, the best fitness individuals are also the ones who survive the evolutionary process and contribute more genetic material. But again, a direct correlation between fitness and a larger impact is difficult. Also, the results for all four impact metrics are again comparable. This suggests the quality of the initial population is not a large influence on how EA assemble their final solutions. This time, however, we do see a bit less influence from the mutation operator than for the other two populations. Influence from mutation is still very high, though. Nevertheless, the current test data does not support there to be a large difference in population dynamics. In the next sections, we will find that operators have a far larger influence on the gene-level population dynamics than the different initial populations used.

Single Test Run in Detail

Finally, we also want to look at one single test run in more detail. Here, we chose test run 15 of the 31 test repeats of the previous algorithmic configuration using GM and UX as an exemplary case. However, it has to be noted that convergence for other test runs can look different, as due to the stochastic nature of the algorithms, different individuals from the initial population survive the first generations. Here, we focus on presenting the capabilities of the T-EA when evaluating on one singular test run. Visualizing data this way will be infeasible for most scientific work. This would require an evaluation on a broader scale, visualizing and evaluating each of the 31 test repeats separately, to gain robust results. Again, the results are discussed for each of the populations used.

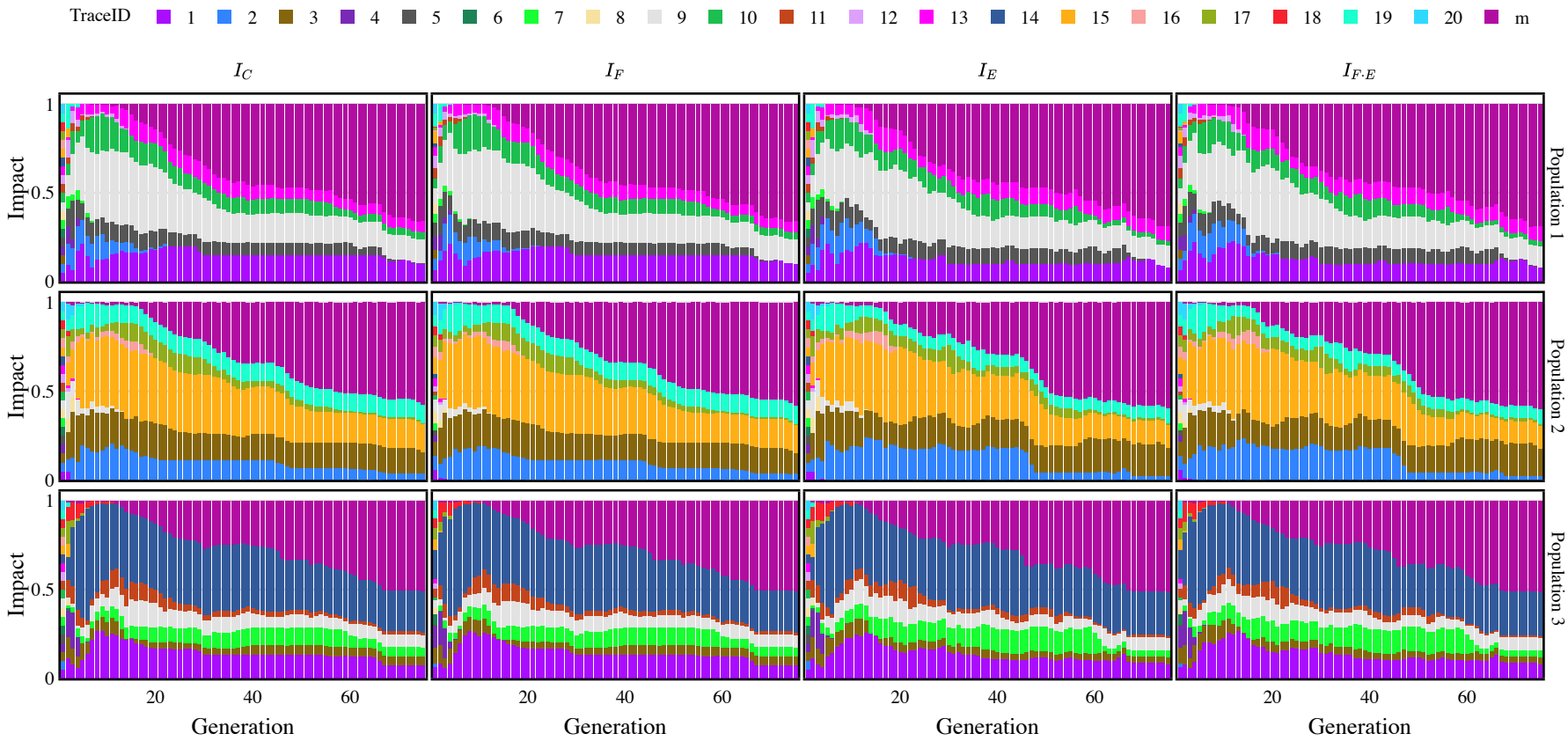


Figure 7.6: Impact metrics over the generation for test run 15 of the three populations using UX and GM.

Figure 7.6 shows the results of the four impact metrics over the 75 generations of the selected test run 15 as a stacked bar chart. Each traceID is assigned a color. The impact of population 1 is shown in the top row. At first glance, all four impact metrics feature the same trend in their population dynamics. In the initial population, all individuals are present. In the first few generations, only six initial individuals are still contributing genetic material (traceIDs 1, 2, 5, 9, 10, and 13). After this, the number of contributors becomes more stable, with mainly the influence of the mutation operator increasing. After this, only traceID 2 becomes extinct around generation 20, and traceID 5 after generation 70. As mentioned, the results between the impact metrics are relatively similar, with all four showing the same trend. However, we can also find differences. TraceID 1, for example, has a relatively stable amount of impact in the I_C and I_F in the generations 30 to 60. Considering the entropy in the I_E and $I_{F.E}$ metrics, however, we see that the influence decreases, and is more volatile between each generation. In general, we can find such differences more often in the earlier generations. When the population converges in later generations, differences in fitness get smaller, and the genomes of the individuals become more alike, meaning the scaling with these factors gets less relevant. We also like to point out that we find some interesting points during the evolutionary process, which might be of interest. Around generation 35, the influence of some traceIDs stays the same. Here, it might be interesting to further investigate which genetic material was retained. Especially for real-world problems, where domain experts might gain interesting insights through this information, this would be useful. Furthermore, it might be interesting to investigate what changed around generation 85, where the impact metrics show a shift, and traceID 5 goes extinct. A detailed evaluation of this is beyond the scope of this work, and the information gain through extracting such specific features for a benchmark problem might not yield the highest information gain. However, these more detailed investigations are interesting for real-world problems with expensive fitness evaluations, giving additional information to the domain experts *how* the EA has assembled their solution.

Results for Population 1

⇒ Only 30 % of traceIDs survive the first generations.
⇒ Impact metrics show some differences earlier in the evolutionary process.
⇒ We can identify some interesting points in the evolutionary process.

Results for the the second population can be found in the middle row of Figure 7.6. We also find that the majority of individuals do not survive the initial generations, with five traceIDs (excluding mutation) found in the final generation. In contrast to the previous population, a few more individuals contribute. In this example, more differences between the entropy-scaled impact metrics (I_E $I_{F.E}$) and the others can be observed, especially for traceID 2 from the initialization to around generation 50. This might again be an interesting point in the evolutionary process, especially for real-world applications. We again observe the impact of mutation increasing gradually. This is can be observed for populations 1, 2, and even 3.

Results for Population 2

⇒ Again, the genetic material of many individuals is discarded in the early generations.
⇒ Entropy-scaled impact metrics show differences.
⇒ Again, interesting points in the evolutionary process can be identified.

The population with the best initial fitness is presented in the bottom row of Figure 7.6. We again find the genetic material of only six individuals (traceIDs 1, 3, 7, 9, 11 and 14) to survive the first few generations, with again around 70 % of the initial population being discarded by the algorithm. After these first 10 generations, we also find some form of equilibrium. Interestingly, in this exemplary run, traceID 3 only retains a tiny amount of genetic material in generations 14 and 15, then gains

Results for Population 3

⇒ Similar trends as for population 1 and 2 are found.

more impact in the later evolutionary process. Also, we again observe some differences between the impact metrics, especially in the earlier generations. Most differences are again found between the entropy-scaled and the non-entropy-scaled metrics.

Summary of Single Test Run

While all three populations produce different gene-level population dynamics, we can still identify similarities. First, only around 30 % of the individuals in the initial population survive to the end. This is also found in related works for GAs, where studies showed the heritage could be traced back to only a small number of individuals from the initial population [14, 15, 61, 104, 105, 134, 135]. Here, 30 % of contributing individuals is more towards the higher end. Some works find this to be lower for GP [13, 103]. As we will see in the next sections and chapters, reasons for this difference might include the population size, the crossover or mutation operator, or the problem itself. In these visualizations, we could also identify some interesting points in the evolutionary process, which might be of interest. We see the most potential for real-world problems, where the decision variables hold more meaning and are more interpretable than the benchmarks used in this theoretical work. Finally, it needs to be reiterated that these results are only intended as a proof of concept for the T-EA. Evaluating EAs will be challenging in the context of scientific works, as algorithms need to be tested on a wide scale for robust results.

7.3.2 Operator Comparison

The following evaluation of the population dynamics of the four algorithmic configurations is split into three parts. First, a short overview of the performance in terms of fitness is given. Then, the impact data of all individuals is visualized. Finally, we compare the results of all four impact metrics.

Table 7.1: Performance for all test configurations, measured as the median for the best solutions found by each algorithm configuration.

	UX + UM	UX + GM	LX + UM	LX + GM
population 1 (worst)	6.505498	3.626226	18.521968	14.725612
population 2 (median)	8.664245	3.606875	16.551082	15.880422
population 3 (best)	5.207914	2.027834	17.316310	10.891847

Performance

Overview of Performance

⇒ UX performs better than LX.
⇒ GM performs better than UM.
⇒ Better fitness initial populations perform better, except in two cases.

Although performance is not the focus of our evaluations, we want to give a small overview to set the rest of the results into context. Each row shows the median best fitness reached for a given operator combination over the 31 test repeats. The columns show the results for each initial population. In general, the algorithms using UX outperformed the ones using LX by a considerable margin. For mutation, we find GM to outperform UM in all cases. This means the best performance was reached by the algorithm using UX and GM. We observe population 3 to always show the best performance. This makes sense, as it is also the population with the best initial fitness. However, for two configurations (UX+UM and LX+GM), we find population 3 to perform better than population 2. This is interesting, as population 3 is worse both in terms of the best fitness in its initial population, and its median fitness (Figure 7.1).

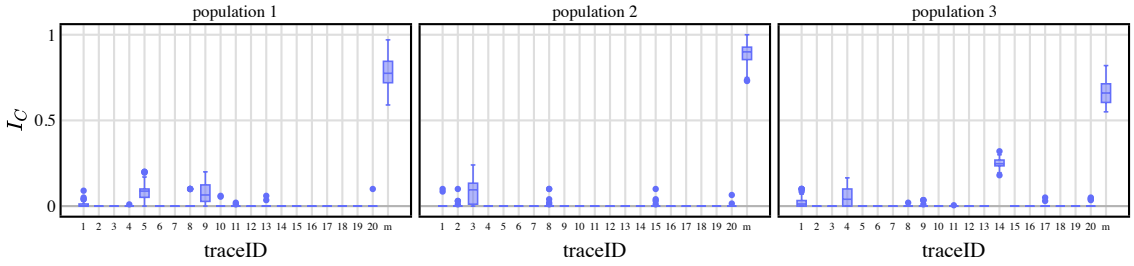


Figure 7.7: I_C for the tree populations using UX and UM.

<

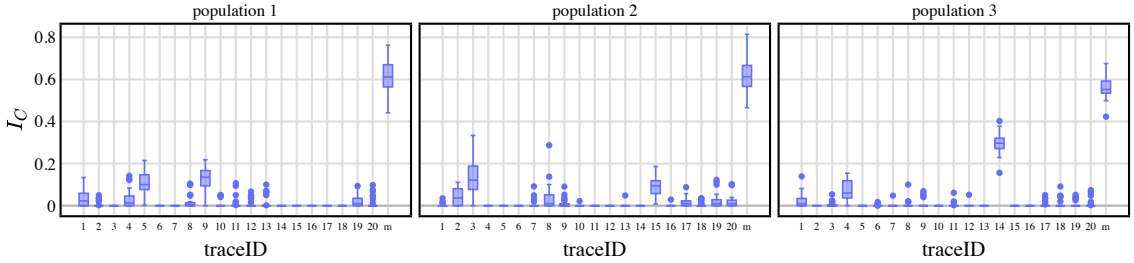


Figure 7.8: I_C for the tree populations using UX and GM.

Final Generation Impact

To evaluate the differences between the four operator combinations tested, the I_C of all test configurations is visualized. Figure 7.7 shows the results for UX+UM, Figure 7.8 shows the results for UX+GM, Figure 7.9 shows the results for LX+UM, and Figure 7.10 shows the results for LX+GM. In this section, we focus on the results from the I_C metric. The next section will go into more detail about the differences using the other three impact metrics.

Results for UX+UM

⇒ Few initial individuals consistently survive.
⇒ Large impact of mutation.

For the gene-copying UX and value-independent UM, shown in Figure 7.7, we can see two things across all three initial populations used. First, the impact of mutation is very high, with over 60 % of the genetic material in the results of population 3 stems just from mutation, and even over 85 % for population 2. Second, only a small number of individuals consistently contribute genetic material. In this case three individuals for populations 3 and 1, and only one individual (traceID 3) for population 2. The rest of the population only shows influence in occasional edge cases. Recalling the fitness of the initial populations in Figure 7.1, we find the individuals who consistently contribute genetic material to be of good fitness. However, for population 2, it is not the best fitness individual 15, but the second-best individual 3, which has the highest impact. Interestingly, individual 15 does not show a consistent influence at all.

Results for UX+GM

⇒ More individuals contribute genetic material.
⇒ Still large impact of mutation.

The impact of value-dependent mutation GM and the gene-copying UX are shown in Figure 7.8. Compared to the previous result using a value-independent mutation operator, two things become apparent. First, the impact of the mutation operator is a bit lower. Second, more

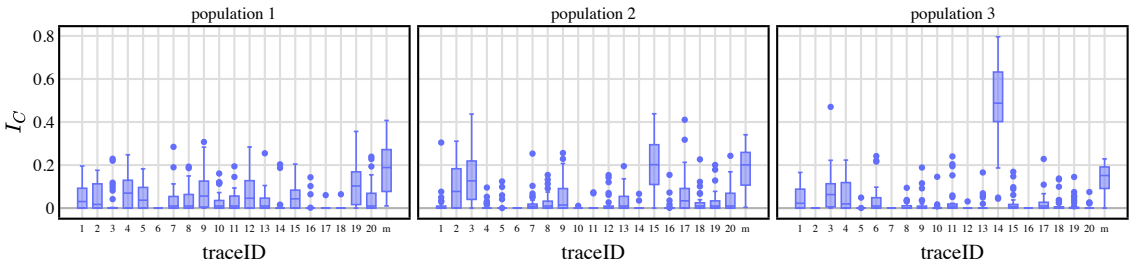


Figure 7.9: I_C for the tree populations using LX and UM.

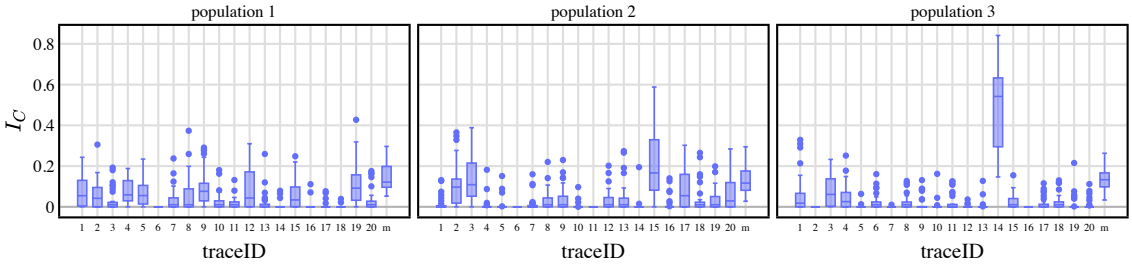


Figure 7.10: I_C for the tree populations using LX and GM.

individuals from the initial population contribute genetic material. A decrease in the impact from the mutation operator is to be expected, as the operator now only partially influences genes. However, we still find a considerable impact through mutation. This time, the mutation impact between the three populations is more similar, roughly between 55 and 60 %. Interestingly, more individuals from the initial population now contribute genetic material, which might indicate the algorithm was able to better use the genetic material of the initial population than previously. However, we find a counterargument for this reasoning in the data of the next operator combination. The previous section already discussed the results for this operator combination in more detail. Here, we again want to highlight that the individuals contributing genetic material to the final result are of better fitness, but we also find cases where better-fitness initial individuals have a lower impact than others.

Results for LX+UM

⇒ Far more individuals contribute genetic material for gene-combining crossover than for gene-copying crossover.
⇒ Impact of mutation is far lower, too.

Changing the crossover operator from gene-copying to gene-combining drastically changes the gene-level population dynamics, as can be seen in the data in Figure 7.9. Previously, only up to six individuals from the initial population contributed their genetic material. This ratio is now flipped, with around six individuals not contributing genetic material in the median of most test runs. It makes sense that we find more traceIDs in the final population, as one gene now allows multiple parents to survive. A second difference is the much lower impact from the mutation operator. While previously, mutation provided around 60 % of the genetic material in the final generation, now it is only around 15 to 20 %. For population 1, we do not find individuals who have a particularly high influence. But for the populations 2 and 3, we do. For population 2, the traceIDs 2, 3, and 15 have a larger influence. For population 1, traceID 14 is the one with the single largest influence of around 50 %. Now, one might think that this

increase in the use of the genetic material of the initial population might mean that the algorithm is more successful in using it, building better offspring. However, the previously discussed performance data shows this not to be the case, as the gene-combining LX is outperformed by the gene-copying UX by quite some margin (see Table 8.1). How the genetic material is used probably cannot be used as a measure of the success of an algorithm.

Results for LX+GM

Finally, we want to discuss the results for the gene-combining LX and the value-dependent GM. Essentially, the results here are similar to previous observations: The impact of value-dependent mutation is lower, and the amount of traceIDs surviving the evolutionary process through gene-combining crossover is higher. While this pattern is noticeable, it has to be acknowledged that here we only generated test results for one benchmark problem. Results for tests with a different fitness landscape might vary.

⇒ Comparable patterns to the other operator combinations can be observed.

Summary of Operator Comparison

In this evaluation, we were able to observe interesting changes in the gene-level population dynamics for different mutation and crossover operators, as well as different quality initial populations. First, the quality of the initial population did not significantly change the population dynamics, as all three populations showed analogous effects. Second, comparing the gene-copying UX with the gene-combining LX, we observed a large difference. For the UX operator, only a small number of individuals from the initial population contributed genetic material. This is similar to previous evaluations of genealogical evaluations of GAs [14, 15, 61, 104, 105, 134, 135]. However, for the gene-combining LX, far more individuals contributed genetic material. This makes sense, as for this operator, multiple parents can survive for each gene. And third, we also saw a difference in the mutation operators used, with impact for the value-dependent GM being a bit lower than the value-independent UM. We find that the better fitness individuals in the initial population survive more to the final result. But a direct correlation between initial fitness and impact cannot be drawn. In the next chapter, we will investigate the reason for this. Finally, we need to acknowledge that these results served the purpose of an initial exploration of the T-EA and gene-level population dynamics. More tests, especially with a larger set of benchmarks, will be necessary for more robust results. We do want to mention that many of the results found for the UX and UM operators were also found for the MaxOnes and Knapsack Problems, found in previous publications of the author [112, 113]. However, numerous works also show the fitness landscape of the test problems to have an influence on the population dynamics of EA, so results always have to be viewed in this context.

Differences between Impact Metrics

Finally, we want to discuss the differences between the impact metric in the final generation. The median differences are shown in Table 7.2 for population 1, in Table 7.3 for population 2, and Table 7.4 for population 3. The values are rounded to three decimal places.

Differences between Impact Metrics

⇒ Little differences are found in the final generation.

Statistically significant differences between the I_C and the other three metrics were calculated with $p = 0.05$. However, as can be seen in the three tables, no significant differences exist in the final generation. This is true for all algorithmic configurations and initial populations used. On the one hand, this can be seen as expected. When SOEAs converge to their optimum, the individuals in the final population become more alike, as well as their performance. This means that differences in fitness and entropy also get smaller. Considering the performance results in Table 7.1, we see that some operator combinations, like LX+UM, find far worse solutions. However, they also show no differences in the impact metrics due to a converged population. This suggests the gene-level population dynamics are comparable and more dependent on the generation than on the convergence of the algorithm. We believe the main application for the four impact metrics is the earlier generations. Here, differences between the metrics are more present, as was shown in Section 7.3.1.

Table 7.2: The median values of all four impact metrics of population 1. Results are rounded to three decimal places. Statistically significant differences to the I_C metric are marked with a "*".

traceID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	m
CI	0.014	0.000	0.000	0.011	0.084	0.000	0.000	0.000	0.081	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.013	0.000	0.424
FI	0.013	0.000	0.000	0.011	0.084	0.000	0.000	0.000	0.081	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.013	0.000	0.424
EI	0.013	0.000	0.000	0.011	0.084	0.000	0.000	0.000	0.086	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.015	0.000	0.414
FEI	0.013	0.000	0.000	0.010	0.084	0.000	0.000	0.000	0.086	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.015	0.000	0.413

Table 7.3: The median values of all four impact metrics of population 2. Results are rounded to three decimal places. Statistically significant differences to the I_C metric are marked with a "*".

traceID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	m
CI	0.000	0.027	0.108	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.092	0.000	0.000	0.000	0.000	0.000	0.403
FI	0.000	0.027	0.108	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.092	0.000	0.000	0.000	0.000	0.000	0.405
EI	0.000	0.024	0.115	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.090	0.000	0.000	0.000	0.000	0.000	0.393
FEI	0.000	0.024	0.113	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.090	0.000	0.000	0.000	0.000	0.000	0.393

Table 7.4: The median values of all four impact metrics of population 3. Results are rounded to three decimal places. Statistically significant differences to the I_C metric are marked with a "*".

traceID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	m
CI	0.000	0.000	0.001	0.039	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.306	0.000	0.000	0.000	0.000	0.000	0.000	0.343
FI	0.000	0.000	0.001	0.039	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.306	0.000	0.000	0.000	0.000	0.000	0.000	0.343
EI	0.000	0.000	0.001	0.039	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.317	0.000	0.000	0.000	0.000	0.000	0.000	0.424
FEI	0.000	0.000	0.001	0.038	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.318	0.000	0.000	0.000	0.000	0.000	0.000	0.424

7.4 Summary

Chapter Overview

This chapter provides a first exploration into the heritage data tracked by the T-EA, as well as the four proposed impact metrics. For this, experiments on the Rosenbrock [37] benchmark function were conducted using different recombination operators. Two crossover operators were used: UX [24] as a gene-copying operator, and LX [143] as a gene-combining operator. For mutation, UM represented value-independent, and GM [116] a value-dependent mutation operator. First, a single test run and a single test configuration were evaluated in more detail. Then, the differences between the operators were discussed. Data was evaluated using the four proposed impact metrics. Differences of which, and their potential use, were also discussed.

Main Findings

Through the heritage data from the T-EA, differences in the gene-level population dynamics between the operators could be shown. Evaluating differences between the four operators in this thesis revealed these trends:

- The quality of the initial population, in terms of fitness, does not seem to have a large effect on the population dynamics of an EA.
- For gene-copying crossover operators, only a small number of individuals are contributing genetic material to the final result. This is similar to previous findings for GAs [14, 15, 61, 104, 105, 134, 135]. For gene-combining crossovers, this is different, with only a few individuals not contributing.
- Value-dependent mutation operators show less impact than value-independent operators, as they get credited less for each occurring mutation.
- We find only small differences between the four impact metrics in the later generations, as the algorithm is already more converged. In the earlier generations, more substantial differences can be found. Here might be an interesting application for these metrics, especially considering the use of heritage data in the algorithmic design.

Furthermore, evaluating a single test configuration of the UX and GM operator, it could be shown that the algorithm discards the most genetic material in its first few generations, then gets into a more stable state, in which mainly the impact of the mutation operator rises. Here, we also find interesting points in the evolutionary process, namely, whenever we find larger changes in the later generations. However, while such investigations might be promising, they also feature several challenges subject for future work.

Outlook and Future Work

Evaluations in this chapter allowed for an interesting first exploration on gene-level population dynamics in EA. However, it can only be seen as a starting point. For this first proof of concept, we limited our test setup to one test problem and a relatively small population size. For more robust results, tests with more benchmarks and algorithmic configurations are required. Nevertheless, we found a lot of interesting starting points worth further exploration in future works. We also believe it would be interesting to further explore the differences between the four impact metrics. We especially find differences in the earlier generations when the population is not yet converged. One application of this might be the application of gene heritage data in the algorithmic design. Finally, we often find good fitness individuals to also be influential in the search process. However, a direct correlation does not seem to exist, as we also saw cases where the best fitness individual does not feature the highest influence. This leads to the question of which individuals are influential in the search process, and why. This will be the aim of the next chapter.

Influential Individuals in SOEAs

8

This chapter is largely based on the author's publication [118], with additional test data.

8.1 Motivation and Overview 81
8.2 Test Setup 82
8.3 Evaluation 83
8.3.1 Influence of Optimal Genetic Material 83
8.3.2 Differences in Distance to x^{opt} 89
8.4 Summary 92

The goal of this chapter is to find which individuals from the initial population are influential in the search process of SOEAs. For this, we create individuals of specific fitness targets and genome properties, as specified in Section 5.1, and track their influence using the T-EA. More specifically, we generate individuals with a fixed fitness, but a varying amount of good genetic material, and for varying distances to $\vec{x}^{opt}$ in the search space. This way, we can identify which factors are important for the evolutionary process.

8.1 Motivation and Overview

Motivation

Individuals in EAs are represented in two spaces: The objective space, represented by the fitness value, and the search space, represented by the genome. As criticized throughout this thesis, the quality of these individuals is usually only measured in the form of their fitness. This makes sense for assessing the final performance, as the fitness function is designed to measure the quality of a solution for a given problem. However, individuals in EAs do not only represent the solutions found by the algorithm at any given time, but also build the base for the ongoing search with the goal of finding the global optimum. This is especially true for the initial population as the starting point for each evolutionary process, containing the genetic material that will be recombined through crossover and mutation until the termination criterion is met. It is generally believed that it is the better-fitted individuals of a population that lead the algorithm to the final result. This is reflected in the bias towards fitter individuals in the mating selection or in the survival selection. However, previous evaluations in this thesis did not find this trend to be generally true. The evaluation in the previous chapter shows that the best-fitness individuals are not always the most influential. This was also shown in a previous publication of the author [112] on the MaxOnes and Knapsack problems. This raises the question of which individuals are influential in the evolutionary process, leading the algorithm towards its final solution.

What is a "Good" Genome?

One can easily imagine two individuals, both with the same fitness, but different genomes. For example, one x_1 featuring optimal genetic material but a few "bad" genes far off their optimal value. Then, a second x_2 where each gene is roughly the same distance to their respective optimum, neither close nor far from the optimal value. Would one individual be better at leading the algorithm towards the global optimum than the other? This question is not trivial, especially when considering different crossover operators. For operators which are designed to (partially) copy the genome values directly from the parents to the offspring, such as UX [24], it is intuitively better to have the already optimal genetic material. However, for crossover operators that combine two genome values, like

the popular SBX [25], this is different. As the operator combines the genome of both parents, it might be more desirable for all genes to be close to their respective optimal values. Furthermore, if an individual already has good fitness, a good genome is assumed. However, it is unclear if the algorithm can take advantage of good genome values seeded into a bad fitness individual.

Population Initialization

Initialization techniques for EAs can be seen as a similar research direction, and were discussed in more detail in Section 6.1. The vast majority of studies in this area only evaluate population initialization techniques in terms of performance gains compared to a uniformly random population. This leaves a knowledge gap for why some genomes work better than others. One of the few studies investigating what makes a good initial population was done by Daida [141]. They investigate which individuals, and what genetic material, from the initial population were influential and beneficial for the performance of GP algorithms. In their conclusion, they mention that GP needs a sufficient amount of "good genetic material"[1] in the initial population of the algorithm. This leads to the question of whether such important genes can also be found for EAs.

Chapter Overview

In this chapter, we address this research gap to better understand which individuals are influential in the search process of an EA. This is done by seeding the initial population with individuals of desired properties. These seeds are created for different fitness targets, featuring different genome distributions and different distances to the global optimum in the search space. The experiments are performed using both the genome copying UX [24] and the genome combining SBX [25] operators in multiple dimensions ($\mathfrak{d}$=[4, 8, 16, 32]). Tests are performed with three benchmark functions[2]. Evaluation of the results is done both in terms of performance gains of the algorithm and population dynamics, tracking the influence of the seed individual using the T-EA. With this, performance enhancements can directly be linked to the seeded individuals.

8.2 Test Setup

Benchmark Functions

Three benchmark functions are used: The Sphere function (Equation 2.7), the Rastrigin function [36] (Equation 2.8) and the Rosenbrock function [37] (Equation 2.9). The Sphere function is one of the easiest SOPs, but also one of the few for which exact seed individuals can be generated. The Rastrigin benchmark [36] is highly multi-modal, featuring many local minima, but is also separable. The Rosenbrock benchmark [37] is less multi-modal, but also not separable. A more detailed overview of the test function can be found in Section 2.4.1. Each test is repeated with four different genome sizes ($\mathfrak{d} = [4, 8, 16, 32]$).

Seed Individuals

Seed individuals are generated as described in Chapter 5. As the fitness of the seeds influences their chance of being selected for recombination, four different fitness targets, based on the rest of the initial population, are used. The best (b) seed is generated with half the fitness of the otherwise best individual of a population. The other seeds are also based on the

[1] as in nodes of a GP genome which are also part of x^{opt}.
[2] In the original publication, only the Sphere function was used.

initial population used, with the fitness targets of the upper quartile (uq), median (m) and lower quartile (lq) of the population. Four genome configurations, with 0%, 25%, 50%, and 75% of optimal genetic material, and two search space distance targets, "far" and "close", are generated. This results in 32 different seed individuals for each problem (Rastrigin and Rosenbrock) in each dimension ($\mathfrak{z} = [4, 8, 16, 32]$). For the Sphere function, no "far" and "close" seeds can be generated, as individuals with the same fitness always have the same Euclidean distance in the search space. For this reason, we have only 16 seed individuals for the sphere function.

Algorithmic Setup

Experiments are run using pymoo version 0.6.1.1 [144]. The standard implementation of a GA is used. This means Tournament selection with a tournament size of 2 is used to select offspring for recombination. Fitness selection is used for survival, which in practical terms is a greedy approach, selecting the best-fitness individuals for the next generation. Two crossover operators are used. The gene-copying crossover UX [24], and the gene-combining crossover SBX [25] with $\eta = 20$. Finally, PM [26] ($\eta = 20$) is used with a mutation probability of 1 %. The population size is set to $n = 40$. For each dimension, an initial population of 39 individuals is generated, which is augmented with the seed. The initial populations are kept the same for each dimension to maintain comparability with the heritage data. Each test was done over 100 generations and repeated 31 times.

Evaluation Metrics

The impact of the seed individual is measured using the I_C metric (Equation 4.16). Performance is compared in terms of the fitness of the best individual in the population.

8.3 Evaluation

Evaluating the results is split into two parts: First, the influence of optimal genetic material is discussed. Second, the differences for the seeds with different distances to $\vec{x}^{opt}$ are discussed. The chapter closes with a summary and a critical discussion of the limitations and future improvements necessary for this research direction.

8.3.1 Influence of Optimal Genetic Material

Evaluating the influence of optimal genetic material is done for each function separately. We begin with the Sphere function, followed by the Rastrigin and finally the Rosenbrock function.

Sphere Function

The evaluation of the Sphere function data is split into two parts. First, the differences between different search space dimensions are compared. This is followed by comparing the differences in seed quality. A comparison of seed closeness is done for the following test functions, as the Sphere function is the only one for which "close" and "far" seeds could not be generated.

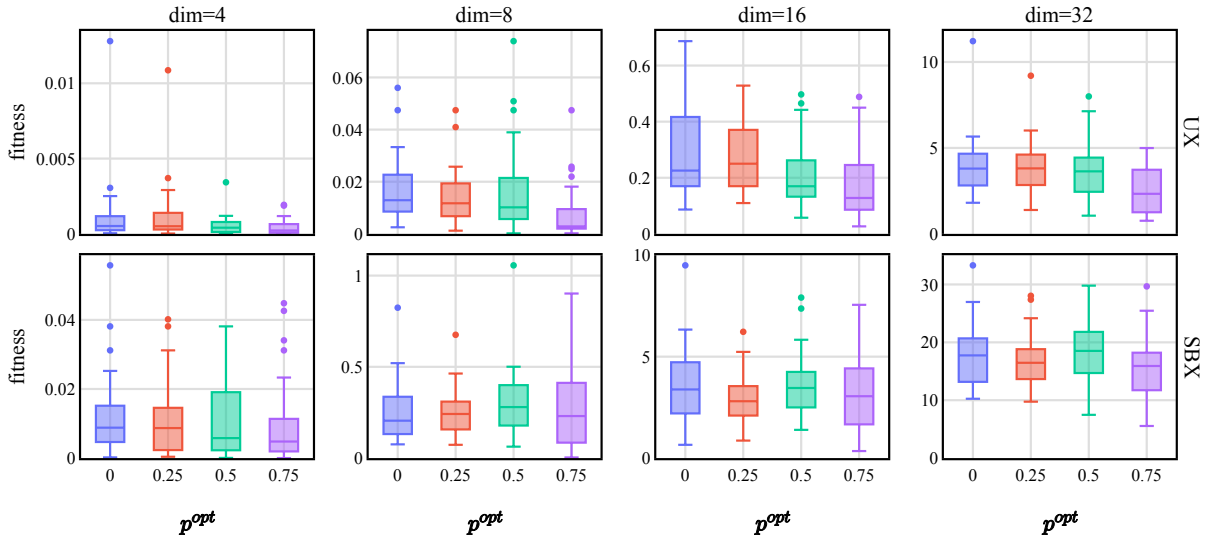


Figure 8.1: Performance on the Sphere function of the upper quartile seed over multiple search space dimensions.

Comparing Search Space Dimensions First, the differences found for different search space dimensions will be the focus. The upper quartile seeds were chosen as an example, as all seed qualities showed similar results over the different search space dimensions. Here, the performance of the upper quartile seed is visualized in Figure 8.1, and its respective seed influence is presented in Figure 8.2. Plots of the other tests can be found in Appendix B.

Performance Differences

⇒ Different search space dimensions ♁ do produce very similar results. More optimal genetic material in the seed results in better performance.

To evaluate the performance benefits of seeding different individuals, Figure 8.1 shows the minimum (best) fitness values in the final generation of the lower quartile seed as a box plot. The top row shows the results of the UX (genome copying) and the bottom row the results for the SBX (genome combining) crossover operators. For the higher dimensions, it can clearly be seen that the performance of the algorithm improves with more optimal genes. This is true both for the combining and copying crossover operators, and can also be seen for the lower dimensions. However, in lower dimensions, the performance is more equal as the algorithm has already converged more to the optimum value. The positive influence of the optimal genetic material is noticeable, as the fitness of the seed individuals and the distance to $\vec{x}^{opt}$ is the same.

Seed Impact Differences

⇒ Again, different search space dimensions ♁ do produce very similar results. More optimal genetic material in the seed results in a larger seed influence, linking the performance benefits to the increased use of its genetic material.

Evaluating the impact data clearly attributes this effect to the seed individual. Figure 8.2 shows the impact of the seed, the rest of the initial population, and the mutation operator. Again, the data is shown as a box plot, with the rows representing the different dimensions and the columns the crossover operator used. Similar to the performance plot, we can see that seed individuals with a higher percentage of optimal genes contribute more to the final result than those with a lower percentage. As all seeds in this test have the same fitness, their selection for offspring creation is equal, so the higher influence can be linked to the optimal genes performing better in crossover. This is to be expected for the gene-copying crossover, as it keeps the already optimal genome value. However, this is also true for the gene-combining SBX. Here, one could assume that each gene being closer to the optimum would lead to a better performance, but this is not the case.

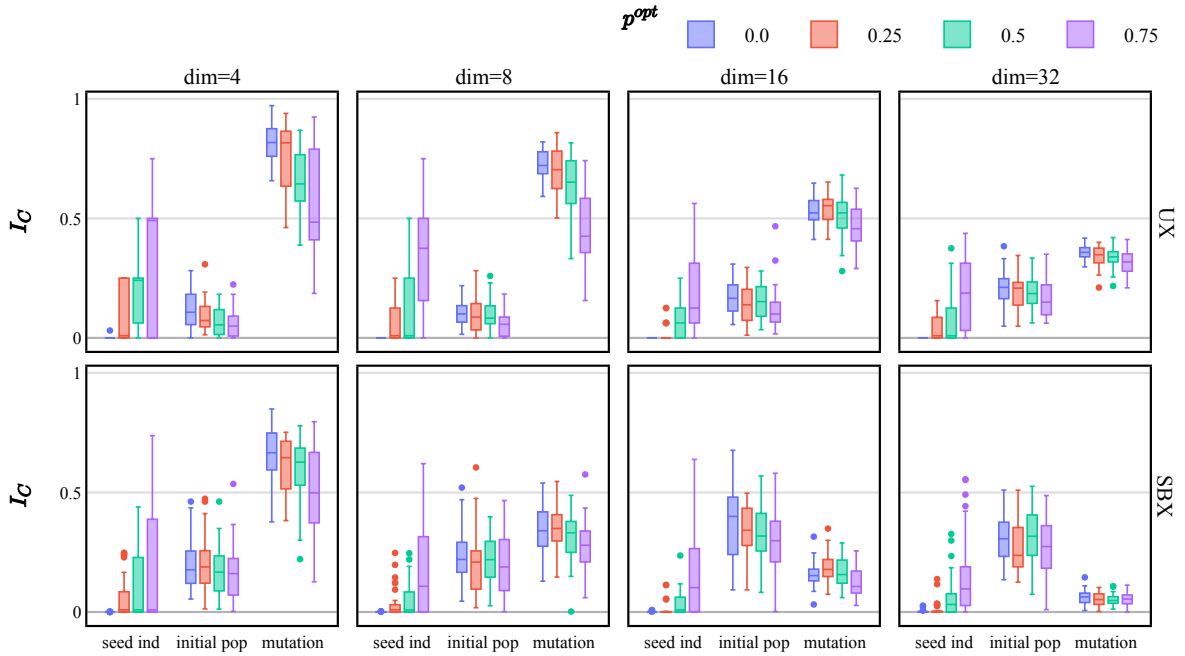


Figure 8.2: Seed influence for the Sphere function of the upper quartile seed over multiple search space dimensions.

Summary

In this first look at the data, seeds containing more optimal genetic material resulted in a better performance for the Sphere function, compared to seeds with the same fitness, but an overall more mediocre genome. The impact data from the T-EA could clearly link these performance benefits to the use of this optimal genetic material. Comparing the results for different search space dimensions, little differences can be found. This does not only hold for the upper quartile seed individual, but also for the other seeds. Results for the other two test problems are also the same, as can be seen in Appendix B. For this reason, the remaining evaluations in this chapter will be compared for $\mathfrak{z} = 8$, exemplary for all other search space dimensions.

Comparing Fitness Targets The fitness of the seed individual is an important factor for its chances of survival. The following evaluation compares the differences in performance using seeds of a different fitness level. The fitness of the seed individuals is derived from the initial population. First, the seed is constructed as the best individual of the population (half of the otherwise best individual in the population). Furthermore, the lower quartile, median, and upper quartile are used as fitness points. This way, we hope to capture a very good, good, medium and bad individual. As results over different search space dimensions are similar, $\mathfrak{z} = 8$ is chosen as an exemplary case. The performance data for this is shown in Figure 8.3, and the I_C of the seed individual, initial population, and mutation operator is shown in Figure 8.4. Data for the other test cases can be compared in Appendix B.

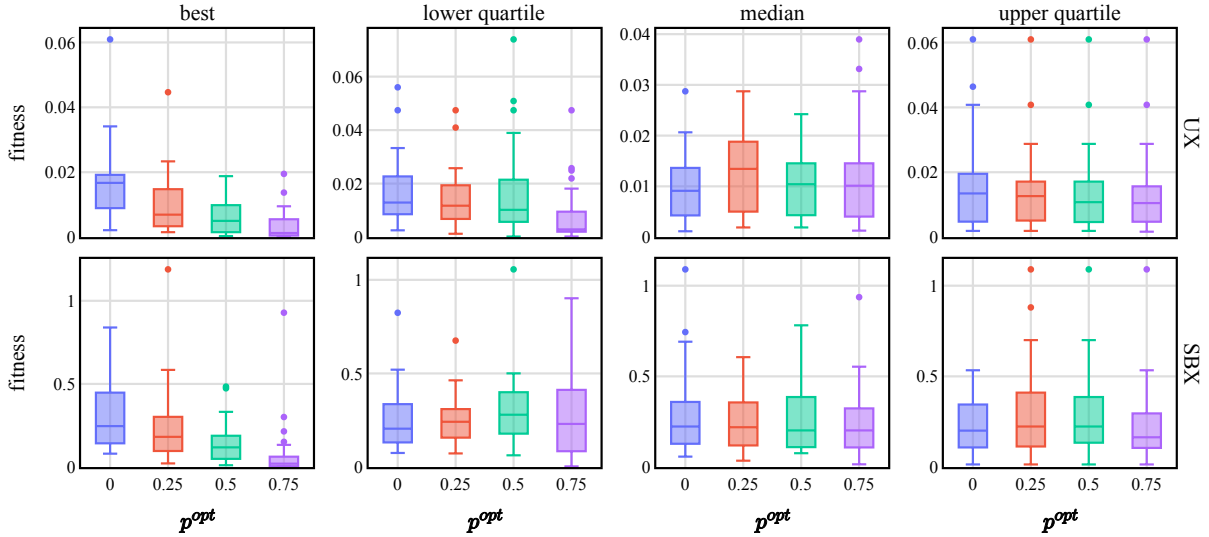


Figure 8.3: Performance on the Sphere function of the different seed qualities for $\flat = 8$.

Performance Differences

⇒ Performance benefits for individuals with a higher amount of optimal genetic material decrease with a worsening in seed fitness. Probably because the seeds do not survive the selection process.

The performance data in Figure 8.3 shows a similar trend as for the comparison of different search space dimensions: Seeds with a larger percentage of optimal material improve algorithmic performance. However, these gains shrink with a worse fitness from the seed individual. There are two possible reasons for this. On the one hand, as the algorithm uses tournament selection for offspring creation, the optimal genetic material is less likely to be chosen. On the other hand, it is also possible that the very bad genes do not produce good offspring, even if some parts of the genome are already optimal. Interestingly, for the median and upper quartile seeds with 25 % and 50 % of optimal genes, performance for SBX is worse than no optimal genetic material (Figure 8.3). Performance for the UX operator, on the other hand, still improves with the higher amount of optimal genetic material. This also indicates other influences on the population dynamics, as already observed in the case study of the single test configuration.

Impact Differences

⇒ Impact of the seeds is also decreasing with a worsening in seed fitness.

This is supported by the impact data in Figure 8.4. The influence of the seed individual is going down with a worse fitness. This is somewhat expected, as lower fitness individuals are not chosen as much for recombination. For the better fitness targets, an increase in the amount of optimal genetic material not only increases the influence of the seed, but also decreases the influence of the rest of the initial population, and the mutation operator. Also to note is that for the outlying performance results in the upper quartile fitness target, impact data in Figure 8.4 shows that the influence of individuals is still increasing with the amount of optimal genetic material. This means the genetic material of the seed is still exploited by the algorithm, but it is leading to a worse performance. One reason for this could be a lack of good genetic material in the decision variables, where no optimal genetic material was seeded.

Rastrigin Function

Similar to the Sphere function, results for the Rastrigin function are very similar across the different search space dimensions. For this reason, we

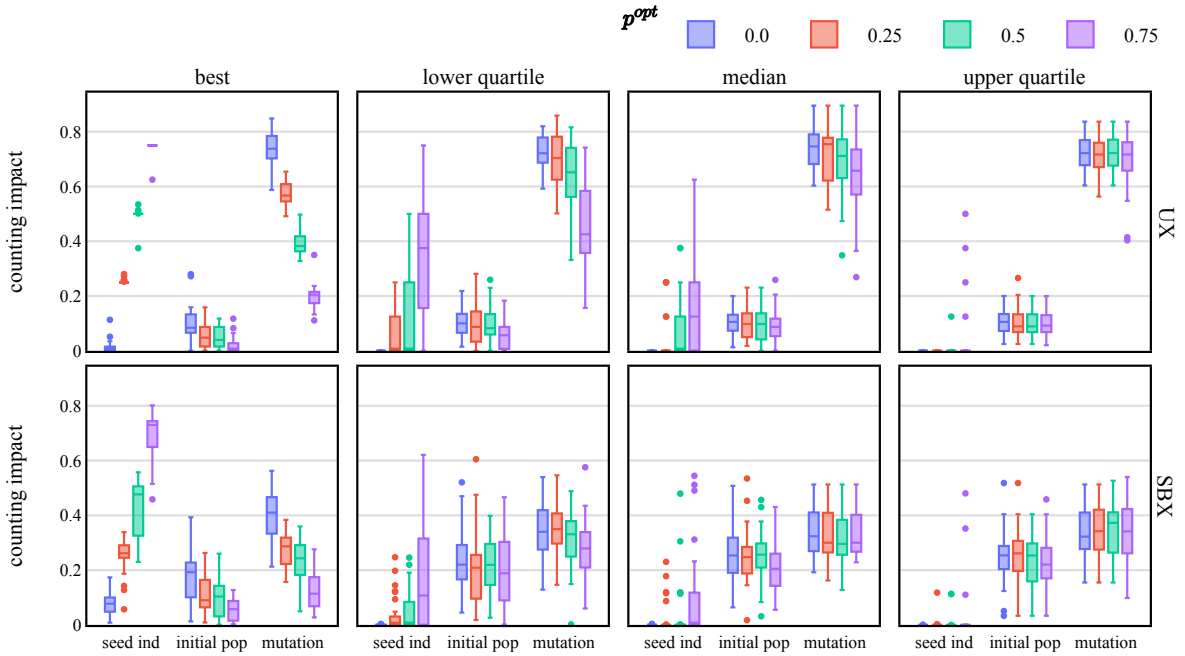


Figure 8.4: Seed influence for the Sphere function of the different seed qualities for $\mathfrak{z} = 8$.

select $\mathfrak{z} = 16$ as an exemplary case to discuss the found effects. Furthermore, as we will see in the next section, the results between the "far" and "close" seeds are quite similar. For this reason, the "far" seeds are selected as the exemplary case here. Visualizations for the additional data can be found in the Appendix B.

Performance Differences

⇒ Same as for Sphere function: Best and lower quartile fitness show performance differences. A high number of optimal genes improves performance. A low number of optimal genes even deteriorates it.

Results for the performance of the different seed individuals for the Rastrigin function for $\mathfrak{z} = 16$ are shown in Figure 8.5. Especially for the seeds with a good fitness (best and lower quartile), performance differences can be seen. Similar to the Sphere function, these performance differences depend on the amount of optimal genetic material in the genome. Individuals with a comparably good fitness, but with no to a low amount of optimal genes, see a deteriorating performance, while the seeds with a large amount of optimal genes improve it. Seeds with a worse fitness (median and upper quartile) show more equal performance, which is to be expected as they are less likely to be chosen for recombination. Interestingly, comparing the performance between the different seed qualities for the SBX and UX operators reveals no real performance benefits for seeding the population with better fitness individuals. Only the seeds with good fitness and a higher percentage of optimal genes perform noticeably better. This further indicates that fitness is not the most important factor when it comes to the ability to create good offspring and improve the overall performance.

Impact Differences

⇒ Similar findings to the Sphere function: More optimal genes means more seed impact, for fitter individuals.

The impact of the seed individual is visualized in Figure 8.6. Equivalent to the improved performance, the influence of individuals with more optimal genetic material is larger. Again, this effect is more pronounced for seed individuals with better fitness, as they are more likely to be chosen for recombination. However, in contrast to the performance improvements, the influence of seed individuals for a high p^{opt} is also

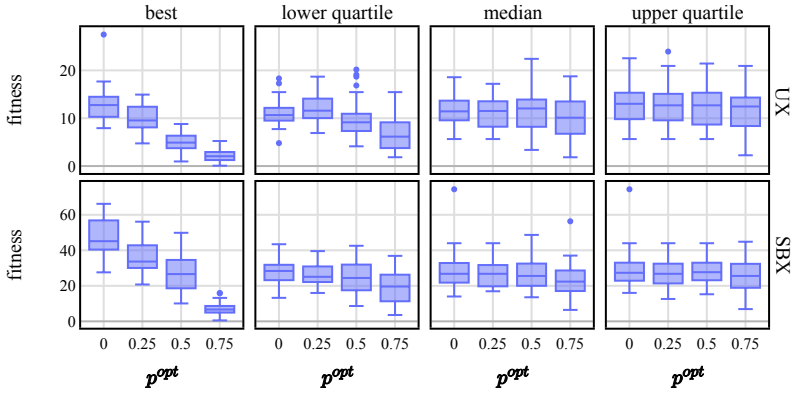


Figure 8.5: Performance on the Rastrigin function ($\mathfrak{z}$= 16).

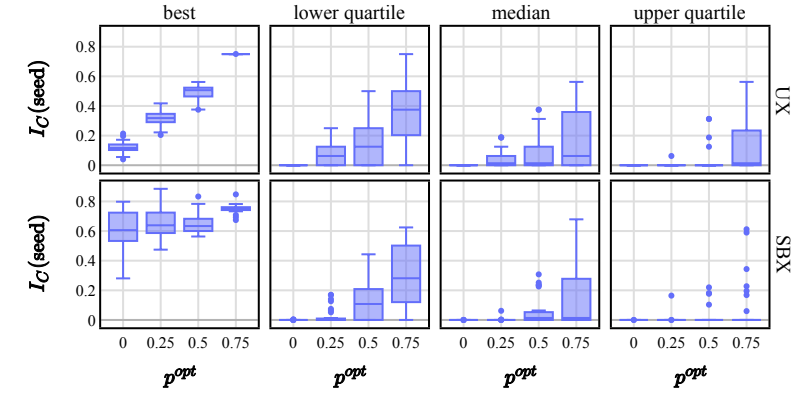


Figure 8.6: Seed influence for the Rastrigin function ($\mathfrak{z}$= 16).

present for seed individuals with of median or upper quartile fitness. These effects are found in both crossover operators, which overall show quite similar results. Only the seed of the best quality shows outlying results for the SBX operator, with a large influence of the seed individual across the board.

Rosenbrock Function

Again, the results for the Rosenbrock functions are similar over the different search space dimensions, and between the "far" and "close" seeds. For this reason, the dimensionality of $\mathfrak{z}$= 16 is selected as an exemplary case to discuss the found effects. Visualizations for the additional data can be found in the Appendix B.

Performance Differences

⇒ Same as for Sphere and Rastrigin functions: Best and lower quartile fitness show performance differences. A high number of optimal genes improves performance. A low number of optimal genes even deteriorates it.

Performance results for the Rosenbrock function with $\mathfrak{z}$= 16 in Figure 8.7 repeat the trends found in the Rastrigin function (Figure 8.5). Again, performance benefits from a larger p^{opt} can be found for good fitness seed individuals, although here the lower quartile fitness seed in the UX function already does not find this trend. This hints at other factors influencing the performance of the algorithm besides the availability of optimal genetic material. One aspect not explored in this work is the distribution of the optimal genetic material in the genome. As the seeds in this paper are created very structurally, all not optimal genes are placed at the start of the genome. We believe the availability of good genetic material at each genome to be the main factor for the performance differences seen here, as there are few other variables from the setup of the experiments.

Finally, the impact of the seed individuals in Figure 8.8 further shows the trends already described for the Rastrigin function in Figure 8.6. The resulting data for both show very similar trends as for the sphere functions, validating the findings across multiple problems. However, outlying solutions still leave gaps to be explored, as the number of optimal genetic material and the fitness of the individual cannot be the only factors when it comes to algorithm performance.

Impact Differences

⇒ Similar findings to the Sphere and Rastrigin functions: More optimal genes means more seed impact, for fitter individuals.

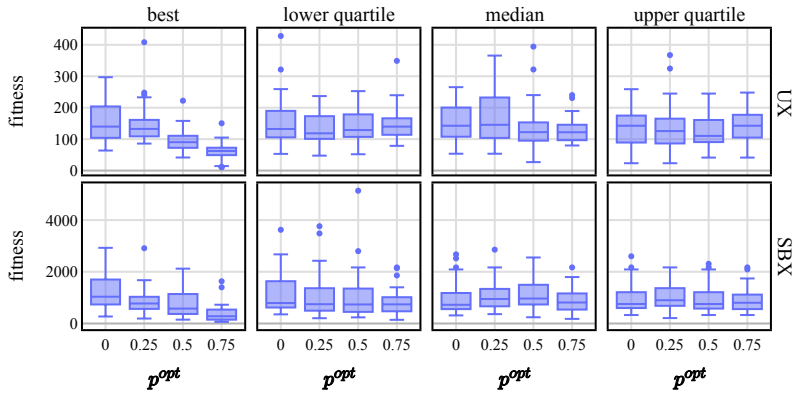


Figure 8.7: Performance on the Rosenbrock function (♭= 16).

8.3.2 Differences in Distance to x^{opt}

One aspect not focused on in the previous section is the distance of individuals in the search space. It might be assumed that if two individuals, both with the same fitness but with different distances to $\vec{x}^{opt}$, are compared, the one which is closer will perform better. To test this hypothesis, seed individuals with the same fitness, but different search space distances are created for the Rastrigin and Rosenbrock functions, as described in Section 5.1.

Differences in Distance from Seed to x^{opt}

Starting the evaluation, we show the differences in distance from the seed individuals to x^{opt}. Certain fitness targets or amounts of optimal genetic material have a different potential for generating genome values, which should be considered when evaluating the performance and heritage data. Figure 8.9 shows the differences in Euclidean distance for the seed individuals generated close and far from the optimal solution, as a heatmap. First, it can be seen that the distances increase with a higher dimensionality, which makes sense, as the maximum possible distance increases for higher dimensions. Furthermore, with a higher percentage of optimal genetic material, the distances between the individuals decrease. While for $p^{opt} = 0\,\%$ seeds with larger distances could be generated, higher p^{opt} show a significantly lower distance. Also, the best and upper quartile fitness targets show a smaller distance. Recalling the 1D fitness landscape of the Rastrigin function in Figure 5.1, this makes sense, as for the best or upper quartile fitness targets, the range of possible genome values is limited. For f^{target} = uq in dimension 16 and 32, only one value was found. Also, the seeds of the medium fitness target in dimension 4 are very close, with a difference of $5.329 \cdot 10^{-15}$ rounded to 0.0. For the Rosenbrock function, the differences, shown in Figure 8.10, are more coherent. All seed qualities show similar results in their respective dimensions. Again,

[1] Results rounded to three decimal places.

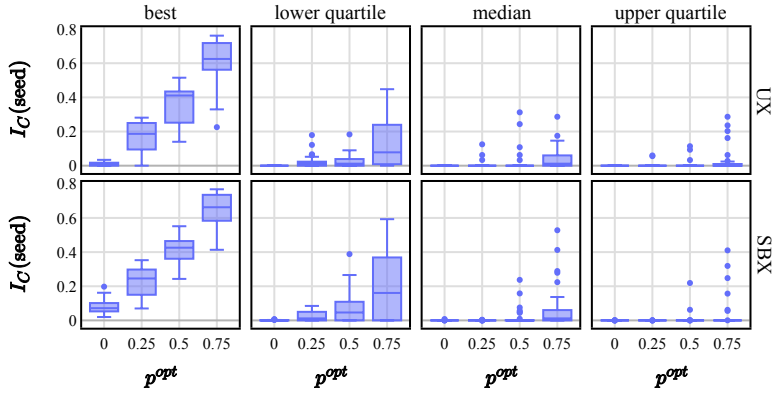


Figure 8.8: Seed influence for the Rosenbrock function ($\mathfrak{d}$= 16).

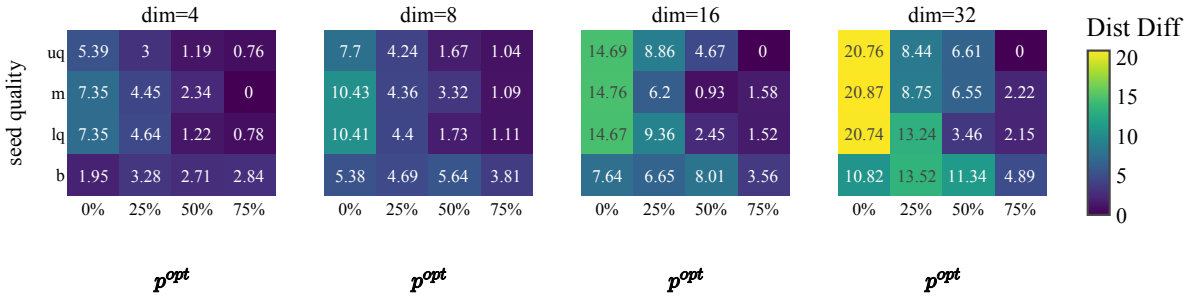


Figure 8.9: Difference in distance to x_{opt} for the close and far Rastrigin seeds[1].

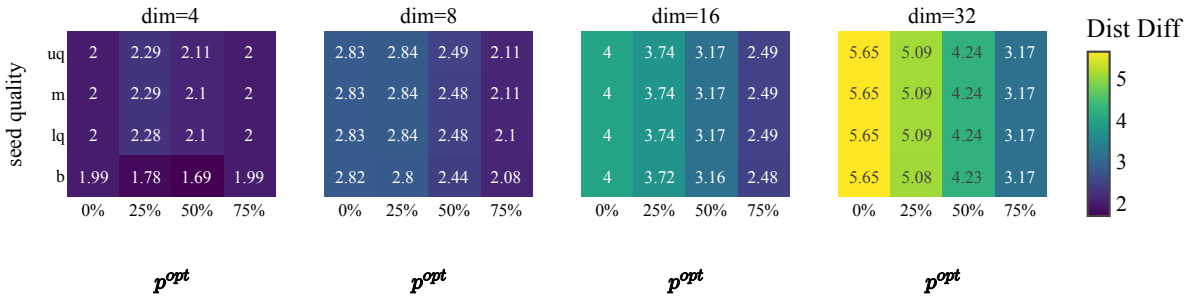


Figure 8.10: Difference in distance to x_{opt} for the close and far Rosenbrock seeds[1].

larger differences are found for the higher dimensions, with $\mathfrak{d}$= 4 actually showing very similar results for all seed configurations. Overall, the results for the Rosenbrock function are much lower than for the Rastrigin function.

Performance Differences between "Far" and "Close" Seeds

To gain an overview of the performance differences between the seed individuals generated close and far from the optimum in the search space, their performance results are shown in Table 8.1. Solutions marked with "c" mean the median performance of the seed generated close was better, "f" means solutions generated far had a better median performance, and

		Rastrigin								Rosenbrock							
		UX				SBX				UX				SBX			
		b	lq	m	uq	b	lq	m	uq	b	lq	m	uq	b	lq	m	uq
4	0 %	c	f	c	c	**c***	c	c	c	**c***	c	=	f	c	f	f	=
	25 %	f	f	c	c	**c***	f	c	f	c	f	c	f	**c***	f	c	f
	50 %	c	=	f	=	c	f	c	=	c	f	f	=	c	f	=	f
	75 %	c	=	=	f	c	c	c	c	f	=	f	=	c	f	c	f
8	0 %	f	f	f	c	**c***	c	c	=	**f***	c	c	f	c	f	f	c
	25 %	f	f	=	f	**c***	f	c	=	f	c	f	f	c	c	c	f
	50 %	f	f	=	=	**c***	f	f	=	c	c	f	f	c	c	c	c
	75 %	c	=	f	=	c	c	c	c	f	f	f	=	c	c	c	f
16	0 %	f	f	=	=	**c***	c	c	c	f	f	c	c	**f***	f	f	=
	25 %	c	=	=	=	**c***	f	f	f	f	c	c	c	**f***	f	c	c
	50 %	f	f	=	=	**c***	f	c	c	f	c	=	f	f	f	c	f
	75 %	f	=	=	=	f	c	f	=	f	f	f	c	f	f	f	c
32	0 %	**c***	c	=	=	**c***	c	f	f	f	f	c	=	**f***	f	f	f
	25 %	**c***	=	=	=	**c***	c	c	f	**f***	f	f	f	f	c	c	=
	50 %	c	=	=	=	**c***	f	=	=	**f***	f	c	f	f	f	c	=
	75 %	=	=	=	=	c	f	f	=	**f***	f	f	f	**f***	f	f	=

Table 8.1: Performance comparison between the close (c) vs. far (f) seed individuals.

"=" means the median was equal. Entries marked with a * showed a statistical performance difference using the Man Whitney U test for a $p = 0.05$.

Performance Differences

⇒ "Close" seeds do not seem to perform better than seeds "far" from $\vec{x}^{opt}$.

Interestingly, we do not find strong evidence for the hypothesis of seed individuals closer to x^{opt} performing better. First, only a few tests actually show a statistically significant performance difference (23 of the 256 tests). Furthermore, for the Rosenbrock function, in eight of the ten instances with significant differences, the far seed has the better median performance. For the Rastrigin function, the seed close to x_{opt} performed better in the instances where a significant difference was found, especially for SBX. This indicates the fitness landscape of the problem, and the recombination operator, influence whether the algorithm can take advantage of individuals closer to the optimum in the search space. One observation which can be made is that the statistically significant results are only occurring in the best fitness quality. This is to be expected that the best fitness seeds show the strongest differences in performance, as results of the previous section have already found their influence on the result to be the largest. However, we still find many instances with no significant performance differences. Also to note is that the larger differences in the distance to x^{opt} for the p^{opt}= 0% seeds found in Figure 8.9 and Figure 8.10 did not seem to translate to larger performance differences. This is a further indication of the distance to the optimum in the search space being a less significant factor for the same fitness individuals.

Impact Differences between "Far" and "Close" Seeds

Similar to the performance differences shown in Table 8.1, the differences in seed influence on the result can be seen in Table 8.2. The letters again show the comparison of the median impact results, with statistically significant differences (Man-Whitney U with $p < 0.05$) again being marked with a *. It has to be noted that some tests only show differences in their outlying solutions, resulting in an equal median performance with a statistically significant influence difference, with overall quite similar results.

Table 8.2: Influence comparison between the close (c) vs. far (f) seed individuals.

		Rastrigin								Rosenbrock							
		UX				SBX				UX				SBX			
		b	lq	m	uq	b	lq	m	uq	b	lq	m	uq	b	lq	m	uq
4	0 %	c*	=	=	=	f*	f	=	=	f*	=	=	=	f*	=	=	=
	25 %	f	=	=	=	f*	c	=	=	f*	=	=	=	f*	=	=	=
	50 %	=	=	c	=	f*	c	=	=	f	=	=	=	c	c*	=	=
	75 %	=	=	=	=	f	f	f	=	c	=	=	=	f	=	=	=
8	0 %	c	=	=	=	f*	=	=	=	=*	=	=	=	c*	c*	=	=
	25 %	c	=	=	=	f*	=	=	=	c	=	=	=	f	=	=	=*
	50 %	=	=	=	=	f*	c	=	=	c	=	=	=	c*	c	=	=
	75 %	=	=	=	=	f	f	c	=	c	c	=	=	c	c	=	=
16	0 %	c*	=	=	=	f*	f*	=	=	f	=	=	=	c	c*	=	=
	25 %	c*	=	=	=	f*	f	=	=	c	=	=	=	c*	c*	=	=
	50 %	c	=	=	=	f*	c	=	=	c	=	=	=	c*	c*	=	=
	75 %	=	=	=	=	f*	c	=	=	c	c	=	=	c*	c*	=	=
32	0 %	c*	=	=	=	f*	f*	=	=	f	=	=	=	c	c*	=*	=
	25 %	c*	=	=	=	f	f	=	=	c*	=*	=	=	c*	c*	=*	=
	50 %	c*	=	=	=	f*	f	=	=	c*	c*	=	=	c*	c*	=*	=
	75 %	=	=	=	=	f	c	=	=	c*	c*	=	=	f	c*	c*	=

Impact Differences

⇒ "Close" seeds do not show a larger impact than seeds "far" from x^{opt}.

Similar to the evaluation in the previous section, we find more differences in the influence data compared to the performance, with a better performance usually also translating to a higher influence. One notable exception is found in the Rastrigin problem for the UX operator in $\mathfrak{d}$= 32. The close seed performs better, but is actually found to have a lower influence on the result compared to the far seed. While we do see more statistical differences in the heritage data, again, the close individuals are not leading to a clearly higher influence, with very mixed results similar to the performance evaluation. Overall, we do not see much evidence for the individuals generated "close" to the optimum to outperform the "far" ones, as results are quite similar and mixed among the significantly different results. This indicates that individuals with genomes closer to the optimum solution are not producing better offspring than solutions further from the optimum, which seems counterintuitive. However, to find the reason for this, a deeper evaluation is required in future works.

8.4 Summary

Chapter Overview

This chapter investigates the effects of optimal genetic material in the initial population of EAs. The underlying goal is to identify influential individuals for the search process and to better understand the role of the genome in the evolutionary process. For this, experiments were designed, seeding a randomly generated population with pre-computed individuals, containing different amounts of optimal genetic material. This results in the comparison of two types of genomes. First, genomes which contain optimal genetic material, but also "bad" genes that drag down the fitness. Secondly, genomes in which all genes are mediocre. Furthermore, these individuals were generated for different fitness targets and with different distances to the optimum in the search space. Three test functions were used for the evaluation: The Sphere function, the Rastrigin function [36], and the Rosenbrock function [37]. Each in four search space dimensions $\mathfrak{d}$=[4, 8, 16, 32]. Furthermore, the effects of different crossover operators, which combine or copy the genome values, were assessed. Results were evaluated both in terms of performance and population dynamics, using the T-EA [114] to compute the influence of the seed individual on the result.

Main Findings

In our evaluation, we could find four main effects when seeding different genome compositions into the initial population of EAs.

1. *Amount of Optimal Genes*. The evaluation clearly shows the benefits of including optimal genetic material. Increasing the number of optimal genes in a genome resulted in a better performance of the algorithm in most tests. Analysing the heritage data clearly links these performance gains to the optimal genetic material. This is an interesting consideration when solving real-world problems, seeding the initial population with problem knowledge.
2. *Fitness of the Seed*. Seed fitness was also an important factor. As worse fitness individuals have a lower chance of surviving the selection process, they are also less likely to pass their genetic material, meaning the optimal genes cannot be exploited by the algorithm.
3. *Gene Copying vs. Gene Combining Crossover*. Also interesting is that little differences in the use of this optimal genetic material between the gene-combining SBX and the gene-copying UX crossover operators could be found. This contrasts with findings in the previous chapter, where different crossover operators lead to large differences in population dynamics. Both seem to exploit the optimal genes of the seed similarly. While this may seem trivial for operators only copying the gene values, it is interesting to see that the same effects are also found when combining the genome values.
4. *Closeness to x^{opt}*. Interestingly, we did not find the seeded individuals who have a closer distance to the optimum to significantly influence the convergence behavior or performance of the algorithms. This is unexpected as, intuitively, it should be easier to move towards the global optimum when the starting point is closer to it. This raises the question of why this is the case.

Limitations and Future Work

In this chapter, the influence of the genome on the result of EAs is explored, showing interesting trends and results for the influence of optimal genetic material as well as differences in the distance to the optimal solution in the search space. However, the limitations of the test cases need to be acknowledged. The five most relevant limitations and challenges for future work are:

1. *A limited amount of test functions is used.* Usually, evaluating EAs is done with a multitude of benchmarking functions featuring different fitness landscapes. The bbob functions [34] are a good example for this. However, generating the seed individuals and evaluating the algorithm beyond population dynamics creates a large overhead and is a current bottleneck for larger evaluations. Furthermore, the test functions are not shifted or rotated, like in the coco [35] framework, as this would require generating new seed individuals for each function instance. Future work needs to address this to enable testing with more test functions of a wider variety of fitness landscapes.
2. *Only genome composition and distance to the optimum are targeted.* While the results in this paper show interesting trends towards the quality of a genome, there are other factors which are not covered in this paper. The quality of the genome in one individual also depends on the population around it and the algorithmic configuration used. More general trends can only be found when also considering these

aspects. However, introducing more variables to the evaluation is difficult within the current capacity.

3. *Only one permutation of seed individual is tested.* This is aimed at the second simplification of the seed individual generation. A different permutation of the genome might affect the result. If the optimal genetic material is, for example, found in a genome where all other individuals of the population only feature poor material, the influence of that individual is potentially increased. We believe this to be one factor in the outlying results found in the evaluation. Again, further research into this is required.
4. *Only one non-optimal value is calculated for each seed.* This is aimed at the first simplification of the seed individual generation. While these simplifications are necessary, they arguably produce unrealistic individuals when compared to a random initialization. Generating individuals with a wider variety of genome values from (near) optimal to worst would be better, but also dramatically increase the number of test configurations.
5. *Results are only generated for the genome quality in the initial population.* Currently, the generated individual is seeded into a randomly generated population. However, the quality of a genome also depends on the surrounding population, so results for a randomly generated initial population vs. an already converged one might vary. Furthermore, the number of generations, or how many resources the algorithm has to evolve the given material, is also a potential factor not covered here. To gain a deeper understanding of what is desirable in a genome, these two factors have to be evaluated in the future.

Part III

Multi-Objective Population Dynamics

Related Work 9

This chapter provides an overview of related works on population dynamics in MOEAs. Here, we specifically only focus on findings from experimental results. An overview of existing tools and approaches for this can be found in Section 3.3. Similar to this previous section, we separate between findings on the population level, the individual level and the gene level. Furthermore, research on population initialization of MOEAs is briefly discussed, as some parallels can be found in the two areas. A short summary of all findings is provided in the final section of this chapter.

9.1 Population Initialization 97
9.2 Population Dynamics . 98
9.2.1 Population Level 98
9.2.2 Individual Level 99
9.2.3 Gene Level 99
9.3 Summary 100

9.1 Population Initialization

This section is largely based on the author's publication [120].

Research on different population initialization techniques can be viewed as related work in the search for influential individuals. In general, the strategies for population initialization are very similar to the ones discussed for SOEAs (see Section 6.1), both for the areas of sampling and seeding. A more general overview and classification of approaches can be found in [145]. Here, we mainly focus on the findings regarding the population dynamics.

Sampling

In general, the same sampling strategies which exist for SOEAs (see Section 6.1 have also been applied to MOEAs. The most standard technique for initializing a population is uniformly random. Other approaches, like LHS [22], GSS [124], OBL[125], or quasi-random numbers such as Sobol sampling [126], have also been applied. Their impact on the performance of MOEAs has been studied in [146–148]. Some general trends suggest that NSGA-II and non-dominated sorting genetic algorithm III (NSGA-III) do perform better using Sobol sampling for large-scale problems [146, 148], or MOEA/D to benefit from OBL [148]. However, while general trends can be observed, the reason for certain strategies working well for some algorithms but not others remains unclear. As Gong et al. mention in [148]: "No initialization method works well on all the examined problems. These observations explain why there exists no clear guideline about the selection of an initialization method for each EMO algorithm".

Seeding

For SOEAs, seeding primarily means inserting known or expert solutions into the initial population (see Section 6.1). However, for MOEAs, seeds are also generated as a known good solution for just one objective, which can be obtained with a pre-optimization for each objective separately. Gong et al. [145] call this "heuristic solutions inclusion strategy". One of the earlier works of seeding in MOEAs comes from Hernandez-Diaz et al. [149] and uses this technique. A gradient-based method was used to generate seeds for a MOEA, tested on the ZDT [38] problems. Friedrich and Wagner conduct a large computational study of different seeding techniques, comparing five algorithms with 48 test problems (the DTLZ, LZ, ZDT, and WFG test suites) in [121]. They find seeding to be beneficial,

with a large impact for some algorithms on certain problems. They noted that the experiments could not reveal why this is the case for a particular combination of seeding, algorithms and function landscapes. In [122], the initial population of MOEAs are seeded with known Pareto-optimal solutions for the DTLZ 1–4 problems. Results showed that even seeding one optimal solution can be very beneficial. In [150], the initial population was seeded with greedy solutions for variations of the multi-objective travelling salesman problem. Besides the performance evaluation, in both [122, 150] an algorithm behavior analysis is conducted, visually showing that the population tends to converge towards the seed solutions in the search space. Similar to [122], Sato, Sato, and Yamada [151] seed solutions obtained by a similar test problem into the initial population of NSGA-II. For this, they use the ZDT benchmarks, shifting the problems to create similar instances. The aim was to create a scenario closer to the real-world, as solutions to similar problems might already exist when dealing with new optimization tasks. They overall report similar benefits to [122].

The Answer is in the Search Space

In the quest of finding a good starting population for MOEAs, findings in this section take a top-down approach, competitively testing different initialization strategies. While general trends can be observed, no clear answer to why certain initialization strategies work for some algorithmic configurations on some problems, but not for others, can be given so far. We believe the answers to these questions lie in the search space. The next section focuses on the current state of evaluation methods for algorithmic behavior in this space, also commonly referred to as population dynamics of EAs.

Comment on the Benchmarks Used

Finally, we want to comment on the benchmark problems used in the discussed studies on population initialization in MOEAs. Most works, especially for seeding, claim the potential for performance improvements [121, 122, 149–151]. However, we would like to highlight the test problems used in these studies, especially the properties of the PSs, which were described in Section 2.4.2. Of the described papers studying this area, only one uses real-valued benchmarks with complicated PSs (LZ problems) [121], and one using combinatorial optimization problems [150]. All other studies rely on problems with easy PSs, meaning most of the genome in these algorithms needs to converge to only one single optimal value.

9.2 Population Dynamics

Only very few works focus on evaluating the population dynamics of MOEAs, especially compared to SOEAs (see Section 6.2.). Here, we again present the findings in this area for each population dynamics level.

9.2.1 Population Level

Dimensionality Reduction of the Search Space

The first approaches of evaluating a population in the search space use dimensionality reduction techniques to make the high dimensional search space visualizable [152].

Search Trajectory Networks (STN)

STN, proposed by Ochoa, Malan, and Blum [71], have also been adapted for the use with MOEAs [72]. STNs visualize the path of a population through the search space in a graph-based model, showing frequently

visited areas and deceptive points where the algorithm gets stuck. In their study, Lavinas, Aranha, and Ochoa compare NSGA-II [30] and MOEA/D [31] on the continuous UF benchmark set [49]. Evaluating the search trajectories, they found MOEA/D to explore the search space more extensively than NSGA-II. Furthermore, the performance of NSGA-II was more dependent on having good starting points than for MOEA/D. For combinatorial optimization, however, a different result was reached [80]. Here, NSGA-II [30] did sample the search space more thoroughly in experiments using the ρmnk-landscapes [153]. However, for larger problem instances, the authors note that NSGA-II [30] was attracted to lesser quality solutions, so while it had a higher rate of discovery, it was also getting trapped more easily [80].

Evaluating influential algorithm components using STN

Lavinas et al. [79] evaluate the influence of different algorithmic components for an automatically configured MOEA/D algorithm. The irace [154] package was used for automatic algorithm configuration. Tests were run with the DASCMOP [155] problems, a constrained multi-objective benchmark set. Besides HV and IGD as performance measures, they use STNs to evaluate the population dynamics for each algorithmic configuration. In their results, they find the restart strategy and the update strategy of the algorithm to be the most influential components. This study was repeated by Lavinas et al. two years later, with a similar setup, however this time also using problems real-world analytical problems from the CRE family [52], and the real-world simulation MOPs MAZDA [157] and MOON benchmark [158]. This time, results of the most influential components were more mixed, with the additional problems showing different components as most influential compared to the DASCMOP, suggesting the fitness landscape to play a role in the population dynamics and algorithmic design.

9.2.2 Individual Level

Exploration and Exploitation

Liu, Črepinšek, and Mernik [159] compare the two algorithms strength Pareto evolutionary algorithm 2 (SPEA2) and vector evaluated genetic algorithm (VEGA) in terms of their population dynamics on the multi-objective 0/1 knapsack problem. They use their exploration and exploitation metrics developed in [96], quantifying exploration and exploitation in MOEAs using genealogical data. With these metrics, they show VEGA to have a higher exploitative power, with SPEA2 being more balanced in exploration and exploitation. Liu, Črepinšek, and Mernik argue that this is the main reason why SPEA2 was able to perform better in their experiments.

9.2.3 Gene Level

No Studies Exist

To the best of our knowledge, there exist no related studies outside the ones published for this thesis.

Table 9.1: Literature overview of papers evaluating population dynamics of MOEAs, excluding the contributions of this thesis.

Author(s)	Work	Year	Level	Description / Contribution	Algorithm
Liu, Črepinšek, and Mernik	[159]	2012	Individual	Measuring exploration vs. exploitation	GA
Lavinas, Aranha, and Ochoa	[72]	2022	Population	STNs for MOEAs	EA
Lavinas et al.	[79]	2022	Population	Influential components of MOEAs using STNs	EA
Ochoa et al.	[80]	2023	Population	STNs for combinatorial MOPs	GA
Lavinas et al.	[156]	2024	Population	Extending [72] with real-world problems	EA

9.3 Summary

Chapter Overview

In this chapter, the related works researching the population dynamics MOEAs are presented. First, population initialization approaches are discussed. This is followed by the discussion of the population dynamics for each level of population dynamics. We found that fewer works focus on the population dynamics of MOEAs compared to one single objective (see Chapter 6). An overview is presented in Table 9.1. Nevertheless, we can observe two interesting trends across the literature.

NSGA-II Performance depends on good Initial Population

One finding across multiple studies seems to be that NSGA-II is more dependent on a good initial population than MOEA/D. Lavinas, Aranha, and Ochoa [72] showed this by evaluating the search dynamics using STNs. Studies on different sampling techniques [146, 148] for MOEAs reached similar conclusions, where NSGA-II was found to perform better using the more uniformly distributed Sobol sampling, while MOEA/D did not see the same performance benefits.

MOEA/D is more Explorative

Similarly, for continuous problems, it has been shown that MOEA/D explores more regions of the search space [72], which might be a reason why MOEA/D benefits less from a more different sampling strategies. However, results for combinatorial optimization showed NSGA-II to have better exploration of the search space [80].

Research Gap

Only few studies focus on the population dynamics of MOEAs. The majority of evaluations are done on the population level using STNs. These studies provide important information about the search behavior of NSGA-II and MOEA/D and highlight the value of evaluating MOEAs beyond their performance. However, we also find a gap for the other levels. To the best of our knowledge, no related works on the gene-level population dynamics exist for MOEAs. The dynamics of SOEAs and MOEAs are inherently different, as will be discussed in more detail in later sections. For this reason, we cannot assume findings of SOEAs to also hold for multiple objectives. In the following chapters of this thesis we aim towards this, by discussing the differences between optimizing a single or multiple objectives, with experiments building towards a more solid base of knowledge for the population dynamics of MOEAs.

Influence of Population Size on the Population Dynamics of MOEAs

10

This chapter is largely based on the author's publication [115].

The goal of this chapter is to set a starting point for research into the gene-level population dynamics of MOEAs. For this, we evaluate the influence of the population size on the search behavior of MOEAs as a proof of concept. For this, we choose a typical algorithm configuration, running it with a small ($n = 10$) to a large ($n = 300$) population size. With this, we will gain first insights into the gene-level dynamics of MOEAs.

10.1 Motivation and Overview 101
10.2 Test Setup 102
10.3 Results 103
10.3.1 Performance 103
10.3.2 Population Dynamics . 104
10.4 Summary 109

10.1 Motivation and Overview

Motivation

Using EAs requires multiple parameters and operators to be chosen. Finding the best setting for each hyperparameter is not trivial, as they are dependent on the problem itself, and are also interdependent between each other. One of these parameters required in every algorithmic configuration is the size of the population to be evolved. The optimal choice for this is often considered to be a trade-off. A higher number of individuals in a population has a higher exploration capacity, as it can hold a larger base of genetic material. However, with a higher population size the computational effort also increases, as more individuals need to be evaluated in each generation. For a fixed computational budget, the algorithm runs for fewer generations.

Small Overview of Related Works

The task of finding the optimal choice of population size is an old research question. There have been many studies on this parameter for specific problems and algorithmic configurations [160–167]. Mora-Melià et al. [160] and Fukumoto and Oyama [161] create a measure for the efficiency depending on computational effort and quality of solutions found. While a larger population size did produce better results, both showed that they also can be less efficient. Weise et al. [162] also argue that the optimal size of the population is dependent on the computational budget. Chen et al. [163] showed that a large population size can even be harmful under certain circumstances, trying to solve the TrapZeros problem. In [164], Hu and Banzhaf propose the rate of evolution (R_e) as a measure in evolutionary computation. Experiments showed larger populations not to have a higher R_e. However, a larger population size still reached a better average fitness. There are also approaches trying to approximate the optimal size of the population using the complexity of a problem, like Alander [165]. Hidalgo and Fernandez [166] evaluate the population size parameter linked to the number of generations and number of runs for a test. Their experiments used a fixed number of function evaluations, with the largest population run once, reaching the best results in their tests. Besides research on the population size, there are also approaches trying to solve the parameter tuning as an optimization problem itself, like the parameter-less genetic algorithm [168] or the i-race algorithm [154]. Finally, there is also research on variable population sizes in EAs [169, 170] or the offspring population size [171]. However, this section is focused on researching the effects from the size of the population for

traditional ($\mu + \mu$) EAs, meaning the population size is fixed and the number of offspring generated is equivalent to the parent population size.

Chapter Overview

In this chapter, the convergence behavior of the initial population for MOEAs with different population sizes is systematically evaluated. The purpose of this is to obtain a first idea on population dynamics for MOEAs. This is done in terms of performance, and convergence behavior of the T-EA, using the metric of contributors and the influence of mutation. Furthermore, the influence of the NDS on the final population of the algorithm is explored. Our results for SOEAs in Chapter 8 already showed the fitness of an individual not to be the most important factor [118]. Rather, the presence of good genetic material proved to be a good indicator for an influential individual, which leads an algorithm towards the optimum. It will be interesting to observe a first trend for MOEAs. To evaluate the effects of different population sizes, we use the common NSGA-II algorithm [30] and evaluate the results of nine different benchmark problems run with five different sizes of population 10, 50, 100, 200, and 300. The performance of the test runs and the influence of the initial population and mutation are evaluated. Furthermore, the number of contributors is discussed with a special focus on the role of NDS.

10.2 Test Setup

Population Sizes

We use five different population sizes of 10, 50, 100, 200, and 300. This represents very low to high population size for EAs.

Algorithmic Setup

As a test algorithm, NSGA-II [30] was chosen, as it is one of the most widely used MOEAs. For crossover, SBX [25] is used with $\eta = 20$ a crossover probability of 90%. Individuals are selected for crossover with a binary tournament. PM [26] was selected for mutation with an η of 20 and a mutation probability of 1 %. Each test is run for 100 generations. All tests are run independently 31 times. Compared to the other evaluations using the T-EA, each test is also run with a different initial population, as keeping it the same was not required for the evaluation in this paper, providing further diversity for the test cases.

Benchmark Problems

Nine different test problems are used: ZDT 1-3 [38], UF 1-3 [49], and DTLZ 1-3 [39]. Each problem is used with the respective genome sizes they were proposed with. Table 10.1 shows an overview of the test problems.

Table 10.1: The dimensionality in search and objective space of the used test problems

problem	#variables	#objectives
ZDT1	30	2
ZDT2	30	2
ZDT3	30	2
UF1	30	2
UF2	30	2
UF3	30	2
DTLZ1	7	3
DTLZ2	10	3
DTLZ3	10	3

Evaluation Metrics

The performance of the algorithm is evaluated using the GD [59] and IGD [172] metrics. The heritage data of the T-EA is evaluated using the metric of contributors, and the I_C of the mutation operator.

10.3 Results

Starting the evaluation, we briefly discuss the performance of the problems and population sizes using the GD [59] and IGD [172] metrics. Following that, the number of contributing individuals is evaluated for each population size. After this, we have a closer look at the NDS as the best individuals of the initial population, and if show a higher chance to contribute their genetic material. Finally, the impact of mutation is discussed in this section.

Table 10.2: Median GD and IGD values for each problem and population size.

metric	**GD**					**IGD**				
pop size	**10**	**50**	**100**	**200**	**300**	**10**	**50**	**100**	**200**	**300**
ZDT 1	0.83578	0.05470	0.01791	0.00982	0.00759	0.75805	0.05342	0.01691	0.00766	0.00490
ZDT 2	1.305101	0.083267	0.03088	0.01226	0.00884	1.57157	0.09766	0.03254	0.01166	0.00787
ZDT 3	0.82666	0.03181	0.01052	0.00747	0.00661	0.66415	0.04738	0.01269	0.00566	0.00350
UF 1	0.37683	0.05228	0.03943	0.03188	0.03178	0.41403	0.15268	0.11234	0.09138	0.07635
UF 2	0.16878	0.05103	0.03363	0.02416	0.02135	0.20638	0.07614	0.05625	0.04459	0.03901
UF 3	0.51818	0.46349	0.44899	0.42875	0.41249	0.57915	0.46846	0.43948	0.39101	0.36167
DTLZ 1	47.17126	13.09367	4.28797	1.06734	3.37583	17.17650	3.3758	1.45376	0.50197	0.16768
DTLZ 2	0.06598	0.05631	0.05730	0.05746	0.05747	0.26965	0.10185	0.07189	0.05151	0.04226
DTLZ 3	157.49812	83.52813	60.19957	24.37980	15.78323	78.39477	26.85434	18.08392	9.09572	7.39326

10.3.1 Performance

Performance Findings

Starting the evaluation of the results is done giving a short overview on the performance findings. Table 10.2 shows the median performance over the 31 test runs in terms of GD [59] and IGD [172] for each test configuration. Larger population sizes clearly show a better result in nearly all cases. This is expected, as each test configuration was run for 100 generations, resulting in more function evaluations for bigger population sizes. The only exception to this is found for the DTLZ 1 problem, which showed a higher median GD value for the population size of 300 compared to the result for the population size of 200. It is also to be noted that the DTLZ 1 and DTLZ 3 problems showed very high GD and IGD values, indicating a worse algorithm performance than for the other problems. Keeping the number of function evaluations the same for each test configuration was not strictly required for our test purpose, as the aim was evaluating the population dynamics.

10.3.2 Population Dynamics

Number of Contributors

Number of Contributors in the Final Generation

⇒ Some differences can be found between the used test problems, but overall the number of contributors for each population size stays quite similar.

The number of contributing individuals for every test run can be seen in Figure 10.1a. Each box plot shows the results of all problems for one population size. Upon first inspection, the results for the population sizes 50, 100, 200, and 300 look similar. In all of these four graphs, the problems ZDT 2, UF 1, and UF 3 show the lowest median number of contributors and DTLZ 2 the highest. All other feature median values in close range with each other. Differences can be found in the spread of the results, for example with the UF 3 problem with the population size of 200 having a low spread of results, while the other population sizes feature a higher spread. The results for the population size of 10 are a little different. While the DTLZ 2 problem still shows the highest median number of contributors, the DTLZ 1 and ZDT 2 problems show higher results in the maximum and outlying values. Overall, the results and spread of solutions seem a lot more similar between the problems than for the other population sizes. A reason for this could be the very low population size of 10. While some differences between the different problems can be found for each population size, it has to be noted that the results are overall all fairly close to each other.

Number of Contributors vs. Performance

⇒ We do not observe a correlation between the number of contributors and better algorithmic performance.

Comparing the results in Figure 10.1a to the performance values in Table 10.2, no real correlation can be found. The high GD and IGD values of DTLZ 1 and DTLZ 3 do not seem to translate into a higher or lower number of contributors. The higher contributor count for the DTLZ 2 problem and the lower values for the ZDT 2, UF 1, and UF 3 problems also are not reflected in the performance metrics.

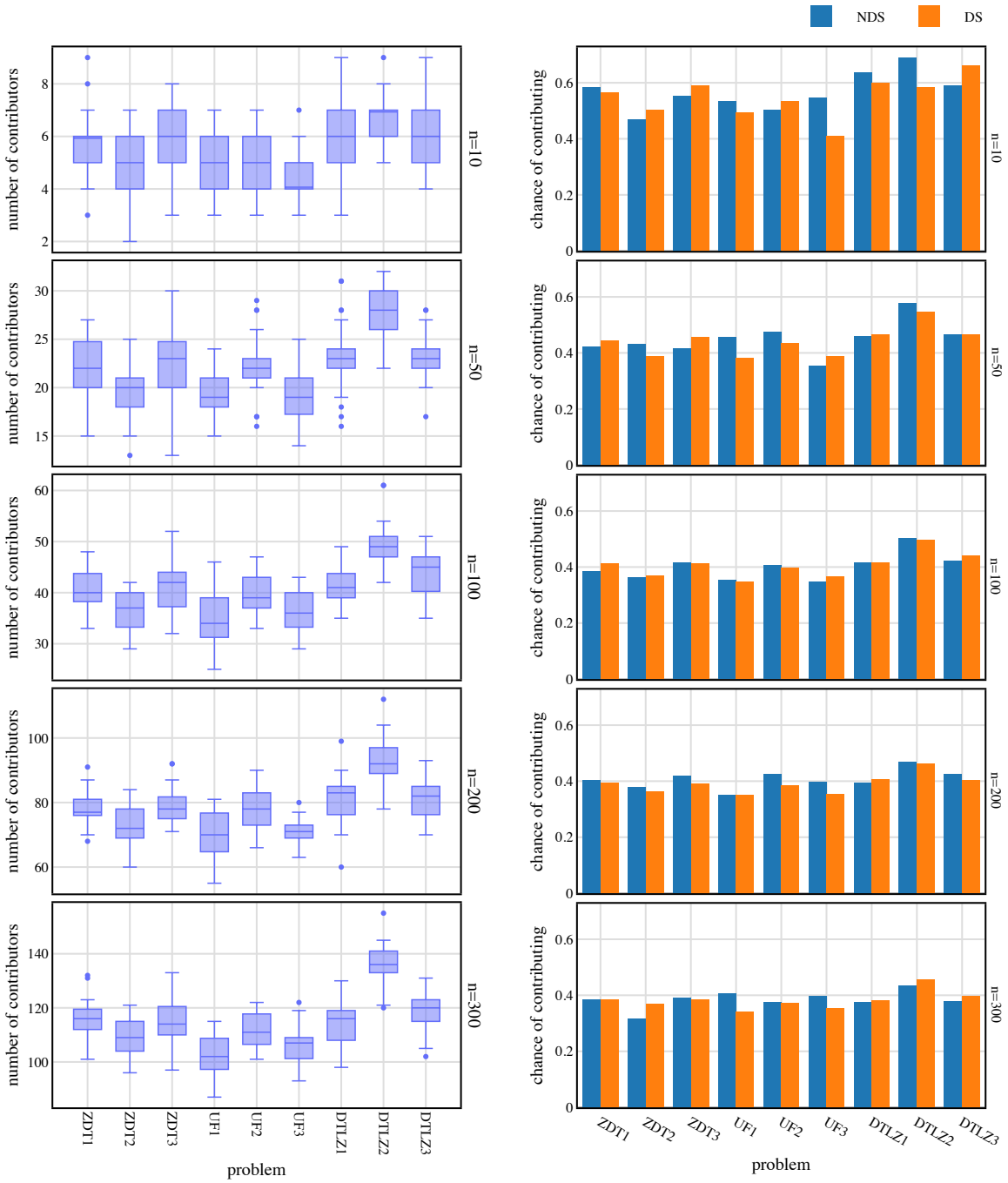
Table 10.3: Lowest, median, and highest number of contributors per population size for all problems as absolute and relative values.

pop size	10	50	100	200	300
lowest	2/20%	13/26%	25/25%	55/27.5%	87/29%
median	6/60%	22/44%	40/40%	78/39%	113/37.6%
highest	9/90%	33/66%	62/62%	109/54.5%	149/49.6%

Percentage of Contributors

⇒ The percentage of contributing individuals is lower for larger population sizes.

For a better overview, the lowest, average, and highest number of contributors for all problems and population sizes are shown in Table 10.3. In absolute numbers, more individuals contribute to a result when the size of the initial population is bigger. However, it is interesting to see that in relation to the size of the initial population, the percentage of individuals contributing to the result is lowering with increased population sizes for the median and highest values. From a statistical perspective, the chance of survival should be the same for all population sizes, as the same operators, with the same probabilities and selective pressures have been used. This lowering in the percentage of contributors might hint at what one might call the need for genetic material. Or, in other words, one could theorize that for larger population sizes, the initial population offers an abundance of genetic material, meaning fewer individuals need to survive. In our tests, we did not find a ceiling in the form of an absolute number. Here, tests with larger population sizes would be insightful. The one exception to the steadily declining percentage of contributors for larger population sizes is the lowest number of contributors recorded for all test cases. Here, the percentage of contributors is rising slowly for bigger populations, meaning the overall differences between each test



(a) Number of contributors in the final generation as box plots for all test runs, for each population size n.

(b) Probability of NDS or domonated-solutions (DS) from the initial population to be contributing to the final population, for each population size n.

Figure 10.1: Results for the population size tests [115].

run is decreasing for bigger populations. Only the population size of 100 shows a slightly outlying result for the lowest number of contributors.

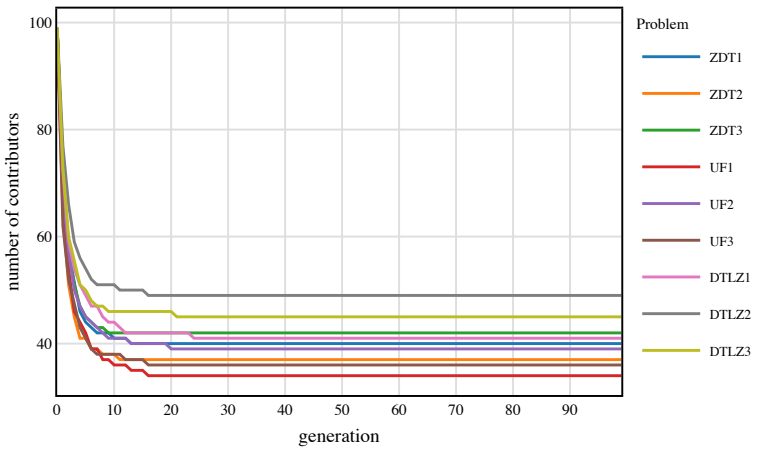


Figure 10.2: The median number of contributing individuals in every generation for the tests with the population size of 100 [115].

Number of Contributors throughout Generations

⇒ The number of contributors drops significantly in the first 20 generations, then becomes stable for all problems.

Finally, we want to show the development of the number of contributors over the generations. Figure 10.2 shows the exemplary median number of contributors for all problems with the population size of 100. This size was chosen representatively, as the median population size for all tests, as again the same behavior can be observed for the other population sizes. In the first generation, the number of contributors is equal to the population size. In the following generations, a rapid decline in the number of contributors can be observed, reaching a stable number of contributors from generation 20 onwards. This is similar to what we could observe for SOEAs in Section 7.3.1. Some problems, like ZDT 1 and ZDT 2, reach that stable number of contributors even sooner. This drop can be explained with the evolutionary process itself. The only way for a traceID to be discharged is for the individual they appear in to not be selected for recombination or survival. Tournament selection only selects a subset of the population for recombination, thereby removing some individuals from the process. Throughout the generations, the trace lists of each gene get longer, with the crossover operation inevitably combining the trace lists of the parents. As in the later generations, many traceIDs influence each gene, the genetic material of the initial population is already distributed across the trace lists of many individuals. Therefore, in later generations it is unlikely that a traceID is only present in one individual, not selecting specific individuals for recombination then is unlikely to lower the number of contributors. Nevertheless, it is interesting to observe that the median number of contributors does not change after the first fifth of the evolutionary process. While this convergence behavior might be largely explained by the evolution of the trace-list in the crossover process, the problem itself also seems to play a role in the convergence behavior, as the median values in Figure 10.1a for $n = 100$ already showed.

Influence of the initial Non-dominated Solutions

Comparing the Probability to Contribute between NDS and dominated solutions.

⇒ No evidence can be found that the NDS show a higher chance to contribute.

To test if the NDS of the initial population have a higher influence on the last generation, their probability to contribute is compared to the probability of dominated solutions. Figure 10.1b visualizes this comparison of the probability to contribute both for the NDS and the dominated solutions. The results show the average over the 31 test runs per configuration, for each population size n. For a better overview, all NDS are grouped in one category. As the NDS of the initial population can be considered to be the best initial solutions, it could be assumed that they might have a higher probability of contributing to the final population. However, the results in Figure 10.1b show a different result. It can clearly be seen that in nearly all cases, the chances to be found in the final generation are very similar, with only some exceptions. The results for the population size of n= 10 in Figure 10.1b show the most differences. For the UF 3 problem, the difference in the probability of NDS to be contributing is over 50 %, while the probability of the dominated solutions is only about 40 %. While this difference could be considered significant, no continuity between the results of each problem or population size can be found. While the UF 3 problem shows a higher probability of NDS to contribute for the population sizes 10 and 50, the results for the population sizes 200 and 300 show the dominated solutions to have a higher probability. Only for the DTLZ 1 problem, NDS consistently feature a marginally larger chance to contribute. On the other hand, the UF 2 problem features consistently better results for dominated individuals. Results for different population sizes are also mixed. We also again observe the percentage of contributing, both for NDS and dominated solutions, to drop for larger population sizes, as seen in Table 10.3. Furthermore, the DTLZ 2 problem again consistently shows the highest probability to contribute in Figure 10.1b and shows also the highest number of contributors in Figure 10.1a.

Why are NDS not Contributing More?

The results in Figure 10.1b show that the probability of contributing is not higher for the NDS. This indicates the quality of solutions does not increase the chances of surviving the evolutionary process to the final generation. This is an interesting observation, since the binary tournament used for mating selection, and the survival selection of NSGA-II both have a selective bias towards NDS. Here we find hints at the genome in MOEAs also playing a significant role if individuals are influential and beneficial in the search process, similar as shown for SOEA in Chapter 8 and [118]. However, more research towards this theory is required. The next chapter will get into more details for this question.

Mutation Impact

The influence of mutation in the final generation is presented in Figure 10.3, similar to the number of contributors before. The impact of mutation refers to the I_C for all traceIDs of mutations in the final generation.

Impact of Mutation

⇒ Mutation Impact is dependent on the benchmark problem used.

The influence of mutation in the final generation for the population sizes of 50, 100, 200, and 300 are similar. The median mutation impact for the ZDT 1 and ZDT 2 problems is around 0.6 for all of these population sizes, while the median of the ZDT 3 problem is always a little smaller. The results of the DTLZ 1, DTLZ 2, and DTLZ 3 problems all are found around or a little above a mutation impact of 0.2. Values for the UF problems are not as close as for the other two problem types. The UF 3 problem always shows the smallest impact of mutation, with around 0.12 for every test run. UF 2 is the highest among the UF problems, with a median between 0.3 and 0.4. The UF 1 problem shows the most differences between each median for the four described population sizes, ranging from around 0.18 for the population size of 300, gradually increasing with a lower population size to around 0.28 for the size of 50. The population size of 10 again has the most differences between all other configurations, with a considerably lower influence in the ZDT problems. The differences for the other two problem types are lower, however, still higher than between the other four population sizes. It appears that the problem itself has not only an influence on the convergence of the algorithm, but also the influence of mutation. Previously, the influence of mutation could only be linked to the quality of the initial population [113]. Comparing the mutation impact to the performance metrics in Table 10.2 does not show a significantly higher or lower influence of mutation for higher or lower GD and IGD values (Table 10.2). This indicates that different test problems need a different amount of new genetic material throughout the evolutionary process to build successful offspring solutions.

Comparing Mutation Impact and Number of Contributors

⇒ No correlation between the number of contributors and the impact of mutation can be found.

Comparing the results of the mutation impact in Figure 10.3 with the previously discussed results of the number of contributing individuals in Figure 10.1a, also no real link can be found. While it could be assumed that test cases with a higher impact of mutation show a lower number of contributing individuals, this does not seem to be the case. Indeed, the UF 3 problem shows both, the lowest influence of mutation and also low numbers of contributing individuals. This means that while the number of individuals contributing to the final result is relatively similar for all five problems, the influence of each individual is lower in the ZDT problems compared to the DTLZ problems.

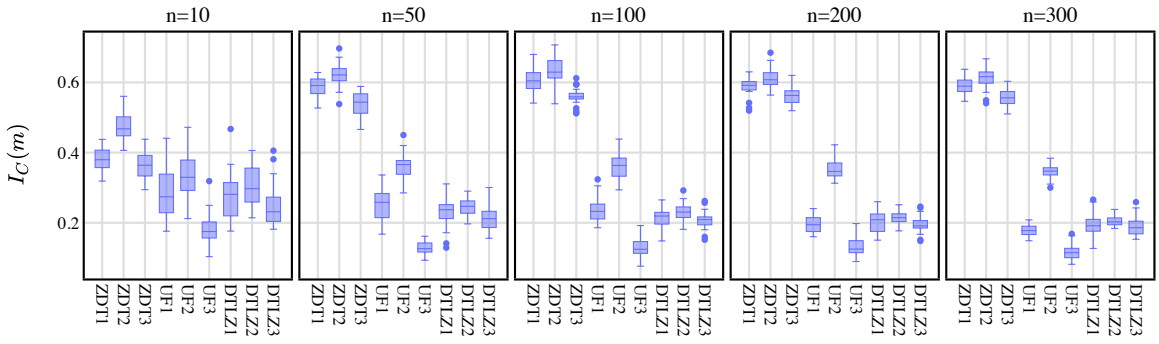


Figure 10.3: Results for impact of mutation in the final generation as box plots for all test runs. The top graph shows the results of the population size of 10, followed by 50, 100, 200, and 300 is shown in the bottom graph [115].

Theories for Varying Mutation Impact

Reasons for the differences in mutation impact could be a fast convergence to the Pareto front, as the probability of a better offspring is lower for already optimal solutions. However, the DTLZ 1 and DTLZ 3 problems show both a low convergence in the GD and IGD values (Table 10.2) as well as a low influence of mutation (Figure 10.3). The combinability of individuals, or the ability of a population to produce good offspring, can also play a role when it comes to the influence of mutation. However, as each test was run with a new randomly generated initial population, it can be assumed that the differences in the impact of mutation are due to the problem itself. Another possible explanation is the mutation operator working better for certain problems than for others, meaning it produces better values in certain problems, creating a higher influence of mutation, while not being as optimally suited for other problems. We believe the most probable reason for the varying impact of mutation to be linked to the problem itself. We assume that different test problems require a different amount of new genetic material through mutation to build successful offspring individuals. While the impact of mutation is likely dependent on the problem, the size of the initial population does not seem to have a big influence in the test data. Differences were only observed in the very low population size of 10 for the ZDT problems.

10.4 Summary

Chapter Overview

The goal of this chapter was to gain a first overview on the gene-level population dynamics of MOEAs. For this, the effects of different population sizes on the convergence were evaluated using the T-EA. The number of contributing individuals and the impact of the mutation operator was evaluated using the T-EA [114, 115]. Furthermore, differences in the chance to survive between the NDS and dominated solutions from the initial population were explored. Experiments were performed with the benchmarking problems ZDT 1-3 [38], UF 1-3 [49], as well as DTLZ 1-3 [39], and run with five different population sizes 10, 50, 100, 200, and 300.

Main Findings

The evaluation showed that a higher population size achieved a better convergence in all benchmarks, similar to the findings in the related work on single-objective problems. This was to be expected, as due to the focus of the evaluation the number of function evaluations was not kept the same, meaning larger population sizes had a larger search capacity. Discussing the gene-level population dynamics revealed interesting new findings. The number of contributing individuals as well as the influence of mutation seem to be dependent on the problem itself. However, they do not appear to be influenced by the performance of the algorithm. The size of the initial population also did not seem to be a big factor when it comes to the impact of mutation. The percentage of contributors was found to be lower for large populations. It could also be shown that the number of contributors is dropping fast in the first generations before reaching a static number. These findings lead us to the theory that EAs require a certain amount of genetic diversity in the population. For smaller populations, more individuals are required to satisfy this demand, while for larger population sizes, the algorithm is free to discard individuals which are not strictly needed for the search process.

Of course, for MOEAs, the population size is also dictated by the number of objectives, as more individuals are required to cover a PF of a high dimensionality. Data for the evaluation of NDS showed the rank of an individual not to translate into a higher probability to be contributing. Reasons for this could include the overall quality of the population to be fairly equal for all individuals, or better individuals of a population not to produce higher quality offspring than lower ranked individuals. This hints at similar findings that were reached in Chapter 8 and [118], where the probability of an individual being influential and beneficial in the search process was found to be dependent more on the genome, and not only on its fitness. The next chapter in this thesis is aimed at further evaluating this question on which individuals are influential in the search process of MOEA.

Future Work

Again, as this chapter is intended as a first proof of concept, further tests are required to reach more robust results. This would be especially interesting for very large population sizes of 1000+ individuals. It would also be interesting to dive deeper into some effects found here. For example, the impact of the mutation operator, which was largely unaffected by the population size, but seems to be influenced by the test problem. Another point for future research are the findings on NDS. It is unintuitive that we do not find these best individuals in the initial population to show a larger chance to pass on their genetic material than any other individual in the initial population. This question motivated research into influential individuals in the search process of MOEAs, discussed in the following Chapter 11. Besides the population size, research on more parameters and operators on the convergence of a MOEA will also be an interesting field of study.

Influential Individuals in MOEAs 11

This chapter is largely based on the author's publication [120].

This chapter aims at identifying influential individuals in the search process of MOEAs. We cannot directly transfer findings from the previous Chapter 8 due to the inherent differences between single- and multi-objective EAs. It is also not possible to create seed individuals the same way as was done before, as we cannot define what makes a good gene for MOPs. Due to these differences, we take a step back and use a slightly simpler test setup to gain more information towards the posed question. The initial population is again seeded with individuals of known good quality, as specified in Section 5.2. Computational experiments are done, comparing the search behavior of the two most population MOEAs, NSGA-II [30] and MOEA/D [31], on benchmark problems with different PSs properties.

11.1 Motivation and Overview 111
11.2 Test Setup 113
11.3 Evaluation 114
11.3.1 Performance 114
11.3.2 Population Dynamics . 116
11.3.3 Impact of the Single Seeds Across the PF . . 119
11.3.4 Impact of Combined Seeds Across the PF . . 120
11.4 Summary 122

11.1 Motivation and Overview

Motivation

The goal of MOEAs is to find a set of Pareto-optimal solutions, which are created over multiple generations by recombining and mutating the individuals in a population. The quality of individuals is assessed by fitness functions, assigning a score to how well they solve given objectives. These individuals not only represent solutions to the optimization problem, but also build the base for the ongoing search. While tools to evaluate the population dynamics in EAs exist, such as genealogical evaluations [62] or STNs [71], their use remains scarce. Algorithmic development is usually guided by the user's intuition and performance evaluations. However, these intuitions can also be misleading. For example, we typically assume that good fitness individuals are generating better offspring and are a driving factor for the evolutionary process. However, evaluations for SOEAs in Chapter 8 showed the genome to be a major factor both for the impact of individuals, and algorithm performance. Furthermore, the previous Chapter 10 showed the NDS of the initial population in MOEAs not to survive more than the initially dominated solutions. A similar trend was found in another previous work, in which we tried to create a quality measure for the initial population of MOEAs based on the fitness values [173]. Here, we frequently found cases where populations with an overall worse fitness were outperformed. This lead to the conclusion that the fitness of the initial population by itself is not a sufficient measure to estimate the quality of the initial population in MOEA. This opens the question, which individuals are influential in the search process of MOEAs?

Findings for SOEAs

As mentioned, previous research on uni-modal SOPs [118] has found that these optimal genes help EAs to converge. This was even when they appear in individuals not featuring the best fitness, otherwise consisting of genes far from their optimal value. This was tested in Chapter 8, by seeding individuals of the same fitness but different genome configurations into the initial population of an EA. Test results clearly linked performance improvements to the number of optimal genes

in the genome. This demonstrates the importance of the genome when trying to generate a good initial population. However, these findings cannot be directly transferred to MOEAs, as they are searching for a set of multiple solutions, usually not featuring one optimal value for each gene.

SOEAs vs. MOEAs

SOEAs feature one single optimum in the objective space, and, with the exception of multi-modal problems, one single optimum in the search space. This allows for comparing the quality of two individuals directly by their fitness. The quality of a single gene in the genome of an individual, given x^{opt} is known, can then also be compared. For MOEAs, however, this is not possible. They try to find a set of solutions, both in the search space and the objective space, which does not allow for comparing the quality of individuals or their genes using their fitness (recall Section 2.3). This also changes the convergence, as the algorithms now need to find a diverse set of individuals, with potentially a diverse range of possibly optimal values for each decision variable. These differences also influence how the algorithms search for solutions.

"Easy" vs. "Complicated" PSs

Early MOP benchmarks are frequently criticized for featuring unrealistic properties [40–45] (recall Section 2.4.2). One critique especially relevant here are the PS features. Li and Zhang [41] differentiate between problems with *easy* and *complicated* PSs. Recalling Section 2.4.2, in problems with easy PS, parts of the genome only control the distance to the PF, usually converging to one singular value [47, 48]. These include the ZDT [38], DTLZ [39], or WFG [46] benchmark suites, which are, to this day, frequently used. This was demonstrated by the example of the PS of the three-objective DTLZ problems in Equation 2.10 (Section 2.4.2). Problems with complicated PS do not feature these properties. Here, a range of values can be optimal for each gene. This differentiation becomes important when considering the search dynamics of MOEAs. Problems with simple PSs only need to find one optimal value for large parts of the genome, similar to SOEAs. Problems with complicated PSs need to find the right combination of values. This might be especially relevant for studies on population initialization, as seeding the single optimal gene value might be exploited by the algorithms, similar to SOEAs [118] (see Chapter 8). However, for all studies on population dynamics, discussed in detail in Section 9.1, only two use problems with complicated PSs[1] [121, 147], and one was conducted on combinatorial problems [150]. All other studies only use problems with easy PSs. This means the potential performance gains described in these studies might be overestimated due to the selected benchmarks used.

Chapter Overview

The goal of this chapter is to address the question of which individuals are influential in the search process of MOEAs, and if differences in the search behavior between problems with simple and problems with complicated PSs can be found. For this, we seed the initial population of MOEAs with known (good) solutions and track their heritage and influence in the evolutionary process. We will evaluate the influence of different seed locations on the PF for a set of various test problems. Using the heritage information, we can examine the influence of individuals in the search process beyond intuition and assumptions. Results in this work often confirm these general intuitions. However, we believe it is important to establish a solid base towards a deeper understanding of population dynamics in MOEAs.

[1] Both studies also use problems with easy PSs

11.2 Test Setup

Seed Individuals

To evaluate the population dynamics of MOEAs, seed individuals are created and evaluated. This is done as specified in Section 5.2. For the two-objective problems, two seeds $e1$ and $e2$ are generated at the edges of the PS, and one in the center c. Besides the single seeds, we test a combination of just the edge seeds $e1 + e2$, and a combination of all seeds $e1 + e2 + c$. This results in five seed combinations for two-objective problems ($e1$, $e2$, c, $e1 + e2$, $e1 + e2 + c$). For the three-objective problems, an additional seed for the third objective $e3$ is used, resulting in six different seed combinations ($e1$, $e2$, $e3$, c, $e1 + e2 + e3$, $e1 + e2 + e3 + c$).

Test Problems

15 different test problems from four benchmark families with simple and complicated PSs are used for the test. For the simple variants, we take ZDT [38] and DTLZ [39] problems. More specifically, we use the ZDT 1–3 problems, which are two-objective problems ($d_o = 2$), and we configure the DTLZ 1–3 problems with three ($d_o = 3$). For problems with more complicated PSs, we use the UF [49] test suite, which was proposed for the CEC 2009 competition and features some overlap with the LZ [41] problem suite. In this paper, the two-objective UF 1–3 problems are chosen alongside the UF 8–10 problems, which feature three objectives. Finally, we use the MACO problem [50] featuring a complicated PS. The problem features interesting variations, which allow adjusting the multi-modality and complexity of the problem. We use the *base* version of the problem, which is highly multi-modal, with many globally optimal PSs. Alongside this, the *weights=steep* variation, in which there is only one global PS with multiple local optima, is used. We choose the *p-norm=-10* variant, which changes the parameter dependencies of the base version. With these variations, we aim to isolate changes in population dynamics through certain problem properties, specifically the multi-modality and parameter dependency. The number of decision variables is kept as recommended in the original papers: $\mathfrak{z} = 10$ for all test problems, except for ZDT 1-3 with $\mathfrak{z} = 30$ and DTLZ 1 with $\mathfrak{z} = 7$.

Algorithmic Setup

The two most popular algorithms in the field of MOEAs, NSGA-II [30] and MOEA/D [31], were chosen for the tests. UX [24] is chosen for crossover, and PM [26] for mutation, with a probability of $1/\mathfrak{z}$ and an $\eta = 20$ are chosen. All tests are run with a population size of 91 over 100 generations using the pymoo [144] framework version 6.1.3. The code for the tests is available on Zenodo [174].

Evaluation Metrics

Each of the seed configurations is tested for all test problems with both algorithms. This results in 108 separate configurations with two-objective problems, and 84 additional configurations with three-objective problems. Each of these tests is repeated 31 times. To evaluate the results in terms of performance, we use IGD+ [57] and HV [56]. For the HV, the reference point is chosen following the suggestion of Ishibuchi et al. [58], as $f^{max} + \frac{1}{H}$, with H being the number of reference directories used. That is, $H = 90$ for the two-objective problems and $H = 12$ for the three-objective problems. The statistical significance between the performance of the seeded populations to the random population (r) is assessed using the Mann-Whitney U test, with $p = 0.05$. The population dynamics are tracked with the T-EA and evaluated using the I_C metric, as presented in Equation 4.16. For this, we show the influence of the seed individuals on the final result.

11.3 Evaluation

For the evaluation, first, the performance results are briefly discussed. Following that, the population dynamics data from the T-EA are evaluated. First, the impact of the seed on the NDS found by the algorithm presented. Following that, the impact of the seed(s) across the PF is visualized.

11.3.1 Performance

The average performance over the 31 in terms of IGD and HV for all test configurations is presented in Table 11.1 for the bi-objective, and in Table 11.2 for the three-objective problems. Statistically significant results are marked with a *, and results are rounded to three decimal points.

Performance on Easy PS Problems

⇒ Seeding optimal individuals improves performance significantly for easy PS problems.

Overall, NSGA-II performed better than MOEA/D for most test problems, with the only exception being the three-objective UF 10 and DTLZ 2 problems. For easy PS problems, both algorithms show a large performance benefit when seeding, regardless of seed location. Only the DTLZ 2 problem shows little improvement with the seeds, as the unseeded population r already approximates the true PF quite well, leaving little room for improvement. This can also be seen in the IGD+ values. These results for the ZDT and DTLZ problems were overall expected, as similar effects were obtained by Gong et al. [122]. While we find certain seed positions are especially advantageous for the search, the overall performance improvements are clearly visible for all seed locations. Performance benefits for problems with complicated PSs are by far not as strong.

Performance on Complicated PS Problems

⇒ Seeding optimal individuals also improves performance for complex PS problems, but results are more dependent on seed locations, and the benefit is overall much lower.

For further comparison, the percental difference between the seeded populations and the random population r is shown in Figure 11.1 for the two-objective problems, and Figure 11.2 for the three-objective problems. Here, we additionally observe two phenomena in problems with complicated PS: 1) They show less improvement in terms of IGD+ than problems with simple PS, and 2) we do not see the same coherent performance improvements regardless of seed location. For the UF 1 and UF 2 problems in Figure 11.1, we even observe the inclusion of optimal seeds to slightly decrease the performance in the statistically significant cases. It seems both MOEA/D and NSGA-II cannot find the whole PF after local convergence towards the seeded optimal solution(s), which can also be observed in visualizations of the PF in the later Section 11.3.4. For the MACO problems in Figure 11.1, we observe improved performance, especially for MOEA/D. The performance of NSGA-II did not significantly change with $e1$ as the seed. One probable reason might be the comparably easy f_1 objective in the MACO problem. However, we see the c seed for NSGA-II in the MACO base version to decrease performance, but for the other (similar) problem variations to increase performance again. Further to note is that for the MACO $w = steep$ using MOEA/D, the single seeds seem to also improve performance more than the combined seeds.

Table 11.1: Average HV and IGD+ for two-objective problems[1] [120].

		average IGD+									average HV								
		MACO b	MACO p=-10	MACO w=steep	UF 1	UF 2	UF 3	ZDT 1	ZDT 2	ZDT 3	MACO b	MACO p=-10	MACO w=steep	UF 1	UF 2	UF 3	ZDT 1	ZDT 2	ZDT 3
NSGA-II	r	0.122	0.092	0.212	0.081	0.037	0.147	0.093	0.19	0.054	0.008	0.008	0.008	14.129	1.721	109.072	5.48	4.976	4.535
	e1	0.124	0.094	0.206	0.09*	0.035	0.136*	0.003*	0.015*	0.007*	0.009*	0.008*	0.008*	14.362*	1.719	113.066*	5.685*	5.301*	4.708*
	e2	0.09*	0.065*	0.136*	0.096*	0.043*	0.127*	0.003*	0.009*	0.004*	0.009*	0.008*	0.008*	14.371*	1.75*	110.43*	5.684*	5.303*	4.709*
	c	0.138*	0.086*	0.165*	0.084	0.038	0.109*	0.004*	0.005*	0.024*	0.008	0.008	0.008*	14.152	1.718	110.066*	5.681*	5.31*	4.681*
	e1+e2	0.09*	0.065*	0.137*	0.113*	0.037	0.134*	0.003*	0.009*	0.004*	0.009*	0.008*	0.008*	14.385*	1.76*	114.009*	5.685*	5.303*	4.71*
	e1+e2+c	0.084*	0.063*	0.1*	0.103*	0.041*	0.104*	0.003*	0.004*	0.004*	0.009*	0.008*	0.008*	14.402*	1.754*	114.07*	5.685*	5.314*	4.71*
MOEA/D	r	0.31	0.286	0.456	0.142	0.077	0.222	0.868	1.492	0.548	0.007	0.007	0.008	13.182	1.55	106.591	4.365	3.719	3.101
	e1	0.366*	0.322*	0.461	0.147	0.081	0.178*	0.306*	0.319*	0.301*	0.007*	0.007*	0.008	13.287	1.547	107.567*	5.147*	4.96*	3.639*
	e2	0.144*	0.126*	0.233*	0.164*	0.07	0.257	0.074*	0.105*	0.091*	0.008*	0.008*	0.008*	13.699*	1.669*	108.14*	5.246*	4.847*	3.778*
	c	0.162*	0.158*	0.185*	0.162*	0.087	0.177*	0.145*	0.133*	0.172*	0.008*	0.008*	0.008*	12.829*	1.507	107.266*	5.188*	4.949*	3.721*
	e1+e2	0.155*	0.166*	0.44	0.153	0.071	0.242	0.079*	0.102*	0.093*	0.008*	0.007*	0.008	13.944*	1.696*	108.693*	5.241*	4.969*	3.815*
	e1+e2+c	0.099*	0.147*	0.346*	0.155	0.069	0.167*	0.068*	0.09*	0.095*	0.009*	0.008*	0.008*	13.898*	1.698*	110.242*	5.262*	4.93*	3.823*

Table 11.2: Average HV and IGD+ for three-objective problems[1] [120].

		average IGD+						average HV					
		UF8	UF9	UF10	DTLZ1	DTLZ2	DTLZ3	UF8	UF9	UF10	DTLZ1	DTLZ2	DTLZ3
NSGA-II	r	0.139	0.209	0.841	0.924	0.035	9.213	1233.913	1278.562	17789.246	4608.46	0.611	803343.736
	e1	0.146	0.187*	0.358*	0.04*	0.037*	0.04*	1237.505*	1282.456*	18007.444*	4610.158*	0.606*	804549.879*
	e2	0.101*	0.155*	0.454*	0.042*	0.036	0.038*	1248.675*	1295.8*	18648.131*	4610.159*	0.609	804554.587*
	e3	0.144	0.176*	0.164*	0.041*	0.036	0.04*	1234.679	1294.336*	18725.208*	4610.159*	0.607*	804554.482*
	c	0.134	0.155*	0.462*	0.042*	0.035	0.037*	1232.533	1277.184	17628.176*	4610.151*	0.612	804551.863*
	e1+e2+e3	0.087*	0.162*	0.093*	0.04*	0.035	0.038*	1248.9*	1297.118*	18754.865*	4610.159*	0.61	804554.601*
	e1+e2+e3+c	0.078*	0.145*	0.085*	0.04*	0.035	0.037*	1248.926*	1297.198*	18754.894*	4610.159*	0.611	804554.606*
MOEA/D	r	0.319	0.299	0.653	6.051	0.031	45.87	1135.9	1191.73	17380.906	4326.701	0.621	605899.488
	e1	0.299	0.289	0.37*	0.106*	0.028*	0.321*	1170.327*	1207.486*	17879.772*	4588.092*	0.623	799354.696*
	e2	0.299	0.291	0.591	0.057*	0.031	0.228*	1195.244*	1214.662*	18284.499*	4600.429*	0.621	801062.926*
	e3	0.33	0.338	0.493*	0.063*	0.031	0.17*	1133.902	1212.296*	18044.345*	4595.132*	0.62	802189.745*
	c	0.344	0.248*	0.47*	0.065*	0.031	0.246*	1107.64*	1210.86*	17538.075*	4587.764*	0.621	798540.134*
	e1+e2+e3	0.21*	0.267*	0.219*	0.04*	0.028*	0.1*	1215.402*	1253.464*	18581.707*	4607.648*	0.624	803839.858*
	e1+e2+e3+c	0.199*	0.231*	0.221*	0.036*	0.028*	0.106*	1213.855*	1259.83*	18461.718*	4608.439*	0.623	803545.744*

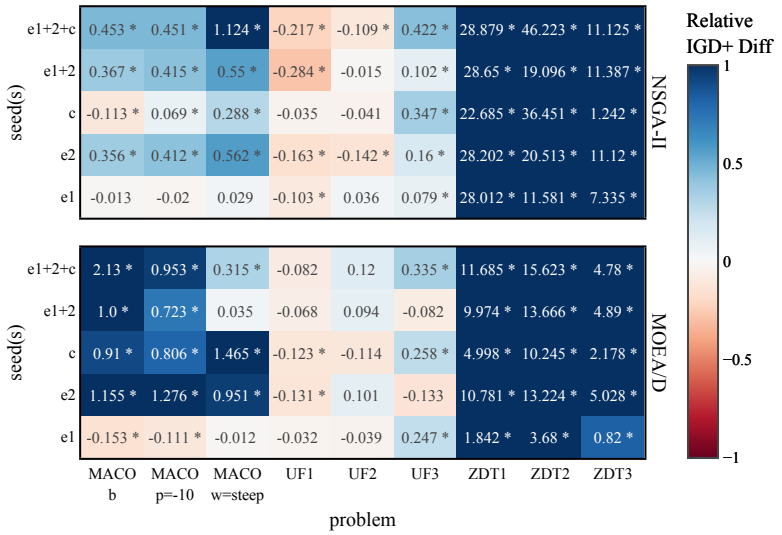


Figure 11.1: Difference in IGD+ compared to the random seed type *r* for the two-objective problems in percentage[12] [120].

Single Seed Locations vs. Multiple Seeds

⇒ Little difference is found.

Also interesting to note is that, while the combined seeds usually do perform the best, they rarely outperform a single optimal seed by much. This might be expected for the easy PS problems, where the genome largely converges towards the same value. However, for complicated PS problems, one could expect a different behavior.

Summary of the Performance Findings

For problems with easy PSs, seeding optimal individuals improves performance considerably. For problems with complicated PSs, there is no coherent result about seeding in problems with complicated PSs. We find seeding to be beneficial in most cases, but in some cases, the performance degrades. Occasionally, a single seed location leads to worse performance, while other times the seed improves it. Both NSGA-II and MOEA/D show similar performance changes, but we also find cases where one algorithm performs better. This makes it difficult to predict performance outcomes for real-world problems, especially as the optimal solutions used for seeding can only be approximated. To further explore these results, we evaluate the heritage information and have a deeper look into the search dynamics of MOEAs.

11.3.2 Population Dynamics

Evaluation of the population dynamics is split into two parts. First, the impact on the final solutions found by the algorithms is discussed, followed by a visualization of the impact of the seeds across the PF.

Impact of the Seed(s)

In this section, we evaluate the population dynamics in terms of the I_C metric, which measures the amount of the genetic material from the seed individuals in the final population. The influence of the different seeds for the two-objective problems are visualized in Figure 11.3. Each column shows the accumulated results of the test 31 test runs for each problem, with NSGA-II (top) and MOEA/D (bottom).

[1] Statistically significant differences compared to the random population *r* are marked with a *.

[2] For comparability, the color scale stops at 1.0 (100 %).

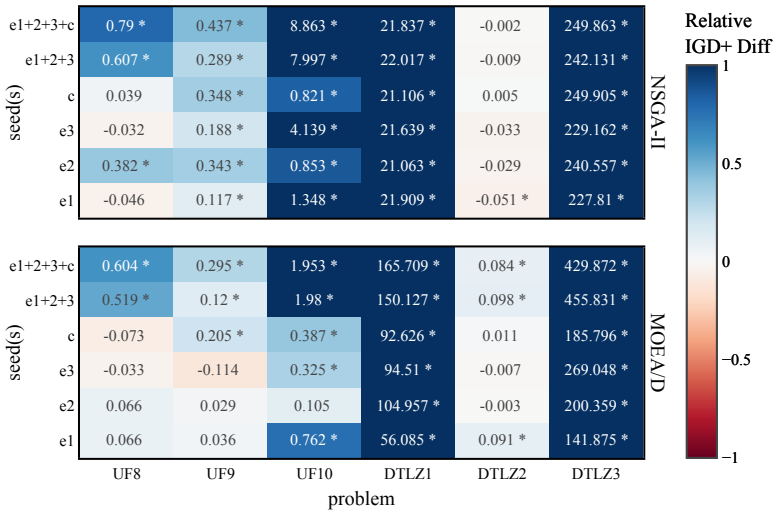


Figure 11.2: Difference in IGD+ compared to the random seed type r for the three-objective problems in percentage[12] [120].

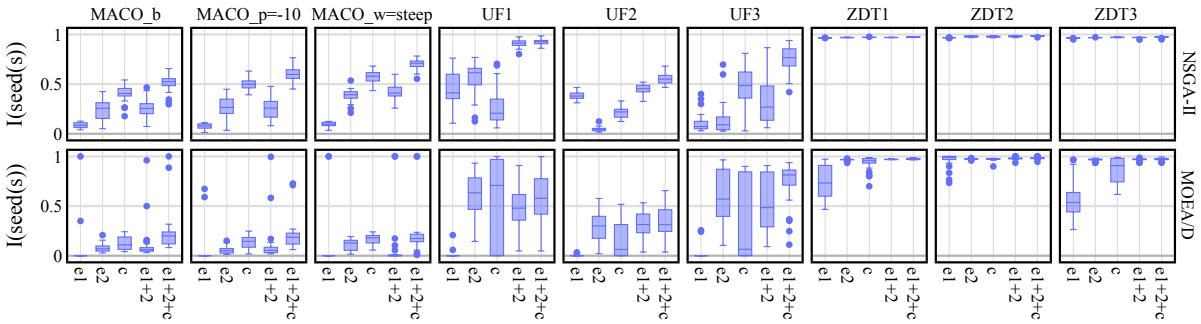


Figure 11.3: Counting impact of the different seed locations for two-objective problems [120].

For the ZDT problem, both NSGA-II and MOEA/D show that all seed types have ≤ 95 % of influence, meaning the vast majority of the genetic material comes from the seed(s). This clearly links the performance benefits to the seed individual, strongly supporting the theory of why simple PSs benefit more from seeding.

Impact on Easy PS Problems

⇒ Seeds have a huge influence in easy PS problems.

Seed influence for MOEA/D is slightly lower. One reason for this could be the mating restrictions between the reference directories, making it harder for the algorithm to utilise the optimal genetic material. Additionally, the different survival strategy of MOEA/D does probably not allow the population to converge towards the seeded optimal solution as the dominance-based survival of NSGA-II. The $e1$ seed shows considerably less influence for ZDT 1 and ZDT 3. This correlates with a lower performance improvement, as can be seen when comparing to Figure 11.1. We assume the reason for this to be a combination of mating restrictions and survival on the reference directories of MOEA/D and the fairly easy f_1 objective of the ZDT problems. The same effect for the simpler PSs of the DTLZ problems can be observed in Figure 11.4. NSGA-II again produces a high seed influence, and MOEA/D typically slightly less. We even find this to be true for DTLZ 2, even though the algorithms did not show large performance differences when seeding (Figure 11.2). However, we find that the population dynamics were still drastically changed by the seeds. Only for the $e1$ and $e2$ seeds of MOEA/D we do not see much influence. One reason could be the already good initial population, with

NSGA-II vs. MOEA/D on Easy PS problems

⇒ NSGA-II seems to exploit optimal genetic material more than MOEA/D.

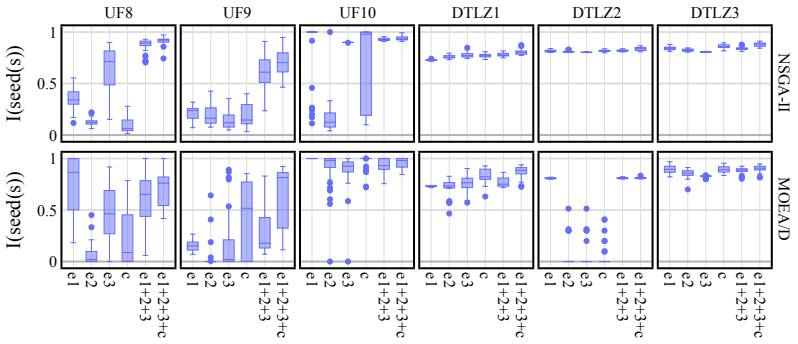


Figure 11.4: Counting impact of the different seed locations for three-objective problems [120].

the seeds not being selected for the reference directories, subsequently not surviving the first few generations.

Impact on Complicated PS Problems

⇒ Seeds have a large, but far less I_C for complicated PS problems.

For problems with complicated PS, we observe both less performance benefit when seeding optimal individuals and a lower influence from the seed. For the UF problems in Figure 11.3 and Figure 11.4, we observe this trend of an overall lower influence compared to ZDT or DTLZ. This is due to the fact that the algorithm must find a wide range of values for all genes. While we do not suggest a higher impact from the seed to equal a better performance, we can still create a link between performance and seed influence. For example, for the problems UF 1 and UF 2, we observe the $e1$ and $e2$ seeds to degrade the performance of NSGA-II (Figure 11.1), while also showing larger seed influence in Figure 11.3. For these problems, the algorithm converges fast to local points on the PF, but is not able to cover the full PF as well as a completely random initial population. The seed can therefore be described as deceptive, even though it is Pareto-optimal. For other problems, like the ZDT or DTLZ, the seeds can directly be linked to a better performance, as they heavily help the algorithm to find the true PF. These effects are also valid for the MACO problem. Interestingly, all variations of the problem exhibit a similar influence of the seed individual, despite the problems differing in the interaction between variables in the $p = -10$ variant, and the multi-modality in the $w = steep$ variants. While we can see the differences in problem difficulty in the performance results, the population dynamics in terms of impact of the seed individual do not seem to be different.

I_C of Different Seed Locations

⇒ Different seed locations have different impact (for complicated PS problems), but it differs between problems. There is no single location which has the largest influence.

The results additionally indicate that there are larger variations in the amount of influence for different seed locations. One could assume that the center seed would be more beneficial than the extreme seeds because it is closest to most of the PF, but we cannot see a general pattern in the UF problem family. Results are even different between NSGA-II and MOEA/D. For example, in the UF 8 problem, NSGA-II shows promising results with the c seed and the combined seeds, while for MOEA/D the $e1$ seed has a large influence. The high influence values of the UF 8–10 problems have to be viewed in context. As can be seen in the performance values (Table 11.2), the algorithm was unable to approximate the PF for these problems, usually converging locally. Visualizations of the NDS found for all problems can be found in the supplementary material of the publication [175]. This high influence in the three-objective UF problems does indicate the seed individual was exploited to converge towards true PF. However, we assume that algorithm convergence across the whole PF

would reduce the seed influence again, more similar to the UF 1–3 and MACO results. For the MACO problems (Figure 11.3), we find $e1$ to have the lowest impact on the final result, with $e2$ having the second largest and c the largest. The combination of seeds always produces similar influence values to the otherwise most influential single seed location, with $e1 + e2$ being similar to $e2$ and $e1 + e2 + c$ showing a slightly higher influence than c.

11.3.3 Impact of the Single Seeds Across the PF

In addition to the impact of the seed individual, we also want to evaluate where the seeds have impacted the found NDS, and evaluate differences between the seed locations. Here, the focus lies on the influence of the singular seed locations ($e1$, $e2$, $e3$, and c). Overall, we find similar trends for the test problems with easy and complicated PSs. For this reason, we chose two test problems as examples to be discussed here. Additional visualizations can be found in [175].

Figure 11.5 shows the NDS found for all 31 repeats of the DTLZ 3 problem, as an example for easy PS problems. The brightness of each solution represents the amount of I_C the seed has on this solution, or in practical terms, how much of the genetic material from the solution comes from the individual seeded. As already seen in the previous section, we find the whole PF to be largely influenced by the seed, regardless of the seed location. One can also see that NSGA-II does approximate the true PF better, as was apparent in the performance indicators (Table 11.2). However, the seeds only show a marginally larger influence in their respective locations. This is somewhat unexpected, as one could assume that solutions found around the $e1$ seed can be created with more genetic material of the seed individual. For NSGA-II, we find the $e1$ and c seeds to have a bit more influence in their respective areas. For MOEA/D we find the seeded parts of the PF to be sampled more densely. Again, the $e1$ and c seeds have a bit more influence, but the effect is overall small. One reason for this could be the very large impact on all NDS found by the algorithm.

Seed Location Impact for Easy PS Problems

⇒ Seeds influence the whole PF massively. Only in some cases we find a bit more impact around the respective seed location.

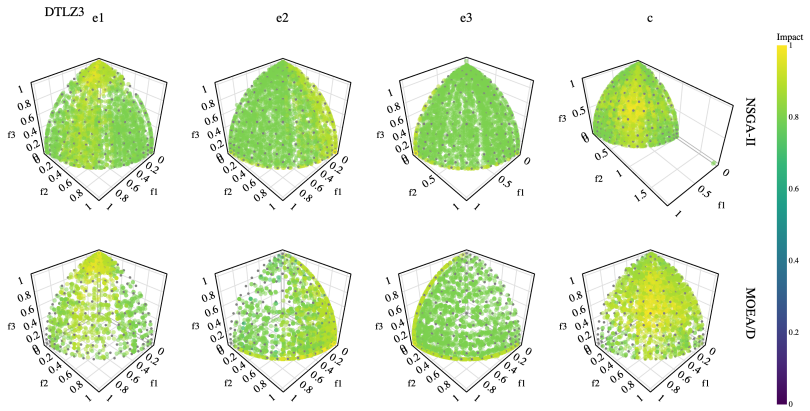


Figure 11.5: Seed Impact on the PF for DTLZ 3 problem. The true PF is shown with a grey line, while the green dots represent the accumulated NDS for the 31 repeats using the respective seed individual [120].

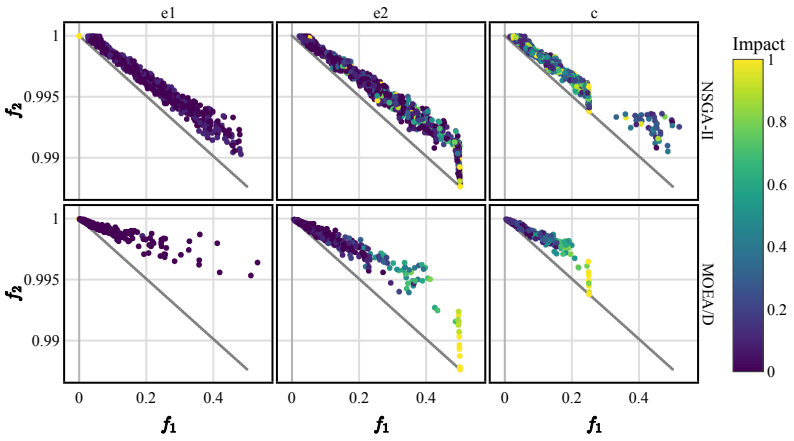


Figure 11.6: Seed Impact on the PF for MACO_b. The true PF is shown with a grey line, while the green dots represent the accumulated NDS for the 31 repeats using the respective seed individual [120].

Seed Location Impact for Complicated PS Problems

⇒ MOEA/D and NSGA-II show differences in their convergence behavior.
⇒ For MOEA/D, we find the seeds to have a larger influence in their respective locations.
⇒ For NSGA-II, seed influence is more scattered across the PF.

The seed influence for the problems with complicated PSs presents a different convergence behavior. As an example, the PFs of the base version of the MACO problem is visualized in Figure 11.6. The true PF is shown in grey, with the NDS being colored, indicating the impact of the seeded individual. This time, we can see differences between the two algorithms and the seed location. For MOEA/D, the $e2$ and c seeds influence the individuals more if they are close to their respective seed location. For NSGA-II, on the other hand, we find the seed impact to be much more scattered throughout the PF. Here, the seed locations are much less visible. We believe the mating restrictions from MOEA/D might be the reason for this difference. Interestingly, the $e1$ seed did not show any influence for other solutions (which was also already visible in Figure 11.3). Again, we believe the reason for this to be the comparatively easy f_1 objective. Comparing the performance between the algorithms, we can observe NSGA-II to approximate the true PFs better than MOEA/D. Similar to the impact evaluation of the previous section (Figure 11.3), we find the seed locations to have a different amount of influence on the result. In the example of the MACO problems, we see the most influence from the c seed. Interestingly, this larger influence does not translate to a better approximation of the true PF. For both algorithms, the $e2$ seed shows the best approximation of the PF, despite considerably less influence on the final result. Our theory for this is that, in the context of the MACO problem, the c seed is better at leading the population towards the true PF, preserving more of its genetic material in the process. However, the second objective in the MACO problem is much harder to optimize for (as can also be seen in the PFs found by the algorithm). So while it may be more difficult for the algorithm to produce a range of good solutions with $e2$, its location in the harder-to-find area of the PF may allow the algorithm to explore this area better. Finally, NSGA-II again utilises the optimal seed individuals more than MOEA/D. These trends are also found for the other problems with complicated PSs, both $d_o = 2$ and $d_o = 3$, with the MACO base version serving as the visualized example.

11.3.4 Impact of Combined Seeds Across the PF

Finally, the impact of each part of the combined seeds are evaluated in this section. Again, selected test problems are chosen as exemplary cases

here. However, results across the board are still very similar, as can be seen in [175].

For the problems with a simple PS, we discuss the found effects on the ZDT 3 problem as one example (Figure 11.7). The NDS found by the algorithms, seeded with $e1 + e2 + c$, are visualized. Each column shows the impact of one seed location. We can observe that the majority of genetic material stems from the seeds. However, this time, the influence is spread over multiple seeds. For NSGA-II, we see a quite homogeneous distribution, with the $e1$ and $e2$ seeds having slightly more influence than the c seed. For MOEA/D we find a contrasting behavior. While each seed location still influences the NDS found, results for one seed location are much less homogeneous and more scattered. We assume this to be a result of mating restrictions between the reference fronts of MOEA/D, whereas in NSGA-II the positional genes in the genome get combined until their individual influence roughly equalizes. Interestingly, we again don't find the extreme seeds to influence their respective location on the PF more.

Easy PS Problems

⇒ Again, seeds do seem to have a significantly larger influence around their respective locations on the PF.
⇒ For NSGA-IIs, seeds influence the PF homogeneously compared to MOEA/D.

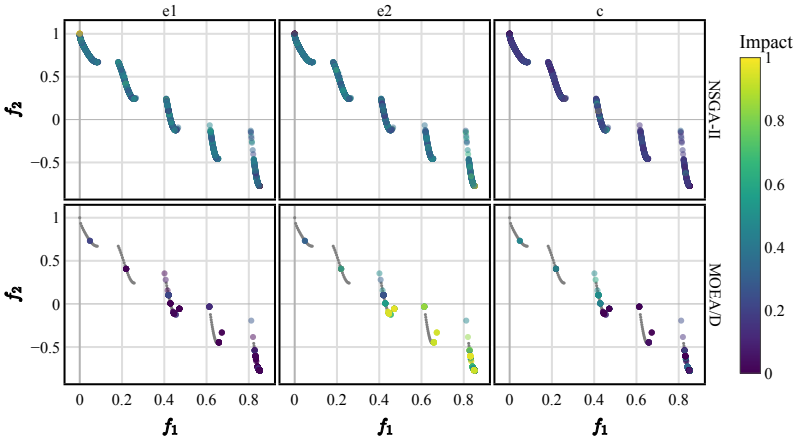


Figure 11.7: Seed impact on the NDS for each seed location of the combined $e1 + e2 + c$ configuration on ZDT 3 [120].

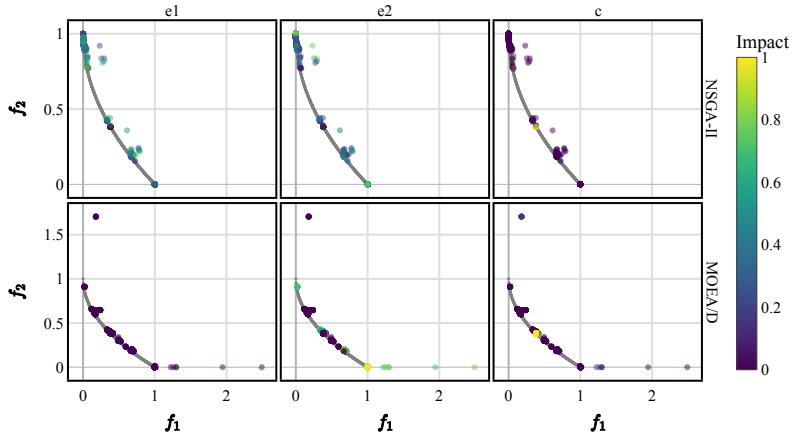


Figure 11.8: Seed impact on the NDS for each seed location of the combined $e1 + e2 + c$ configuration on UF 1 [120].

As an exemplary case for problems with complicated PSs, results for the UF 1 problem are visualized in Figure 11.8. Here, the result is also different compared to the easy PS problems and between the two algorithms. For NSGA-II, we see the $e1$ and $e2$ seed locations to have a bit more influence than the c seed. Similar to the single seed locations on complicated PS problems, we find the influence of the seeds to be more scattered

Complicated PS Problems

⇒ Similar effects than for the single seed locations in complicated PS problems are found.

throughout the PF, and the respective seed position not really being visible. We also see that MOEA/D uses the optimal genetic material much less than NSGA-II. Only for the $e2$ seed we find some influence, this time also scattered throughout the PF. Interestingly, this time, we do not find the seed locations to influence their local neighborhood more, as was visible for the single seed locations in Figure 11.6. The most probable cause for these differences is the fitness landscape of the problem.

11.4 Summary

Chapter Overview

In this chapter, we aim to provide a better understanding of which individuals from the initial population are influential in the search process of MOEAs. For this, we seeded the initial population with various known optimal solutions, and/or their combinations, tracing and evaluating their heritage data using the T-EA. Computational tests were performed using NSGA-II [30] and MOEA/D [31] on nine two-objective and six three-objective benchmark problems.

Main Findings

We want to highlight three main findings of the experiments: First, we can identify differences in the search behavior of NSGA-II and MOEA/D. The effects of different survival strategies and mating restrictions across the reference directories of MOEA/D result in less use of the optimal genetic material in the search process compared to NSGA-II. Second, we do find problems with easy PSs (ZDT, DTLZ) to benefit much more from seeding, with problems with complex PSs (UF, MACO), showing far less performance benefits, in some cases even performance decreases. Heritage data from the T-EA confirmed the seeded individuals to be the reason for this massive difference. This again highlights the criticism on these test problems [40, 44, 45, 49], especially for research on population initialization in MOEAs, which is largely conducted on benchmarks with simple PSs. Third, we could show the differences and similarities in convergence behavior for the different seed locations and algorithms. For easy PS problems, these differences were small, as the optimal genetic material of the seed gets exploited by both algorithms. For complicated PS problems, we see differences. Here, we find cases where the seeds influence their local neighborhood more and cases where the seed influence gets more scattered across the PF. The most probable reasons for this are the differences in mating restrictions between the algorithms and also the fitness landscapes of the used test problems. Furthermore, we again saw the difficulty in researching influential individuals in MOEAs compared to single-objective algorithms. We often find some interesting effects which are difficult to fully explore, as the comparison of the quality of an individual remains difficult.

Outlook and Future Work

We believe it is important to create a proven base of knowledge around the convergence behavior of MOEAs beyond assumptions and intuition. The search behavior of these algorithms is very complex, as it is a non-deterministic process dependent on the algorithm itself, its operators or the multidimensional fitness landscapes, among others. While we can see overall trends in the evaluation, we also frequently find cases where future work with more focused tests and evaluation is necessary. It, for example, remains unclear why for problems with complicated PSs, certain Pareto-optimal seed individuals improve performance, while

others can even be hurtful. It will also be interesting to investigate more the differences between the algorithms used. While our results allow for some assumptions on this, a more detailed evaluation is required. In general, testing a wider variety of problems and using different approaches to generate seed individuals resembling more realistic real-world scenarios might create new and more robust findings. Another open challenge is to figure out how to define a "good" genome for a MOP. This would allow much more focused tests, as were presented in Chapter 8.

Conclusion 12

This final chapter concludes this thesis by summarizing all contributions over the past years in terms of the research questions posed at the start. Finally, an outlook on the possible future research directions in the area of population dynamics research in EAs closes this thesis.

12.1 Summary of Contributions **125**
12.2 Outlook and Future Work **130**

12.1 Summary of Contributions

In this dissertation, the population dynamics of EAs have been evaluated on the gene level. First, an overview of related works and approaches to evaluate the search dynamic of EAs was given. Here we find a gap for gene-level approaches. Second, the traceable evolutionary algorithm (T-EA) was presented as a novel technique to track genetic heritage and influence from individuals of the initial population throughout the evolutionary process. Third, a system was proposed to compute individuals of specified properties to effectively conduct tests on the search dynamics using the T-EA. Fourth and finally, we use our tools to gain knowledge about the search process of EAs, by identifying individuals which are influential in the search process. This was done separately for single- and multi-objective EAs. These four contributions were formulated as four research goals at the beginning of this thesis in Section 1.2. From these goals, five research questions were derived. In the following, we want to discuss the findings to these in detail.

RQ 1 Which techniques exist to evaluate the search dynamics on EAs?

- RQ 1.1 On which factors can we classify these approaches?
- RQ 1.2 What knowledge about the search dynamics of EAs has been gained so far?

One current difficulty for research in the area of population dynamics is the lack of an overview in the related work, which posed the first research question of this thesis. This discussion of the related work was split into three parts. First, Chapter 3 provides an overview of the existing approaches to evaluate the population dynamics of EAs. Here, we also introduced a classification into three levels of population dynamics: First, the population level, second, the individual level, and third, the gene level. Each level looks at different details of the population, illuminating different aspects of the search process. Reviewing the related works, we found a considerable number of existing approaches developed over the past few decades. An overview of these can be found in Table 3.2. The population level is the currently most active area of population dynamics, as the recently developed STNs [71] are available as an easy-to-use web application with STNWeb [83]. In the individual level, we find genealogical evaluations for EAs, which are similar to their biological counterpart. While these approaches offer insightful information into the search process, they also suffer from the credit assignment problem due to information loss in the heritage tracking, which is done on a per-individual basis. These issues can be alleviated at the gene level. However,

we only find two gene-level approaches so far [13, 103], both for GP. Due to the inherent differences in the search dynamics of EAs for SOPs and MOPs, the discussion of the findings for population dynamics was also split to Chapter 6 for SOEAs, and Chapter 9 for MOEAs. A literature overview can be found in Table 6.1 and Table 9.1, respectively. We find a considerably smaller number of works to actually apply the proposed tools to evaluate the population dynamics, especially for MOEAs.

RQ 2 How can we precisely track genetic heritage without information loss and precise credit assignment?

- RQ 2.1 How can we trace heritage in EAs?
- RQ 2.2 How to adapt the approach to all typical genome representations and operators of genetic algorithms?
- RQ 2.3 How can we apply this to multi-objective optimization?
- RQ 2.4 What methods can be used to calculate genetic influence for the wide variety of operators and algorithms?

The second research goal is addressed in Chapter 4, namely how to track heritage throughout the evolutionary process without loss of information. While gene-level approaches exist [13, 103], they are both aimed at GPs. As operators for EAs frequently combine two genes into a new one, a new method for precisely tracking gene heritage for these algorithms was needed. For this reason, the T-EA is proposed, which is specifically designed to track genetic heritage for each gene throughout the evolutionary process of an EA. Two implementations of the T-EA have been presented. First, the trace-vector implementation (Section 4.2), which tracks genetic heritage with a single traceID. This implementation is less resource-intensive in terms of memory and computational overhead, but limited in the choice of supported operators, as only one parent per gene can be tracked. To allow the use of all common crossover and mutation operators for EAs, the *trace-list implementation* was developed. The single traceID was replaced with a trace-list, which can retain the heritage information of multiple parents, at the cost of an increased resource need. In addition to the T-EA, metrics to evaluate the heritage data were proposed. Namely, the *contributors* as the individuals from the initial population, of which genetic material is found in the current generation. And second, the four impact metrics: The I_C, I_F, I_E, and $I_{F \cdot E}$. With a few exceptions to some impact metrics, all of these methods can be directly applied to SOEAs and MOEAs.

RQ 3 How can we create individuals of specific properties to test their influence on the evolutionary process?

- RQ 3.1 How can we design a system to create individuals of desired fitness and genome targets for SOEAs?
- RQ 3.2 Can we create similar individuals for MOEAs?

Our experiments on the population dynamics of EAs required generating seed individuals of specific properties. This was addressed in Chapter 5. Again, due to the differences, we need to differentiate between individuals for SOPs and individuals for MOPs. Some easier SOPs, like the Sphere function, can be solved mathematically. This allows for calculating the needed genome, given some specifications for the genome shape. For problems that cannot be solved mathematically, the generation of the individuals was formulated as an optimization problem itself. This

methodology allowed for generating individuals of a specific fitness target f^{target}, a varying amount of optimal genes, and different distances to the respective optimal solution. Problems with multiple objectives have another difficulty. As MOEAs try to find a set of multiple optimal solutions instead of one singular one, the quality of a gene cannot be directly compared. Difficulties for this are discussed in more detail in Section 5.2. In short, the main difference to SOEAs is the challenge of measuring the quality of a gene value. For this reason, we simplify the task by selecting known optimal solutions, as they are the only ones for which we can say with certainty that they contain good genetic material.

RQ 4 Can we identify influential individuals in the search process of SOEAs?

RQ 4.1 How can we use the T-EA to evaluate the population dynamics of SOEAs?

RQ 4.2 How do the proposed impact metrics differ?

For the final goal of this thesis, we ask the question of which individuals are influential in the search process of EAs. The discussion for this was split into two parts. First, the results for SOEAs were discussed in Part II. Starting this, Chapter 7 provides a proof-of-concept evaluation to gain a better understanding of the collected heritage data from the T-EA. Here, we evaluated how different crossover and mutation operators use the genetic material in the initial population of a single-objective EA solving the Rastrigin [36] function. The T-EA was able to clearly show the differences between the two crossover and two mutation operators used. It could be seen that gene-combining crossover operators use more individuals from the initial population, as for each gene, multiple parents are able to survive. Similarly, the influence of the value-dependent mutation was lower than the influence of the value-independent mutation. We also find that the number of individuals contributing genetic material to the current population is decreasing in the first generations, before reaching an equilibrium. Here, we also found potentially interesting points in the evolutionary process worth exploring. One difference found in the population dynamics, compared to related works, was the high number of contributing individuals for the gene-combining crossover. This is in contrast to most related works, where genetic heritage of most of the population could be traced back to one individual only [13], or at least a smaller number of individuals [62, 63, 92, 94, 100, 103, 138]. These works seem to be more similar than what we found for gene-combining crossover. Finally, we compared the four proposed impact metrics. We do not observe large differences in the final generations, as the population had already converged, lowering the influence of the scaling factors. However, we do find more differences in the earlier generations, and point out their potential for use in the evolutionary process.

The main question of which individuals are influential in the search process of SOEAs was addressed in Chapter 8. For this, individuals with specific properties were generated. The main focus lay on two variations: First, individuals containing very good (optimal) genes, but also genes which are far off their optimal value, are compared to individuals of the same fitness, with an overall mediocre genome. And second, individuals of the same fitness and genome properties, but different distances to

the optimum in the search space. These individuals were seeded into the initial population of an EA, and their influence was tracked using the T-EA. The tests showed four major findings: First, we observed the amount of optimal genetic material in the genome to be clearly linked to an increase in performance. This could be shown through the heritage data of the T-EA. Second, the fitness of the seed individuals was also important, mainly since individuals need to be above a certain fitness threshold to have a realistic chance of surviving the first generations. Third, in contrast to the previously discussed findings for the proof of concept, we did not find major differences in the use of the optimal genetic material between the gene-copying and the gene-combining crossover operators. And fourth, we did not find seeds with a genome closer in distance to the optimum to significantly outperform individuals of the same fitness, but further from the optimum. We believe these four findings to be a valuable contribution towards better understanding the population dynamics in SOEAs. They again underline the importance of the initial population as a starting point for the evolutionary process. However, we also acknowledge that these results have to be viewed with respect to the limited test capacity, and the systematic generation of the seed individuals, which cannot directly be applied to real-world problems. A detailed description of the limitations and future work can be found in Section 8.4.

RQ 5 Can we identify influential individuals in the search process of MOEAs?

RQ 5.1 How do the population dynamics of MOEAs differ to SOEAs?
RQ 5.2 How can the T-EA be used to evaluate the population dynamics of MOEAs?

The evaluation for the search dynamics of MOEAs was done in Part III of this thesis. Again, this was started with a proof-of-concept evaluation in Chapter 10. The T-EA can directly be applied to MOPs, with the only exception being the fitness- and entropy-based impact metrics. As a first starting point, we tested the influence of the population size parameter on the search dynamics by evaluating how many individuals from the initial population contribute genetic material to the final result. Here, we found that the percentage of contributors was higher for smaller population sizes, and gradually decreased with an increase in population size. This hints at the algorithm's need for a certain amount of genetic diversity. If this diversity is met, more genetic material can be discarded. We could also observe the number of contributors to be dependent on the benchmark problem used, indicating the influence of different fitness landscapes on the search behavior. Interestingly, in the tests, the genetic material of the NDSs from the initial population did not show a higher chance of surviving the evolutionary process. Intuitively, one would think the opposite, as these individuals mark the best solutions in the current search process. This observation naturally leads to the same question asked for SOEAs already: Which individuals are influential in the search process of MOEAs?

This question was tackled in Chapter 11. Comparing single- and multi-objective EAs and discussing their differences revealed the difficulties of transferring the previous results of influential individuals in SOEAs. As MOEAs do not converge to a single $\vec{x}^{opt}$, but a PS of optimal solutions,

the definition of what a "good" gene is becomes much more difficult. Additionally, there also exist inherent differences in the benchmark problems, which Li and Zhang [41] classify in problems with *easy* and problems with *complicated* PSs. To gain a better understanding, similar tests as for the SOEAs were designed. Due to the difficulty in defining good genes, optimal individuals were chosen as the only solutions known to contain optimal genetic material. In the evaluation, we saw problems with "easy" PSs to benefit greatly from seeding just one optimal individual. The T-EA showed the algorithm to exploit the genetic material of the seed, so that most (> 95%) of the final population consists of the genetic material from just a single seeded individual. Problems with "complicated" PSs, on the other hand, showed much less performance improvements, in some rare cases even significant decreases in performance. This result might be unsurprising, due to the PS properties of the problems. However, it implies that our community might overestimate performance benefits from seeding, as most of the works in the area of population initialization, and especially seeding techniques, are tested on benchmarks of "easy" PSs. We could also show differences in the search behavior between NSGA-II and MOEA/D, as NSGA-II did exploit the seeded individuals more for almost all test problems, as it does not feature mating restrictions. Finally, for problems with "complicated" PSs, we did find differences in performance and seed impact for different seed locations. Interestingly, we frequently find cases where the seeds did not seem to impact their local neighborhood more but rather spread their influence across the whole PF. Our results contribute towards the question of influential individuals in the search process of MOEAs, and demonstrate the difficulty of it's answer, especially compared to the single-objective counterpart. However, we acknowledge that we could not fund effects to the same extent as for SOEAs. The search dynamics of MOEAs are much more complex, and require more focused evaluations of specific effects to gain a better overview of their search dynamics. Nevertheless, we believe our results to be an important contribution towards this goal.

Final Remarks

In this section, we summarized the author's contribution to the EC community during his PhD journey. In short, tools and approaches to evaluate the population dynamics in EAs on a gene level were developed, and an overview and classification of existing approaches was provided. Furthermore, these tools were used to create a first base of knowledge in this area, mainly through identifying influential individuals in the search process of EAs. For SOEAs, specific properties could be tested and identified. For MOEAs, this question proved more difficult, as it is much harder to even compare the quality of two genes. Maybe here lies one reason why so few works on the population dynamics in EAs exist: Because it is hard. It takes a large amount of effort to confirm even seemingly trivial behaviors. However, we strongly believe that work in this area is worth it. In our opinion, replacing assumptions and educated guesses with a proven base of knowledge on *how* and *why* EAs reach their results can only be beneficial, both for the application and development of our algorithms.

12.2 Outlook and Future Work

The field of evaluating the population dynamics of EAs, although not novel, is still in an early stage. While this thesis contributes through the development of new tools and findings around the population dynamics on the gene level, we also acknowledge that there is a wide area of questions and potential for future research directions. Here, we present eight possible next topics we deem to be of high interest for future works.

Building Foundational Knowledge

We believe one of the most important things in the area of population dynamics is to start building a solid base of understanding. While this work provides important results towards this, we also frequently find effects and behaviors which we cannot explain and do not fully understand so far. Additionally, we believe the field would benefit greatly from review works, creating an overview and connecting the current knowledge of all three levels of population dynamics, as well as creating a shared terminology around this topic.

Problems of Different Properties

This work mainly focused on unconstrained and continuous real-valued problems. It will also be interesting to explore problems of different properties, which might affect the search dynamics of EAs. This includes multi-modal problems, where there exist multiple solutions in the search space that have the same optimal fitness. Or constrained problems, which feature infeasible regions in the search space, which also potentially influence the search dynamics of EAs. Both of these areas are especially relevant for real-world applications. This thesis briefly touched on multi-modality when discussing the data from the MACO benchmark. However, more detailed works in this area are necessary. Dynamic fitness functions, which change over the evolutionary process, would also be of interest.

Heritage Data in Algorithmic Design

This work also does not explore the possibilities of using the heritage information of the T-EA in the search process. A few previous works explore these ideas, like lineage selection [107], or triggering resampling using genealogical data [105]. However, this overall remains a quite unexplored area of research which may hold interesting possibilities for future algorithmic designs. We believe the use of the fitness- and entropy-scaled impact metrics hold potential here.

Phenotype Space

In this thesis, we classify the search behavior of EAs into two categories focused on the search space, and focused on the objective space. While this is sufficient for the purposes of our work, some real-world applications also feature additional spaces worth exploring. The phenotype space is probably the most relevant candidate. With phenotype, we mean to describe the real-world representation of a solution. In path-planning applications, for example, a solution might be encoded as a series of waypoints. The resulting path then is the phenotype. In this application, it might be desirable to increase diversity in the phenotype space instead of the objective space [8]. STNs have already been used to evaluate the trajectory of a population in the phenotype space [81]. We believe this to be an interesting future area of exploration, especially for more application-focused evaluations of the search dynamics.

Significant Events during the Evolutionary Process

The T-EA and this thesis is focused on the initial population and how its genetic material is combined throughout the evolutionary process. To overcome the issue of the large amount of data generated in the evolutionary process, the algorithm only focuses on the genes created in the population initialization. However, some works point out significant events in the evolutionary process, or individuals from which the majority of the later population is derived from [62, 63, 94, 100, 138]. We believe a more detailed exploration on the gene level might be interesting. The evaluation of a single test run in Section 7.3.1 showed some potential for finding these interesting points. However, the main challenge for extended research in this direction is that it can only be done on one test run. Evaluating the typical number of test repeats on a scientific scale is currently infeasible.

The Challenge of Data Handling

As stated, one major challenge when evaluating the population dynamics of EAs is the large amount of data, especially when considering heritage data, throughout the evolutionary process [62]. Increases in computational storage capabilities and computational power allow us to collect ever greater amounts of information. However, our available approaches to evaluate these remain a bottleneck. Current tools all try to focus on specific areas, like the path of a population through the search space with STNs, or tracking the heritage of the initial population with the T-EA. However, even with this focus, evaluating the resulting data still requires a lot of resources and space. Within current guidelines and best practices of testing in EAs [27] it is unrealistic to fully explore the generated data within the boundaries of a scientific publication. One area of future work has to be focused on how to better visualize and explore this large amount of data to better evaluate specific details in the evolutionary process. One interesting area could be the recent advances in LLM. Using them as a tool for data evaluation has, for example, been done by Chacón Sartori, Blum, and Ochoa [86] for STNs.

Real-World Applications

In this thesis, experiments are done using unconstrained benchmark functions, due to their known properties, known PSs and PFs, availability and fast computation times. The results gained in this thesis need to be seen in this context. Real-world problems might feature different challenges for the algorithms. Additionally, as mentioned above, we do not consider constraint- or multi-modal problems here, as they would introduce an additional layer of complexity for the search behavior of EAs. It could already be observed that the fitness landscape of the problem influences the population dynamics, which will be no different for real-world applications. We believe it will be an interesting area of future research to explore using the T-EA on real-world problems, as it also holds the potential to give additional data to decision makers and domain experts. An example of this could be seeding known solutions into the initial population of an EA, and evaluating which of its features survived the evolutionary process, which were combined differently, and which were discarded. We believe this to be an interesting area of future research.

Limitations of the T-EA

The T-EA can be applied to the most common representations and operators used in EAs. However, applying them to variably sized genomes, or permutation-based combinatorial problems, might require some adjustment, as information about the original position of the gene in the genome would be lost. Furthermore, this thesis, and the T-EA, is focused

on a specific type of algorithm. Applying the T-EAs to other types of EAs is generally possible. However, it might need some additional adjustment. For GP, for example, the importance of the root parent is often argued [13, 92], which is implemented in the gene-tracing approach of McPhee and Hopper [13]. Adjusting the T-EA for the specific requirements for each algorithm is important to allow for a wider application and comparison between the population dynamics for all types of EAs.

Bibliography

[1] Lukas Bostelmann-Arp, Sanaz Mostaghim, Andreas Braun, and Thomas Tüting. 'Multi-Objective Evolutionary Game Theory: A Case Study in Cancer Therapy'. In: MIT Press, July 2022. DOI: 10.1162/isal_a_00498.

[2] Jessica Mégane, Nuno Lourenço, J. Ignacio Hidalgo, and Penousal Machado. 'Contribution of Probabilistic Structured Grammatical Evolution to Efficient Exploration of the Search Space. A Case Study in Glucose Prediction'. In: *Proceedings of the Genetic and Evolutionary Computation Conference*. New York, NY, USA: Association for Computing Machinery, July 2025, pp. 1433–1442.

[3] Fernando Lezama, Jose Almeida, Joao Soares, and Zita Vale. 'Explainergy: Towards Explainability of Metaheuristic Performance in the Energy Field'. In: *2023 IEEE Symposium Series on Computational Intelligence (SSCI)*. Dec. 2023, pp. 783–788. DOI: 10.1109/SSCI52147.2023.10371844.

[4] Thomas Seidelmann and Sanaz Mostaghim. 'Finding Cost-Effective Re-Layouting Solutions in Modern Brownfield Facility Layout Planning'. In: *2022 IEEE Congress on Evolutionary Computation (CEC)*. July 2022, pp. 1–8. DOI: 10.1109/CEC55065.2022.9870345.

[5] Tobias Benecke, Oliver Antons, Sanaz Mostaghim, and Julia Arlinghaus. 'A Generalized Circular Supply Chain Problem for Multi-objective Evolutionary Algorithms'. In: *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*. GECCO '23 Companion. New York, NY, USA: Association for Computing Machinery, July 2023, pp. 355–358. DOI: 10.1145/3583133.3590742.

[6] Tobias Benecke, Oliver Antons, Sanaz Mostaghim, and Julia Arlinghaus. 'A Coevolution Approach for the Multi-objective Circular Supply Chain Problem'. In: *2023 IEEE Conference on Artificial Intelligence (CAI)*. June 2023, pp. 222–223. DOI: 10.1109/CAI54212.2023.00103.

[7] Lukas Bostelmann-Arp, Christoph Steup, and Sanaz Mostaghim. 'Multi-Objective Seed Curve Optimization for Coverage Path Planning in Precision Farming'. In: *Proceedings of the Genetic and Evolutionary Computation Conference*. GECCO '23. New York, NY, USA: Association for Computing Machinery, July 2023, pp. 1312–1320. DOI: 10.1145/3583131.3590490.

[8] Jens Weise. 'Evolutionary Many-Objective Optimization for Pathfinding Problems'. In: (2023). ISBN: 9781839107955. DOI: 10.25673/101389.

[9] Lukas Bostelmann-Arp, Christoph Steup, and Sanaz Mostaghim. 'Free-Form Coverage Path Planning of Quadcopter Swarms for Search and Rescue Missions Using Multi-Objective Optimization'. In: *2024 IEEE Congress on Evolutionary Computation (CEC)*. June 2024, pp. 1–8. DOI: 10.1109/CEC60901.2024.10611984.

[10] Filip Karlo Došilović, Mario Brčić, and Nikica Hlupić. 'Explainable Artificial Intelligence: A Survey'. In: *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. May 2018, pp. 0210–0215. DOI: 10.23919/MIPRO.2018.8400040.

[11] Dang Minh, H. Xiang Wang, Y. Fen Li, and Tan N. Nguyen. 'Explainable Artificial Intelligence: A Comprehensive Review'. In: *Artificial Intelligence Review* 55.5 (June 2022), pp. 3503–3568. DOI: 10.1007/s10462-021-10088-y.

[12] J. N. Hooker. 'Testing Heuristics: We Have it All Wrong'. In: *Journal of Heuristics* 1.1 (Sept. 1995), pp. 33–42. DOI: 10.1007/BF02430364.

[13] Nicholas Freitag McPhee and Nicholas J Hopper. 'Analysis of Genetic Diversity through Population History'. In: vol. Proceedings of the Genetic and Evolutionary Computation Conference. 1999, pp. 1112–1120.

[14] James M. Whitacre, Tuan Q. Pham, and Ruhul A. Sarker. 'Credit Assignment in Adaptive Evolutionary Algorithms'. In: *Proceedings of the 8th annual Conference on Genetic and Evolutionary Computation*. GECCO '06. New York, NY, USA: Association for Computing Machinery, July 2006, pp. 1353–1360. DOI: 10.1145/1143997.1144206.

[15] James M. Whitacre. *Adaptation and Self-Organization in Evolutionary Algorithms*. July 2009. DOI: 10.48550/arXiv.0907.0516.

[16] Thomas Bäck and Hans-Paul Schwefel. 'An Overview of Evolutionary Algorithms for Parameter Optimization'. In: *Evolutionary Computation* 1.1 (Mar. 1993), pp. 1–23. DOI: 10.1162/evco.1993.1.1.1.

[17] John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. MIT Press, Apr. 1992.

[18] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA, USA: MIT Press, 1992.

[19] Ingo Rechenberg. 'Evolutionsstrategie — Optimierung technischer Systeme nach Prinzipien der biologischen Evolution'. PhD thesis. 1973. DOI: 10.1002/fedr.19750860506.

[20] David B. Fogel. 'Artificial Intelligence through Simulated Evolution'. In: *Evolutionary Computation: The Fossil Record*. IEEE, 1998, pp. 227–296. DOI: 10.1109/9780470544600.ch7.

[21] Rainer Storn and Kenneth Price. 'Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces'. In: *Journal of Global Optimization* 11.4 (Dec. 1997), pp. 341–359. DOI: 10.1023/A:1008202821328.

[22] M. D. McKay, R. J. Beckman, and W. J. Conover. 'A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code'. In: *Technometrics* 21.2 (1979), pp. 239–245. DOI: 10.2307/1268522.

[23] David E. Goldberg and Kalyanmoy Deb. 'A Comparative Analysis of Selection Schemes Used in Genetic Algorithms'. In: *Foundations of Genetic Algorithms*. Ed. by Gregory J. E. Rawlins. Vol. 1. Elsevier, Jan. 1991, pp. 69–93. DOI: 10.1016/B978-0-08-050684-5.50008-2.

[24] Gilbert Syswerda. 'Uniform Crossover in Genetic Algorithms'. In: *Proceedings of the 3rd International Conference on Genetic Algorithms*. Morgan Kaufmann Publishers Inc., Jan. 1989.

[25] K. Deb and R. Agrawal. 'Simulated Binary Crossover for Continuous Search Space'. In: *Complex Systems* 9 (1995), pp. 115–148.

[26] Kalyanmoy Deb and Samir Agrawal. 'A Niched-Penalty Approach for Constraint Handling in Genetic Algorithms'. In: *Artificial Neural Nets and Genetic Algorithms*. Ed. by Andrej Dobnikar, Nigel C. Steele, David W. Pearson, and Rudolf F. Albrecht. Vienna: Springer, 1999, pp. 235–243. DOI: 10.1007/978-3-7091-6384-9_40.

[27] Hisao Ishibuchi, Lie Meng Pang, and Ke Shang. 'Difficulties in Fair Performance Comparison of Multi-Objective Evolutionary Algorithms'. In: *IEEE Computational Intelligence Magazine* 17.1 (Feb. 2022), pp. 86–101. DOI: 10.1109/MCI.2021.3129961.

[28] Nikolaus Hansen and Andreas Ostermeier. 'Completely Derandomized Self-Adaptation in Evolution Strategies'. In: *Evol. Comput.* 9.2 (June 2001), pp. 159–195. DOI: 10.1162/106365601750190398.

[29] Eckart Zitzler and Simon Künzli. 'Indicator-Based Selection in Multiobjective Search'. In: *Parallel Problem Solving from Nature - PPSN VIII*. Ed. by Xin Yao, Edmund K. Burke, José A. Lozano, Jim Smith, Juan Julián Merelo-Guervós, John A. Bullinaria, Jonathan E. Rowe, Peter Tiňo, Ata Kabán, and Hans-Paul Schwefel. Berlin, Heidelberg: Springer, 2004, pp. 832–842. DOI: 10.1007/978-3-540-30217-9_84.

[30] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. 'A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II'. In: *IEEE Transactions on Evolutionary Computation* 6.2 (Apr. 2002), pp. 182–197. DOI: 10.1109/4235.996017.

[31] 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. DOI: 10.1109/TEVC.2007.892759.

[32] Qian Xu, Zhanqi Xu, and Tao Ma. 'A Survey of Multiobjective Evolutionary Algorithms Based on Decomposition: Variants, Challenges and Future Directions'. In: *IEEE Access* 8 (2020), pp. 41588–41614. DOI: 10.1109/ACCESS.2020.2973670.

[33] Thomas Bartz-Beielstein et al. *Benchmarking in Optimization: Best Practice and Open Issues*. July 2020. DOI: 10.48550/arXiv.2007.03488.

[34] Nikolaus Hansen, Steffen Finck, Raymond Ros, and Anne Auger. *Real-Parameter Black-Box Optimization Benchmarking 2009: Noiseless Functions Definitions*. Research Report RR-6829. INRIA, 2009.

[35] Nikolaus Hansen, Anne Auger, Raymond Ros, Olaf Mersmann, Tea Tušar, and Dimo Brockhoff. *COCO: A Platform for Comparing Continuous Optimizers in a Black-Box Setting*. Sept. 2020. DOI: 10.1080/10556788.2020.1808977.

[36] Frank Hoffmeister and Thomas Bäck. 'Genetic Algorithms and Evolution Strategies: Similarities and Differences'. In: *Parallel Problem Solving from Nature*. Ed. by Hans-Paul Schwefel and Reinhard Männer. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 1991, pp. 455–469. DOI: 10.1007/BFb0029787.

[37] H. H. Rosenbrock. 'An Automatic Method for Finding the Greatest or Least Value of a Function'. In: *The Computer Journal* 3.3 (Jan. 1960), pp. 175–184. DOI: 10.1093/comjnl/3.3.175.

[38] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. 'Comparison of Multiobjective Evolutionary Algorithms: Empirical Results'. In: *Evolutionary Computation* 8.2 (June 2000), pp. 173–195. DOI: 10.1162/106365600568202.

[39] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. 'Scalable Multi-Objective Optimization Test Problems'. In: *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*. Vol. 1. May 2002, pp. 825–830. DOI: 10.1109/CEC.2002.1007032.

[40] Tatsuya Okabe, Yaochu Jin, Markus Olhofer, and Bernhard Sendhoff. 'On Test Functions for Evolutionary Multi-objective Optimization'. In: *Parallel Problem Solving from Nature - PPSN VIII*. Ed. by Xin Yao, Edmund K. Burke, José A. Lozano, Jim Smith, Juan Julián Merelo-Guervós, John A. Bullinaria, Jonathan E. Rowe, Peter Tiňo, Ata Kabán, and Hans-Paul Schwefel. Berlin, Heidelberg: Springer, 2004, pp. 792–802. DOI: 10.1007/978-3-540-30217-9_80.

[41] Hui Li and Qingfu Zhang. 'Multiobjective Optimization Problems With Complicated Pareto Sets, MOEA/D and NSGA-II'. In: *IEEE Transactions on Evolutionary Computation* 13.2 (Apr. 2009), pp. 284–302. DOI: 10.1109/TEVC.2008.925798.

[42] Hisao Ishibuchi, Yu Setoguchi, Hiroyuki Masuda, and Yusuke Nojima. 'Performance of Decomposition-Based Many-Objective Algorithms Strongly Depends on Pareto Front Shapes'. In: *IEEE Transactions on Evolutionary Computation* 21.2 (Apr. 2017), pp. 169–190. DOI: 10.1109/TEVC.2016.2587749.

[43] Hisao Ishibuchi, Linjun He, and Ke Shang. 'Regular Pareto Front Shape is not Realistic'. In: *2019 IEEE Congress on Evolutionary Computation (CEC)*. June 2019, pp. 2034–2041. DOI: 10.1109/CEC.2019.8790342.

[44] Dimo Brockhoff, Anne Auger, Nikolaus Hansen, and Tea Tušar. 'Using Well-Understood Single-Objective Functions in Multiobjective Black-Box Optimization Test Suites'. In: *Evolutionary Computation* 30.2 (June 2022), pp. 165–193. DOI: 10.1162/evco_a_00298.

[45] Andrejaana Andova, Tobias Benecke, Harald Ludwig, and Tea Tušar. 'Towards Constructing a Suite of Multi-Objective Optimization Problems with Diverse Landscapes'. In: *Applications of Evolutionary Computation*. Ed. by João Correia, Stephen Smith, and Raneem Qaddoura. Lecture Notes in Computer Science. Cham: Springer Nature Switzerland, 2023, pp. 442–457. DOI: 10.1007/978-3-031-30229-9_29.

[46] Simon Huband, Luigi Barone, Lyndon While, and Philip Hingston. 'A Scalable Multi-objective Test Problem Toolkit'. In: vol. 3410. Jan. 2005, pp. 280–295. DOI: 10.1007/978-3-540-31880-4_20.

[47] S. Huband, P. Hingston, L. Barone, and L. While. 'A Review of Multiobjective Test Problems and a Scalable Test Problem Toolkit'. In: *IEEE Transactions on Evolutionary Computation* 10.5 (Oct. 2006), pp. 477–506. DOI: 10.1109/TEVC.2005.861417.

[48] Henrik Smedberg and Sunith Bandaru. 'Finding Influential Variables in Multi-Objective Optimization Problems'. In: *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. Dec. 2020, pp. 173–180. DOI: 10.1109/SSCI47803.2020.9308383.

[49] Qingfu Zhang, Aimin Zhou, Shizheng Zhao, Ponnuthurai Suganthan, Wudong Liu, and Santosh Tiwari. *Multiobjective Optimization Test Instances for the CEC 2009 Special Session and Competition*. Technical Report 264. University of Essex, Colchester, UK and Nanyang Technological University, Singapore, 2009.

[50] Sebastian Mai, Tobias Benecke, and Sanaz Mostaghim. 'MACO: A Real-World Inspired Benchmark for Multi-objective Evolutionary Algorithms'. In: *Evolutionary Multi-Criterion Optimization*. Ed. by Michael Emmerich, André Deutz, Hao Wang, Anna V. Kononova, Boris Naujoks, Ke Li, Kaisa Miettinen, and Iryna Yevseyeva. Lecture Notes in Computer Science. Cham: Springer Nature Switzerland, 2023, pp. 305–318. DOI: 10.1007/978-3-031-27250-9_22.

[51] Oliver Antons, Tobias Benecke, Sanaz Mostaghim, and Julia C. Arlinghaus. 'Sustainability in Chemical Production – Multi-Objective Distributed Control'. In: *New Trends in Intelligent Software Methodologies, Tools and Techniques*. IOS Press, 2023, pp. 211–219. DOI: 10.3233/FAIA230236.

[52] Ryoji Tanabe and Hisao Ishibuchi. 'An Easy-to-Use Real-World Multi-Objective Optimization Problem Suite'. In: *Applied Soft Computing* 89 (Apr. 2020), p. 106078. DOI: 10.1016/j.asoc.2020.106078.

[53] Pascal Kerschke and Mike Preuss. 'Exploratory Landscape Analysis'. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. GECCO '19. New York, NY, USA: Association for Computing Machinery, July 2019, pp. 1137–1155. DOI: 10.1145/3319619.3323389.

[54] James McDermott, Kostadin Dimanov Georgiev, and Miguel Nicolau. 'Evolving to Extinction: a Case Study in Heuristic Search Dynamics'. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. GECCO '25 Companion. New York, NY, USA: Association for Computing Machinery, Aug. 2025, pp. 1816–1824. DOI: 10.1145/3712255.3734329.

[55] Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Knapsack Problems*. Berlin, Heidelberg: Springer, 2004.

[56] Eckart Zitzler and Lothar Thiele. 'Multiobjective Optimization using Evolutionary Algorithms — A Comparative Case Study'. In: *Parallel Problem Solving from Nature — PPSN V*. Ed. by Agoston E. Eiben, Thomas Bäck, Marc Schoenauer, and Hans-Paul Schwefel. Berlin, Heidelberg: Springer, 1998, pp. 292–301. DOI: 10.1007/BFb0056872.

[57] Hisao Ishibuchi, Hiroyuki Masuda, Yuki Tanigaki, and Yusuke Nojima. '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. DOI: 10.1007/978-3-319-15892-1_8.

[58] Hisao Ishibuchi, Ryo Imada, Yu Setoguchi, and Yusuke Nojima. 'Reference Point Specification in Hypervolume Calculation for Fair Comparison and Efficient Search'. In: *Proceedings of the Genetic and Evolutionary Computation Conference*. GECCO '17. New York, NY, USA: Association for Computing Machinery, July 2017, pp. 585–592. DOI: 10.1145/3071178.3071264.

[59] David A. Van Veldhuizen. 'Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations'. PhD Thesis. Wright-Patterson Air Force Base, Ohio: Air Force Institute of Technology, 1999.

[60] Tea Tušar and Bogdan Filipič. 'Visualization of Pareto Front Approximations in Evolutionary Multiobjective Optimization: A Critical Review and the Prosection Method'. In: *IEEE Transactions on Evolutionary Computation* 19.2 (Apr. 2015), pp. 225–245. DOI: 10.1109/TEVC.2014.2313407.

[61] James Whitacre, Ruhul Sarker, and Tuan Pham. 'Making and Breaking Power Laws in Evolutionary Algorithm Population Dynamics'. In: *Memetic Computing* 1 (June 2009), p. 125. DOI: 10.1007/s12293-009-0009-8.

[62] Nicholas Freitag McPhee, David Donatucci, and Thomas Helmuth. 'Using Graph Databases to Explore the Dynamics of Genetic Programming Runs'. In: *Genetic Programming Theory and Practice XIII*. Ed. by Rick Riolo, W.P. Worzel, Mark Kotanchek, and Arthur Kordon. Cham: Springer International Publishing, 2016, pp. 185–201. DOI: 10.1007/978-3-319-34223-8_11.

[63] Nicholas Freitag McPhee, Maggie M. Casale, Mitchell Finzel, Thomas Helmuth, and Lee Spector. 'Visualizing Genetic Programming Ancestries using Graph Databases'. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. GECCO '17. New York, NY, USA: Association for Computing Machinery, July 2017, pp. 245–246. DOI: 10.1145/3067695.3075617.

[64] Robert Collier and Mark Wineberg. 'The Problems with Counting Ancestors in a Simple Genetic Algorithm'. In: *Advances in Artificial Life*. Ed. by Fernando Almeida e Costa, Luis Mateus Rocha, Ernesto Costa, Inman Harvey, and António Coutinho. Berlin, Heidelberg: Springer, 2007, pp. 855–864. DOI: 10.1007/978-3-540-74913-4_86.

[65] Mitchell A. Potter and Kenneth A. De Jong. 'Cooperative Coevolution: An Architecture for Evolving Coadapted Subcomponents'. In: *Evolutionary Computation* 8.1 (Mar. 2000), pp. 1–29. DOI: 10.1162/106365600568086.

[66] Andrea De Lorenzo, Eric Medvet, Tea Tušar, and Alberto Bartoli. 'An Analysis of Dimensionality Reduction Techniques for Visualizing Evolution'. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. GECCO '19. New York, NY, USA: Association for Computing Machinery, July 2019, pp. 1864–1872. DOI: 10.1145/3319619.3326868.

[67] H. Hotelling. 'Analysis of a Complex of Statistical Variables into Principal Components'. In: *Journal of Educational Psychology* 24.6 (1933), pp. 417–441. DOI: 10.1037/h0071325.

[68] Warren S. Torgerson. 'Multidimensional Scaling: I. Theory and Method'. In: *Psychometrika* 17.4 (Dec. 1952), pp. 401–419. DOI: 10.1007/BF02288916.

[69] Laurens Van der Maaten and Geoffrey Hinton. 'Visualizing Data using t-SNE.' In: *Journal of machine learning research* 9.11 (2008), pp. 2579–2605.

[70] John Healy and Leland McInnes. 'Uniform Manifold Approximation and Projection'. In: *Nature Reviews Methods Primers* 4.1 (Nov. 2024), p. 82. DOI: 10.1038/s43586-024-00363-x.

[71] Gabriela Ochoa, Katherine M. Malan, and Christian Blum. 'Search Trajectory Networks of Population-Based Algorithms in Continuous Spaces'. In: *Applications of Evolutionary Computation*. Ed. by Pedro A. Castillo, Juan Luis Jiménez Laredo, and Francisco Fernández de Vega. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 70–85. DOI: 10.1007/978-3-030-43722-0_5.

[72] Yuri Lavinas, Claus Aranha, and Gabriela Ochoa. 'Search Trajectories Networks of Multiobjective Evolutionary Algorithms'. In: *Applications of Evolutionary Computation*. Ed. by Juan Luis Jiménez Laredo, J. Ignacio Hidalgo, and Kehinde Oluwatoyin Babaagba. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2022, pp. 223–238. DOI: 10.1007/978-3-031-02462-7_15.

[73] Gabriela Ochoa, Katherine M. Malan, and Christian Blum. 'Search Trajectory Networks: A Tool for Analysing and Visualizing the Behavior of Metaheuristics'. In: *Applied Soft Computing* 109 (Sept. 2021), p. 107492. DOI: 10.1016/j.asoc.2021.107492.

[74] Valentina Narvaez-Teran, Gabriela Ochoa, and Eduardo Rodriguez-Tello. 'Search Trajectory Networks Applied to the Cyclic Bandwidth Sum Problem'. In: *IEEE Access* 9 (2021), pp. 151266–151277. DOI: 10.1109/ACCESS.2021.3126015.

[75] Nasir Saeed, Haewoon Nam, Mian Imtiaz Ul Haq, and Dost Bhatti Muhammad Saqib. 'A Survey on Multidimensional Scaling'. In: *ACM Comput. Surv.* 51.3 (2018), 47:1–47:25. DOI: 10.1145/3178155.

[76] Stefano Sarti and Gabriela Ochoa. 'A NEAT Visualization of Neuroevolution Trajectories'. In: *Applications of Evolutionary Computation*. Ed. by Pedro A. Castillo and Juan Luis Jiménez Laredo. Cham: Springer International Publishing, 2021, pp. 714–728. DOI: 10.1007/978-3-030-72699-7_45.

[77] Kenneth O. Stanley and Risto Miikkulainen. 'Evolving Neural Networks through Augmenting Topologies'. In: *Evolutionary Computation* 10.2 (June 2002), pp. 99–127. DOI: 10.1162/106365602320169811.

[78] Stefano Sarti, Jason Adair, and Gabriela Ochoa. 'Neuroevolution Trajectory Networks of the Behavior Space'. In: *Applications of Evolutionary Computation*. Ed. by Juan Luis Jiménez Laredo, J. Ignacio Hidalgo, and Kehinde Oluwatoyin Babaagba. Cham: Springer International Publishing, 2022, pp. 685–703. DOI: 10.1007/978-3-031-02462-7_43.

[79] Yuri Lavinas, Marcelo Ladeira, Gabriela Ochoa, and Claus Aranha. 'Component-Wise Analysis of Automatically Designed Multiobjective Algorithms on Constrained Problems'. In: *Proceedings of the Genetic and Evolutionary Computation Conference*. GECCO '22. New York, NY, USA: Association for Computing Machinery, July 2022, pp. 538–546. DOI: 10.1145/3512290.3528719.

[80] Gabriela Ochoa, Arnaud Liefooghe, Yuri Lavinas, and Claus Aranha. 'Decision/Objective Space Trajectory Networks for Multi-objective Combinatorial Optimization'. In: *Evolutionary Computation in Combinatorial Optimization*. Ed. by Leslie Pérez Cáceres and Thomas Stützle. Lecture Notes in Computer Science. Cham: Springer Nature Switzerland, 2023, pp. 211–226. DOI: 10.1007/978-3-031-30035-6_14.

[81] Ting Hu, Gabriela Ochoa, and Wolfgang Banzhaf. 'Phenotype Search Trajectory Networks for Linear Genetic Programming'. In: *Genetic Programming*. Ed. by Gisele Pappa, Mario Giacobini, and Zdenek Vasicek. Lecture Notes in Computer Science. Cham: Springer Nature Switzerland, 2023, pp. 52–67. DOI: 10.1007/978-3-031-29573-7_4.

[82] Camilo Chacon-Sartori, Christian Blum, and Gabriela Ochoa. 'Search Trajectory Networks Meet the Web: A Web Application for the Visual Comparison of Optimization Algorithms'. In: *Proceedings of the 2023 12th International Conference on Software and Computer Applications*. ICSCA '23. New York, NY, USA: Association for Computing Machinery, June 2023, pp. 89–96. DOI: 10.1145/3587828.3587843.

[83] Camilo Chacon-Sartori, Christian Blum, and Gabriela Ochoa. 'STNWeb: A new Visualization Tool for Analyzing Optimization Algorithms'. In: *Software Impacts* 17 (Sept. 2023), p. 100558. DOI: 10.1016/j.simpa.2023.100558.

[84] Camilo Chacón Sartori, Christian Blum, and Gabriela Ochoa. 'An Extension of STNWeb Functionality: On the Use of Hierarchical Agglomerative Clustering as an Advanced Search Space Partitioning Strategy'. In: *Proceedings of the Genetic and Evolutionary Computation Conference*. GECCO '24. New York, NY, USA: Association for Computing Machinery, July 2024, pp. 151–159. DOI: 10.1145/3638529.3654084.

[85] Camilo Chacón Sartori and Christian Blum. 'STNWeb for the Analysis of Optimization Algorithms: A Short Introduction'. In: *Metaheuristics*. Ed. by Marc Sevaux, Alexandru-Liviu Olteanu, Eduardo G. Pardo, Angelo Sifaleras, and Salma Makboul. Cham: Springer Nature Switzerland, 2024, pp. 367–372. DOI: 10.1007/978-3-031-62922-8_29.

[86] Camilo Chacón Sartori, Christian Blum, and Gabriela Ochoa. 'Large Language Models for the Automated Analysis of Optimization Algorithms'. In: *Proceedings of the Genetic and Evolutionary Computation Conference*. GECCO '24. New York, NY, USA: Association for Computing Machinery, July 2024, pp. 160–168. DOI: 10.1145/3638529.3654086.

[87] OpenAI et al. *GPT-4 Technical Report*. Mar. 2024. DOI: 10.48550/arXiv.2303.08774.

[88] Ankit Satpute, Noah Giessing, Andre Greiner-Petter, Moritz Schubotz, Olaf Teschke, Akiko Aizawa, and Bela Gipp. *Can LLMs Master Math? Investigating Large Language Models on Math Stack Exchange*. Mar. 2024. DOI: 10.48550/arXiv.2404.00344.

[89] Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. *Large Language Models for Mathematical Reasoning: Progresses and Challenges*. Sept. 2024. DOI: 10.48550/arXiv.2402.00157.

[90] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 'A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions'. In: *ACM Transactions on Information Systems* 43.2 (Jan. 2025), 42:1–42:55. DOI: 10.1145/3703155.

[91] François-Michel De Rainville, Félix-Antoine Fortin, Marc-André Gardner, Marc Parizeau, and Christian Gagné. 'DEAP: A Python framework for Evolutionary Algorithms'. In: July 2012, pp. 85–92. DOI: 10.1145/2330784.2330799.

[92] David Donatucci, M Kirbie Dramdahl, and Nicholas Freitag McPhee. 'Analysis of Genetic Programming Ancestry Using a Graph Database'. In: *Proceedings of the Midwest Instruction and Computing Symposium*. 2014.

[93] Bogdan Burlacu, Michael Affenzeller, Michael Kommenda, Stephan M. Winkler, and Gabriel Kronberger. 'Evolution Tracking in Genetic Programming'. In: *Proceedings of the European Modeling and Simulation Symposium*. Vienna, Austria, 2012.

[94] Bogdan Burlacu, Michael Affenzeller, Michael Kommenda, Stephan Winkler, and Gabriel Kronberger. 'Visualization of Genetic Lineages and Inheritance Information in Genetic Programming'. In: *Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation*. GECCO '13 Companion. New York, NY, USA: Association for Computing Machinery, July 2013, pp. 1351–1358. DOI: 10.1145/2464576.2482714.

[95] Michael Affenzeller, Stephan M. Winkler, Gabriel Kronberger, Michael Kommenda, Bogdan Burlacu, and Stefan Wagner. 'Gaining Deeper Insights in Symbolic Regression'. In: *Genetic Programming Theory and Practice XI*. Ed. by Rick Riolo, Jason H. Moore, and Mark Kotanchek. New York, NY: Springer, 2014, pp. 175–190. DOI: 10.1007/978-1-4939-0375-7_10.

[96] Matej Črepinšek, Marjan Mernik, and Shih-Hsi Liu. 'Analysis of Exploration and Exploitation in Evolutionary Algorithms by Ancestry Trees'. In: *International Journal of Innovative Computing and Applications* 3.1 (Jan. 2011), pp. 11–19. DOI: 10.1504/IJICA.2011.037947.

[97] Jernej Jerebic, Marjan Mernik, Shih-Hsi Liu, Miha Ravber, Mihael Baketarić, Luka Mernik, and Matej Črepinšek. 'A Novel Direct Measure of Exploration and Exploitation based on Attraction Basins'. In: *Expert Systems with Applications* 167 (Apr. 2021), p. 114353. DOI: 10.1016/j.eswa.2020.114353.

[98] Bogdan Burlacu, Michael Affenzeller, Stephan Winkler, Michael Kommenda, and Gabriel Kronberger. 'Methods for Genealogy and Building Block Analysis in Genetic Programming'. In: *Computational Intelligence and Efficiency in Engineering Systems*. Ed. by Grzegorz Borowik, Zenon Chaczko, Witold Jacak, and Tadeusz Łuba. Cham: Springer International Publishing, 2015, pp. 61–74. DOI: 10.1007/978-3-319-15720-7_5.

[99] Bogdan Burlacu, Michael Kommenda, and Michael Affenzeller. 'Building Blocks Identification Based on Subtree Sample Counts for Genetic Programming'. In: *2015 Asia-Pacific Conference on Computer Aided System Engineering*. July 2015, pp. 152–157. DOI: 10.1109/APCASE.2015.34.

[100] Bogdan Burlacu, Michael Affenzeller, and Michael Kommenda. 'On the Effectiveness of Genetic Operations in Symbolic Regression'. In: *Computer Aided Systems Theory – EUROCAST 2015*. Ed. by Roberto Moreno-Díaz, Franz Pichler, and Alexis Quesada-Arencibia. Cham: Springer International Publishing, 2015, pp. 367–374. DOI: 10.1007/978-3-319-27340-2_46.

[101] Emily L. Dolson, Wolfgang Banzhaf, and Charles Ofria. *Ecological Theory Provides Insights about Evolutionary Computation*. Tech. rep. e27315v1. PeerJ Inc., Nov. 2018. DOI: 10.7287/peerj.preprints.27315v1.

[102] Emily Dolson, Alexander Lalejini, Steven Jorgensen, and Charles Ofria. 'Interpreting the Tape of Life: Ancestry-Based Analyses Provide Insights and Intuition about Evolutionary Dynamics'. In: *Artificial Life* 26.1 (Apr. 2020), pp. 58–79. DOI: 10.1162/artl_a_00313.

[103] J.M. Daida, D.J. Ward, A.M. Hilss, S.L. Long, M.R. Hodges, and J.T. Kriesel. 'Visualizing the Loss of Diversity in Genetic Programming'. In: *Proceedings of the 2004 Congress on Evolutionary Computation*. Vol. 2. June 2004, 1225–1232 Vol.2. DOI: 10.1109/CEC.2004.1331037.

[104] Kenny Q. Zhu and Ziwei Liu. 'Population Diversity in Permutation-Based Genetic Algorithm'. In: *Machine Learning: ECML 2004*. Ed. by Jean-François Boulicaut, Floriana Esposito, Fosca Giannotti, and Dino Pedreschi. Berlin, Heidelberg: Springer, 2004, pp. 537–547. DOI: 10.1007/978-3-540-30115-8_49.

[105] Karthik Kuber, Stuart W. Card, Kishan G. Mehrotra, and Chilukuri K. Mohan. 'Ancestral Networks in Evolutionary Algorithms'. In: *Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation*. GECCO Comp '14. New York, NY, USA: Association for Computing Machinery, July 2014, pp. 115–116. DOI: 10.1145/2598394.2598481.

[106] Shih-Hsi Liu, Marjan Mernik, Dejan Hrnčič, and Matej Črepinšek. 'A Parameter Control Method of Evolutionary Algorithms using Exploration and Exploitation Measures with a Practical Application for Fitting Sovova's Mass Transfer Model'. In: *Applied Soft Computing* 13.9 (Sept. 2013), pp. 3792–3805. DOI: 10.1016/j.asoc.2013.05.010.

[107] E.K. Burke, S. Gustafson, G. Kendall, and N. Krasnogor. 'Is Increased Diversity in Genetic Programming Beneficial? An Analysis of Lineage Selection'. In: *The 2003 Congress on Evolutionary Computation*. Vol. 2. CEC '03. Dec. 2003, 1398–1405 Vol.2. DOI: 10.1109/CEC.2003.1299834.

[108] Alexander Lalejini, Matthew Andres Moreno, Jose Guadalupe Hernandez, and Emily Dolson. 'Phylogeny-Informed Fitness Estimation for Test-Based Parent Selection'. In: *Genetic Programming Theory and Practice XX*. Ed. by Stephan Winkler, Leonardo Trujillo, Charles Ofria, and Ting Hu. Singapore: Springer Nature, 2024, pp. 241–261. DOI: 10.1007/978-981-99-8413-8_13.

[109] Alexander Lalejini, Marcos Sanson, Jack Garbus, Matthew Andres Moreno, and Emily Dolson. 'Runtime Phylogenetic Analysis Enables Extreme Subsampling for Test-Based Problems'. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. GECCO '24 Companion. New York, NY, USA: Association for Computing Machinery, Aug. 2024, pp. 511–514. DOI: 10.1145/3638530.3654208.

[110] Alexander Lalejini, Marcos Sanson, Jack Garbus, Matthew Andres Moreno, and Emily Dolson. *Runtime Phylogenetic Analysis Enables Extreme Subsampling for Test-Based Problems*. Feb. 2024. DOI: 10.48550/arXiv.2402.01610. URL: http://arxiv.org/abs/2402.01610.

[111] Daryl Essam and R. I. (Bob) McKay. 'Heritage Diversity in Genetic Programming'. In: *The 5th International Conference on Simulated Evolution and Learning (SEAL'04)*. Busan, Korea, 2004.

[112] Cristian Ramirez-Atencia, Tobias Benecke, and Sanaz Mostaghim. 'T-EA: A Traceable Evolutionary Algorithm'. In: *2020 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–8. DOI: 10.1109/CEC48606.2020.9185615.

[113] Tobias Benecke. 'Tracing the Impact of the Initial Population in Evolutionary Algorithms'. MA thesis. Otto von Guericke University Magdeburg, 2020.

[114] Tobias Benecke and Sanaz Mostaghim. 'Tracking the Heritage of Genes in Evolutionary Algorithms'. In: *2021 IEEE Congress on Evolutionary Computation (CEC)*. June 2021, pp. 1800–1807. DOI: 10.1109/CEC45853.2021.9504916.

[115] Tobias Benecke and Sanaz Mostaghim. 'The Impact of Population Size on the Convergence of Multi-objective Evolutionary Algorithms'. In: *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*. Dec. 2021, pp. 1–8. DOI: 10.1109/SSCI50451.2021.9660164.

[116] R. Hinterding. 'Gaussian Mutation and Self-Adaption for Numeric Genetic Algorithms'. In: *Proceedings of 1995 IEEE International Conference on Evolutionary Computation*. Vol. 1. Nov. 1995, pp. 384–389. DOI: 10.1109/ICEC.1995.489178.

[117] C. E. Shannon. 'A Mathematical Theory of Communication'. In: *The Bell System Technical Journal* 27.3 (July 1948), pp. 379–423. DOI: 10.1002/j.1538-7305.1948.tb01338.x.

[118] Tobias Benecke and Sanaz Mostaghim. 'Effects of Optimal Genetic Material in the Initial Population of Evolutionary Algorithms'. In: *2023 IEEE Symposium Series on Computational Intelligence (SSCI)*. ISSN: 2472-8322. Dec. 2023, pp. 1386–1391. DOI: 10.1109/SSCI52147.2023.10372037.

[119] Pauli Virtanen et al. 'SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python'. In: *Nature Methods* 17.3 (Mar. 2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[120] Tobias Benecke and Sanaz Mostaghim. 'Tracing Genome Influence in Multi-Objective Evolutionary Algorithms'. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. GECCO '25 Companion. New York, NY, USA: Association for Computing Machinery, Aug. 2025, pp. 1807–1815. DOI: 10.1145/3712255.3734275.

[121] Tobias Friedrich and Markus Wagner. 'Seeding the Initial Population of Multi-Objective Evolutionary Algorithms: A Computational Study'. In: *Applied Soft Computing* 33 (Aug. 2015), pp. 223–230. DOI: 10.1016/j.asoc.2015.04.043.

[122] Cheng Gong, Yang Nan, Lie Meng Pang, Qingfu Zhang, and Hisao Ishibuchi. 'Effects of Including Optimal Solutions into Initial Population on Evolutionary Multiobjective Optimization'. In: *Proceedings of the Genetic and Evolutionary Computation Conference*. GECCO '23. New York, NY, USA: Association for Computing Machinery, July 2023, pp. 661–669. DOI: 10.1145/3583131.3590515.

[123] Sabine Helwig and Rolf Wanka. 'Particle Swarm Optimization in High-Dimensional Bounded Search Spaces'. In: *2007 IEEE Swarm Intelligence Symposium*. Apr. 2007, pp. 198–205. DOI: 10.1109/SIS.2007.368046.

[124] Markus Kress, Sanaz Mostaghim, Hartmut Schmeck, and Detlef Seese. 'Gap Search in Particle Swarm Optimization'. In: *9th International Conference on Artificial Evolution*. EA'09, 2009.

[125] Shahryar Rahnamayan, Hamid R. Tizhoosh, and Magdy M. A. Salama. 'A Novel Population Initialization Method for Accelerating Evolutionary Algorithms'. In: *Computers & Mathematics with Applications* 53.10 (May 2007), pp. 1605–1614. DOI: 10.1016/j.camwa.2006.07.013.

[126] I. M. Sobol'. 'On the Systematic Search in a Hypercube'. In: *SIAM Journal on Numerical Analysis* 16.5 (Oct. 1979), pp. 790–793. DOI: 10.1137/0716058.

[127] Borhan Kazimipour, Xiaodong Li, and A. K. Qin. 'Initialization Methods for Large Scale Global Optimization'. In: *2013 IEEE Congress on Evolutionary Computation*. ISSN: 1941-0026. June 2013, pp. 2750–2757. DOI: 10.1109/CEC.2013.6557902.

[128] Borhan Kazimipour, Xiaodong Li, and A. K. Qin. 'A Review of Population Initialization Techniques for Evolutionary Algorithms'. In: *2014 IEEE Congress on Evolutionary Computation (CEC)*. July 2014, pp. 2585–2592. DOI: 10.1109/CEC.2014.6900618.

[129] Mohammad Hasanzadeh and Farshid Keynia. 'An Overview of the Concepts, Classifications, and Methods of Population Initialization in Meta-Heuristic Algorithms'. In: *Journal of Advances in Computer Engineering and Technology (JACET)* 7 (May 2021), pp. 35–54.

[130] Jeffrey O. Agushaka, Absalom E. Ezugwu, Laith Abualigah, Samaher Khalaf Alharbi, and Hamiden Abd El-Wahed Khalifa. 'Efficient Initialization Methods for Population-Based Metaheuristic Algorithms: A Comparative Study'. In: *Archives of Computational Methods in Engineering* 30.3 (Apr. 2023), pp. 1727–1787. DOI: 10.1007/s11831-022-09850-4.

[131] Michael D. Schmidt and Hod Lipson. 'Incorporating Expert Knowledge in Evolutionary Search: A Study of Seeding Methods'. In: *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*. GECCO '09. New York, NY, USA: Association for Computing Machinery, July 2009, pp. 1091–1098. DOI: 10.1145/1569901.1570048.

[132] B. Saavedra-Moreno, S. Salcedo-Sanz, A. Paniagua-Tineo, L. Prieto, and A. Portilla-Figueras. 'Seeding Evolutionary Algorithms with Heuristics for Optimal Wind Turbines Positioning in Wind Farms'. In: *Renewable Energy* 36.11 (Nov. 2011), pp. 2838–2844. DOI: 10.1016/j.renene.2011.04.018.

[133] Saeid Kazemzadeh Azad. 'Seeding the Initial Population with Feasible Solutions in Metaheuristic Optimization of Steel Trusses'. In: *Engineering Optimization* 50.1 (Jan. 2018), pp. 89–105. DOI: 10.1080/0305215X.2017.1284833.

[134] Carlos M. Fernandes, Agostinho C. Rosa, J. L. J. Laredo, and J. J. Merelo. 'Genealogical Patterns in Evolutionary Algorithms'. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. GECCO '19. New York, NY, USA: Association for Computing Machinery, July 2019, pp. 283–284. DOI: 10.1145/3319619.3321914.

[135] Francisco Leonardo Bezerra Martins and José Cláudio do Nascimento. 'Power Law Dynamics in Genealogical Graphs'. In: *Physica A: Statistical Mechanics and its Applications* 596 (June 2022), p. 127174. DOI: 10.1016/j.physa.2022.127174.

[136] Darrell Whitley, Soraya Rana, John Dzubera, and Keith E. Mathias. 'Evaluating Evolutionary Algorithms'. In: *Artificial Intelligence* 85.1 (Aug. 1996), pp. 245–276. DOI: 10.1016/0004-3702(95)00124-7.

[137] Thomas Helmuth and Lee Spector. 'General Program Synthesis Benchmark Suite'. In: *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*. GECCO '15. New York, NY, USA: Association for Computing Machinery, July 2015, pp. 1039–1046. DOI: 10.1145/2739480.2754769.

[138] Bogdan Burlacu, Kaifeng Yang, and Michael Affenzeller. 'Population Diversity and Inheritance in Genetic Programming for Symbolic Regression'. In: *Natural Computing* 23.3 (Sept. 2024), pp. 531–566. DOI: 10.1007/s11047-022-09934-x.

[139] Jose Guadalupe Hernandez, Alexander Lalejini, and Emily Dolson. 'What Can Phylogenetic Metrics Tell us About Useful Diversity in Evolutionary Algorithms?' In: *Genetic Programming Theory and Practice XVIII*. Ed. by Wolfgang Banzhaf, Leonardo Trujillo, Stephan Winkler, and Bill Worzel. Singapore: Springer Nature, 2022, pp. 63–82. DOI: 10.1007/978-981-16-8113-4_4.

[140] Shakiba Shahbandegan, Jose Guadalupe Hernandez, Alexander Lalejini, and Emily Dolson. 'Untangling Phylogenetic Diversity's Role in Evolutionary Computation using a Suite of Diagnostic Fitness Landscapes'. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. GECCO '22. New York, NY, USA: Association for Computing Machinery, July 2022, pp. 2322–2325. DOI: 10.1145/3520304.3534028.

[141] Jason M. Daida. 'Towards Identifying Populations that Increase the Likelihood of Success in Genetic Programming'. In: *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*. GECCO '05. New York, NY, USA: Association for Computing Machinery, June 2005, pp. 1627–1634. DOI: 10.1145/1068009.1068284.

[142] Thomas Bäck. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. USA: Oxford University Press, Inc., 1996.

[143] Alden H. Wright. 'Genetic Algorithms for Real Parameter Optimization'. In: *Foundations of Genetic Algorithms*. Ed. by Gregory J. E. Rawlins. Vol. 1. Elsevier, Jan. 1991, pp. 205–218. DOI: 10.1016/B978-0-08-050684-5.50016-1.

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

[145] Cheng Gong, Ping Guo, Lie Meng Pang, Qingfu Zhang, and Hisao Ishibuchi. 'Population Initialization for Evolutionary Multi-objective Optimization: A Short Review'. In: *2025 IEEE Congress on Evolutionary Computation (CEC)*. June 2025, pp. 1–9. DOI: 10.1109/CEC65147.2025.11043105.

[146] Silvia Poles, Yan Fu, and Enrico Rigoni. 'The Effect of Initial Population Sampling on the Convergence of Multi-Objective Genetic Algorithms'. In: *Multiobjective Programming and Goal Programming*. Ed. by Vincent Barichard, Matthias Ehrgott, Xavier Gandibleux, and Vincent T'Kindt. Lecture Notes in Economics and Mathematical Systems. Berlin, Heidelberg: Springer, 2009, pp. 123–133. DOI: 10.1007/978-3-540-85646-7_12.

[147] Mohammad Hamdan and Osamah Qudah. 'The Initialization of Evolutionary Multi-Objective Optimization Algorithms'. In: *Advances in Swarm and Computational Intelligence*. Ed. by Ying Tan, Yuhui Shi, Fernando Buarque, Alexander Gelbukh, Swagatam Das, and Andries Engelbrecht. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2015, pp. 495–504. DOI: 10.1007/978-3-319-20466-6_52.

[148] Cheng Gong, Lie Meng Pang, Yang Nan, Hisao Ishibuchi, and Qingfu Zhang. 'Effects of Initialization Methods on the Performance of Multi-Objective Evolutionary Algorithms'. In: *2023 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. Oct. 2023, pp. 1168–1175. DOI: 10.1109/SMC53992.2023.10394232.

[149] Alfredo G. Hernandez-Diaz, Carlos A. Coello Coello, Fatima Perez, Rafael Caballero, Julian Molina, and Luis V. Santana-Quintero. 'Seeding the Initial Population of a Multi-Objective Evolutionary Algorithm using Gradient-Based Information'. In: *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*. June 2008, pp. 1617–1624. DOI: 10.1109/CEC.2008.4631008.

[150] Cheng Gong, Yang Nan, Lie Meng Pang, Hisao Ishibuchi, and Qingfu Zhang. 'Initial Populations with a Few Heuristic Solutions Significantly Improve Evolutionary Multi-Objective Combinatorial Optimization'. In: *2023 IEEE Symposium Series on Computational Intelligence (SSCI)*. Dec. 2023, pp. 1398–1405. DOI: 10.1109/SSCI52147.2023.10371937.

[151] Yuji Sato, Mikiko Sato, and Hiroki Yamada. 'Initial Population of NSGA-II for Solving Similar Multi-Objective Optimization Problems'. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. GECCO '25 Companion. New York, NY, USA: Association for Computing Machinery, Aug. 2025, pp. 109–110. DOI: 10.1145/3712255.3734264.

[152] Hartmut Pohlheim. 'Multidimensional Scaling for Evolutionary Algorithms–Visualization of the Path through Search Space and Solution Space using Sammon mapping'. In: *Artificial Life* 12.2 (2006), pp. 203–209. DOI: 10.1162/106454606776073305.

[153] Sébastien Verel, Arnaud Liefooghe, Laetitia Jourdan, and Clarisse Dhaenens. 'On the Structure of Multiobjective Combinatorial Search Space: *MNK*-Landscapes with Correlated Objectives'. In: *European Journal of Operational Research* 227.2 (June 2013), pp. 331–342. DOI: 10.1016/j.ejor.2012.12.019.

[154] Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Mauro Birattari, and Thomas Stützle. 'The irace Package: Iterated Racing for Automatic Algorithm Configuration'. In: *Operations Research Perspectives* 3 (Jan. 2016), pp. 43–58. DOI: 10.1016/j.orp.2016.09.002.

[155] Zhun Fan, Wenji Li, Xinye Cai, Hui Li, Caimin Wei, Qingfu Zhang, Kalyanmoy Deb, and Erik Goodman. 'Difficulty Adjustable and Scalable Constrained Multiobjective Test Problem Toolkit'. In: *Evolutionary Computation* 28.3 (Sept. 2020), pp. 339–378. DOI: 10.1162/evco_a_00259.

[156] Yuri Lavinas, Marcelo Ladeira, Gabriela Ochoa, and Claus Aranha. 'Multiobjective Evolutionary Component Effect on Algorithm Behavior'. In: *ACM Transactions on Evolutionary Learning and Optimization* 4.2 (June 2024), 8:1–8:24. DOI: 10.1145/3612933.

[157] Takehisa Kohira, Hiromasa Kemmotsu, Oyama Akira, and Tomoaki Tatsukawa. 'Proposal of Benchmark Problem Based on Real-World Car Structure Design Optimization'. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. GECCO '18. New York, NY, USA: Association for Computing Machinery, July 2018, pp. 183–184. DOI: 10.1145/3205651.3205702.

[158] Aljoša Vodopija, Jordan N. Cork, and Bogdan Filipič. 'The Lunar Lander Landing Site Selection Benchmark Reexamined: Problem Characterization and Algorithm Performance'. In: *Proceedings of the Genetic and Evolutionary Computation Conference*. GECCO '24. New York, NY, USA: Association for Computing Machinery, July 2024, pp. 1381–1389. DOI: 10.1145/3638529.3654229.

[159] Shih-Hsi Liu, Matej Črepinšek, and Marjan Mernik. 'Analysis of VEGA and SPEA2 using Exploration and Exploitation Measures'. In: *Proceedings of the 5th International Conference on Bioinspired Optimization Methods and their Applications, BIOMA*. 2012, pp. 97–108.

[160] Daniel Mora-Melià, F. Javier Martínez-Solano, Pedro L. Iglesias-Rey, and Jimmy H. Gutiérrez-Bahamondes. 'Population Size Influence on the Efficiency of Evolutionary Algorithms to Design Water Networks'. In: *Procedia Engineering*. XVIII International Conference on Water Distribution Systems, WDSA2016 186 (Jan. 2017), pp. 341–348. DOI: 10.1016/j.proeng.2017.03.209.

[161] Hiroaki Fukumoto and Akira Oyama. 'Study on Improving Efficiency of Multi-Objective Evolutionary Algorithm with Large Population by M2M Decomposition and Elitist Mate Selection Scheme'. In: *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*. Nov. 2018, pp. 1180–1187. DOI: 10.1109/SSCI.2018.8628813.

[162] Thomas Weise, Yuezhong Wu, Raymond Chiong, Ke Tang, and Jörg Lässig. 'Global Versus Local Search: The Impact of Population Sizes on Evolutionary Algorithm Performance'. In: *J. of Global Optimization* 66.3 (Nov. 2016), pp. 511–534. DOI: 10.1007/s10898-016-0417-5.

[163] Tianshi Chen, Ke Tang, Guoliang Chen, and Xin Yao. 'A Large Population Size can be Unhelpful in Evolutionary Algorithms'. In: *Theoretical Computer Science* 436 (June 2012), pp. 54–70. DOI: 10.1016/j.tcs.2011.02.016.

[164] Ting Hu and Wolfgang Banzhaf. 'Nonsynonymous to Synonymous Substitution Ratio k_a/k_s: Measurement for Rate of Evolution in Evolutionary Computation'. In: *Parallel Problem Solving from Nature – PPSN X*. Ed. by Günter Rudolph, Thomas Jansen, Nicola Beume, Simon Lucas, and Carlo Poloni. Berlin, Heidelberg: Springer, 2008, pp. 448–457. DOI: 10.1007/978-3-540-87700-4_45.

[165] J.T. Alander. 'On Optimal Population Size of Genetic Algorithms'. In: *CompEuro 1992 Proceedings Computer Systems and Software Engineering*. May 1992, pp. 65–70. DOI: 10.1109/CMPEUR.1992.218485.

[166] J.I. Hidalgo and F. Fernandez. 'Balancing the Computation Effort in Genetic Algorithms'. In: *2005 IEEE Congress on Evolutionary Computation*. Vol. 2. Sept. 2005, 1645–1652 Vol. 2. DOI: 10.1109/CEC.2005.1554886.

[167] Olympia Roeva, Stefka Fidanova, and Marcin Paprzycki. 'Influence of the Population Size on the Genetic Algorithm Performance in Case of Cultivation Process Modelling'. In: *2013 Federated Conference on Computer Science and Information Systems*. Sept. 2013, pp. 371–376.

[168] Fernando G. Lobo and David E. Goldberg. 'The Parameter-Less Genetic Algorithm in Practice'. In: *Information Sciences* 167.1 (Dec. 2004), pp. 217–232. DOI: 10.1016/j.ins.2003.03.029.

[169] K.C. Tan, T.H. Lee, and E.F. Khor. 'Evolutionary Algorithms with Dynamic Population Size and Local Exploration for Multiobjective Optimization'. In: *IEEE Transactions on Evolutionary Computation* 5.6 (Dec. 2001), pp. 565–588. DOI: 10.1109/4235.974840.

[170] Carlos Fernandes and Agostinho Rosa. 'Self-Regulated Population Size in Evolutionary Algorithms'. In: *Parallel Problem Solving from Nature - PPSN IX*. Ed. by Thomas Philip Runarsson, Hans-Georg Beyer, Edmund Burke, Juan J. Merelo-Guervós, L. Darrell Whitley, and Xin Yao. Berlin, Heidelberg: Springer, 2006, pp. 920–929. DOI: 10.1007/11844297_93.

[171] Thomas Jansen, Kenneth A. De Jong, and Ingo Wegener. 'On the Choice of the Offspring Population Size in Evolutionary Algorithms'. In: *Evolutionary Computation* 13.4 (2005), pp. 413–440. DOI: 10.1162/106365605774666921.

[172] Carlos A. Coello Coello and Margarita Reyes Sierra. 'A Study of the Parallelization of a Coevolutionary Multi-objective Evolutionary Algorithm'. In: *MICAI 2004: Advances in Artificial Intelligence*. Ed. by Gerhard Goos, Juris Hartmanis, Jan van Leeuwen, Raúl Monroy, Gustavo Arroyo-Figueroa, Luis Enrique Sucar, and Humberto Sossa. Vol. 2972. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 688–697. DOI: 10.1007/978-3-540-24694-7_71.

[173] Tobias Benecke and Sanaz Mostaghim. 'Estimating the Quality of Initial Populations in Multi-Objective Evolutionary Algorithms'. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. GECCO '22. New York, NY, USA: Association for Computing Machinery, July 2022, pp. 324–327. DOI: 10.1145/3520304.3528796.

[174] Tobias Benecke and Sanaz Mostaghim. *Code for "Tracing Genome Influence in Multi-Objective Evolutionary Algorithms"*. May 2025. DOI: 10.5281/zenodo.15391664.

[175] Tobias Benecke and Sanaz Mostaghim. 'Supplementary Material for "Tracing Genome Influence in Multi-Objective Evolutionary Algorithms"'. In: (May 2025). DOI: 10.5281/zenodo.15389970.

Publications of the Author

[5] Tobias Benecke, Oliver Antons, Sanaz Mostaghim, and Julia Arlinghaus. 'A Generalized Circular Supply Chain Problem for Multi-objective Evolutionary Algorithms'. In: *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*. GECCO '23 Companion. New York, NY, USA: Association for Computing Machinery, July 2023, pp. 355–358. DOI: 10.1145/3583133.3590742.

[6] Tobias Benecke, Oliver Antons, Sanaz Mostaghim, and Julia Arlinghaus. 'A Coevolution Approach for the Multi-objective Circular Supply Chain Problem'. In: *2023 IEEE Conference on Artificial Intelligence (CAI)*. June 2023, pp. 222–223. DOI: 10.1109/CAI54212.2023.00103.

[45] Andrejaana Andova, Tobias Benecke, Harald Ludwig, and Tea Tušar. 'Towards Constructing a Suite of Multi-Objective Optimization Problems with Diverse Landscapes'. In: *Applications of Evolutionary Computation*. Ed. by João Correia, Stephen Smith, and Raneem Qaddoura. Lecture Notes in Computer Science. Cham: Springer Nature Switzerland, 2023, pp. 442–457. DOI: 10.1007/978-3-031-30229-9_29.

[50] Sebastian Mai, Tobias Benecke, and Sanaz Mostaghim. 'MACO: A Real-World Inspired Benchmark for Multi-objective Evolutionary Algorithms'. In: *Evolutionary Multi-Criterion Optimization*. Ed. by Michael Emmerich, André Deutz, Hao Wang, Anna V. Kononova, Boris Naujoks, Ke Li, Kaisa Miettinen, and Iryna Yevseyeva. Lecture Notes in Computer Science. Cham: Springer Nature Switzerland, 2023, pp. 305–318. DOI: 10.1007/978-3-031-27250-9_22.

[51] Oliver Antons, Tobias Benecke, Sanaz Mostaghim, and Julia C. Arlinghaus. 'Sustainability in Chemical Production – Multi-Objective Distributed Control'. In: *New Trends in Intelligent Software Methodologies, Tools and Techniques*. IOS Press, 2023, pp. 211–219. DOI: 10.3233/FAIA230236.

[112] Cristian Ramirez-Atencia, Tobias Benecke, and Sanaz Mostaghim. 'T-EA: A Traceable Evolutionary Algorithm'. In: *2020 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–8. DOI: 10.1109/CEC48606.2020.9185615.

[113] Tobias Benecke. 'Tracing the Impact of the Initial Population in Evolutionary Algorithms'. MA thesis. Otto von Guericke University Magdeburg, 2020.

[114] Tobias Benecke and Sanaz Mostaghim. 'Tracking the Heritage of Genes in Evolutionary Algorithms'. In: *2021 IEEE Congress on Evolutionary Computation (CEC)*. June 2021, pp. 1800–1807. DOI: 10.1109/CEC45853.2021.9504916.

[115] Tobias Benecke and Sanaz Mostaghim. 'The Impact of Population Size on the Convergence of Multi-objective Evolutionary Algorithms'. In: *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*. Dec. 2021, pp. 1–8. DOI: 10.1109/SSCI50451.2021.9660164.

[118] Tobias Benecke and Sanaz Mostaghim. 'Effects of Optimal Genetic Material in the Initial Population of Evolutionary Algorithms'. In: *2023 IEEE Symposium Series on Computational Intelligence (SSCI)*. ISSN: 2472-8322. Dec. 2023, pp. 1386–1391. DOI: 10.1109/SSCI52147.2023.10372037.

[120] Tobias Benecke and Sanaz Mostaghim. 'Tracing Genome Influence in Multi-Objective Evolutionary Algorithms'. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. GECCO '25 Companion. New York, NY, USA: Association for Computing Machinery, Aug. 2025, pp. 1807–1815. DOI: 10.1145/3712255.3734275.

[173] Tobias Benecke and Sanaz Mostaghim. 'Estimating the Quality of Initial Populations in Multi-Objective Evolutionary Algorithms'. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. GECCO '22. New York, NY, USA: Association for Computing Machinery, July 2022, pp. 324–327. DOI: 10.1145/3520304.3528796.

Acronyms

CSC	circular supply-chain
DE	differential evolution
DEAP	distributed evolutionary algorithms in python
EA	evolutionary algorithm
EC	evolutionary computation
ELA	exploratory landscape analysis
EP	evolutionary programming
ES	evolutionary strategy
ETV	event takeover value
GA	genetic algorithm
GD	generational distance
GM	Gaussian mutation
GP	genetic programming
GPBIG	genetic programming using best individuals of genealogies
GSS	gap-search sampling
HV	hypervolume
IGD	inverted generational distance
IGD+	inverted generational distance plus
LHS	Latin-hypercube sampling
LLM	large language model
LX	linear crossover
MACO	multi-agent coordination problem
MDS	multidimensional scaling
MOEA	multi-objective evolutionary algorithm
MOEA/D	multi-objective evolutionary algorithm based on decomposition
MOO	multi-objective optimization
MOP	multi-objective optimization problem
NDS	non-dominated solution
NEAT	neuroevolution of augmented typologies
NN	neuronal network
NSGA-II	non-dominated sorting genetic algorithm II
NSGA-III	non-dominated sorting genetic algorithm III
NTN	neuroevolution trajectory network
OBL	opposition-based learning
PCA	principal component analysis

PF	Pareto-front
PM	polynomial mutation
PS	Pareto-set
PSO	particle swarm optimization
SBX	simulated binary crossover
SOEA	single-objective evolutionary algorithm
SOO	single-objective optimization
SOP	single-objective optimization problem
SPEA2	strength Pareto evolutionary algorithm 2
STN	search trajectory network
STNWeb	search trajectory networks on the web
T-EA	traceable evolutionary algorithm
t-SNE	t-distributed stochastic neighbor embedding
TSP	traveling salesman problem
UM	uniform mutation
UMAP	uniform manifold approximation and projection
UX	uniform crossover
VEGA	vector evaluated genetic algorithm
XAI	explainable artificial intelligence

Symbols

$I_{F \cdot E}$	fitness and entropy-based impact metric. See Equation 4.9 for trace-vector definition and Equation 4.23 for trace-list definition
$\mathfrak{C}$	contributors. See Equation 4.1 for trace-vector definition and Equation 4.14 for trace-list definition
$\vec{f}$	a vector of $\vec{f}(\vec{x}) = \left(f_1(\vec{x}), \ldots, f_{d_o}(\vec{x})\right)$ objective functions for a o dimensional optimization problem
$f(\vec{x}_i)$	objective function for a o dimensional optimization problem
H	Shannon-entropy of the heritage information in the decision variable j (see Equation 4.7)
I_C	counting-based impact metric. See Equation 4.2 for trace-vector definition and Equation 4.16 for trace-list definition
I_E	entropy-based impact metric. See Equation 4.8 for trace-vector definition and Equation 4.22 for trace-list definition
I_F	fitness-based impact metric. See Equation 4.4 for trace-vector definition and Equation 4.17 for trace-list definition
k	used as a placeholder for a traceID
n	population size, or the amount of individuals in one population X
$\mathfrak{O}$	a o dimensional objective space of an optimization problem
o	dimensionality of the objective space, and the number of objective functions in an optimization problem
θ	number of optimal genes on a genome
p^{opt}	percentage of how many genes in a genome are optimal $p^{opt} = \frac{\tau}{\tau+\theta}$
ρ	Number of non-dominated solution (NDS) of a population $\rho = \lvert X^{NDS} \rvert$
τ	number of not optimal genes on a genome
$\mathcal{S}$	a $\mathfrak{s}$ dimensional search space of an optimization problem

$\mathfrak{z}$	dimensionality of the search space, or genome size of an individual
t	traceID storing the heritage, pointing to its origin in the initial population
TL	trace lists of a population of n individuals ($TV = [tl_1, \ldots, \overrightarrow{tl}_n]$)
tl	trace-list of a gene, containing the trace tuples (tt) which influence the gene
tt	trace tuple $tt = (t, \omega)$
TV	trace vectors of a population of n individuals ($TV = \left[\overrightarrow{tv}_1, \ldots, \overrightarrow{tv}_n\right]$)
$\overrightarrow{tv}$	trace-vector of an individual containing the traceIDs for each gene ($\overrightarrow{tv} = [t_1, \ldots, t_n]$)
ω	second element of a trace tuple (tt), denotes the weight of a traceID t on the gene
X	genomes of a population of n individuals ($X = [\vec{x}_1, \ldots, \vec{x}_n]$)
X^{NDS}	Set of all non-dominated solution (NDS) of a population X
$\vec{x}$	decision vector $\vec{x} \in S$ of an optimization problem, also referred to as the genome of an individual in the context of EAs
$\vec{x}_i$	genome of the ith individual in a population X
$x_{i,j}$	gene value of the gene j in individual i
$\vec{x}^{opt}$	optimal decision vector $\vec{x}^{opt} \in S$ of an optimization problem

List of Figures

1.1 Basic process of an EA. . . . 1

2.1 Basic process of an EA. . . . 8
2.2 Example of one-point crossover. . . . 9
2.3 Example of bitflip mutation. . . . 9
2.4 Visual example of Pareto-dominance. . . . 11
2.5 Visual example of NDS. . . . 11
2.6 Exemplary PS and PF for a bi-objective optimization problem with two decision variables. . . . 12
2.7 Fitness landscape of the Sphere function. . . . 14
2.8 Fitness landscape of the Rastrigin function. . . . 14
2.9 Fitness landscape of the Rosenbrock function. . . . 14
2.10 Genome variable distribution for 100 samples of the ZDT 1 PS. . . . 15
2.11 Genome variable distribution for 100 samples of the UF 1 PS. . . . 15
2.12 Examples of typical plots to evaluate the performance of EAs. Two crossover operators are compared. . . . 17
2.13 Visual example of HV. . . . 18
2.14 Visual example of GD. . . . 19
2.15 Visual example of IGD. . . . 19
2.16 Visual example of IGD+. . . . 19
2.17 Examples of typical plots to evaluate the performance of MOEAs. NSGA-II and MOEA/D are compared with the IGD+ performance indicator. . . . 20
2.18 Example of a boxplot comparing the IGD+ of 31 test repeats for two algorithmic configurations. 20

3.1 Example of a 2D STN visualization, created with the exemplary dataset at `https://www.stn-analytics.com/`. . . . 27
3.2 Genealogy tree of the DEAP framework [91]. . . . 29

4.1 TraceID implementation of T-EA. . . . 41
4.2 Initialization of the T-EA. . . . 41
4.3 Crossover example for trace vectors. . . . 41
4.4 Mutation example for trace vectors. . . . 41
4.5 Trace-list encoding. . . . 43
4.6 Example of trace-list encoded individuals in generation 1 and n. . . . 44
4.7 Trace-list example crossover. . . . 44
4.8 Trace-list example mutation. . . . 45

5.1 Fitness landscape of Rastrigin with $\mathfrak{z}= 1$ and an $f^{target} = 10$. . . . 52
5.2 Examples seed locations on the PF for two- and three-objective problems. . . . 53

7.1 Fitness of the three initial populations. The median fitness is marked with a dashed line. . . . 69
7.2 I_C for the three different initial populations using UX and GM. . . . 69
7.3 I_F for the three different initial populations using UX and GM. . . . 70
7.4 I_E for the three different initial populations using UX and GM. . . . 70
7.5 $I_{F \cdot E}$ for the three different initial populations using UX and GM. . . . 71
7.6 Impact metrics over the generation for test run 15 of the three populations using UX and GM. 72
7.7 I_C for the tree populations using UX and UM. . . . 75
7.8 I_C for the tree populations using UX and GM. . . . 75
7.9 I_C for the tree populations using LX and UM. . . . 76
7.10 I_C for the tree populations using LX and GM. . . . 76

8.1 Performance on the Sphere function of the upper quartile seed over multiple search space dimensions. . . . 84
8.2 Seed influence for the Sphere function of the upper quartile seed over multiple search space dimensions. . . . 85
8.3 Performance on the Sphere function of the different seed qualities for $\mathfrak{z}$= 8. . . . 86
8.4 Seed influence for the Sphere function of the different seed qualities for $\mathfrak{z}$= 8. . . . 87
8.5 Performance on the Rastrigin function ($\mathfrak{z}$= 16). . . . 88
8.6 Seed influence for the Rastrigin function ($\mathfrak{z}$= 16). . . . 88
8.7 Performance on the Rosenbrock function ($\mathfrak{z}$= 16). . . . 89
8.8 Seed influence for the Rosenbrock function ($\mathfrak{z}$= 16). . . . 90
8.9 Difference in distance to x_{opt} for the close and far Rastrigin seeds. . . . 90
8.10 Difference in distance to x_{opt} for the close and far Rosenbrock seeds. . . . 90

10.1 Results for the population size tests [115]. . . . 105
10.2 The median number of contributing individuals in every generation for the tests with the population size of 100 [115]. . . . 106
10.3 Results for impact of mutation in the final generation as box plots for all test runs. The top graph shows the results of the population size of 10, followed by 50, 100, 200, and 300 is shown in the bottom graph [115]. . . . 108

11.1 Difference in IGD+ compared to the random seed type r for the two-objective problems in percentage. . . . 116
11.2 Difference in IGD+ compared to the random seed type r for the three-objective problems in percentage. . . . 117
11.3 Counting impact of the different seed locations for two-objective problems [120]. . . . 117
11.4 Counting impact of the different seed locations for three-objective problems [120]. . . . 118
11.5 Seed Impact on the PF for DTLZ 3 problem. The true PF is shown with a grey line, while the green dots represent the accumulated NDS for the 31 repeats using the respective seed individual [120]. . . . 119
11.6 Seed Impact on the PF for MACO_b. The true PF is shown with a grey line, while the green dots represent the accumulated NDS for the 31 repeats using the respective seed individual [120]. . . . 120
11.7 Seed impact on the NDS for each seed location of the combined $e1 + e2 + c$ configuration on ZDT 3 [120]. . . . 121
11.8 Seed impact on the NDS for each seed location of the combined $e1 + e2 + c$ configuration on UF 1 [120]. . . . 121

B.1 Performance on the Sphere function ($\mathfrak{z}$ = 4). . . . 162
B.2 Seed influence for the Sphere function ($\mathfrak{z}$ = 4). . . . 162
B.3 Performance on the Sphere function ($\mathfrak{z}$ = 8). . . . 162
B.4 Seed influence for the Sphere function ($\mathfrak{z}$ = 8). . . . 162
B.5 Performance on the Sphere function ($\mathfrak{z}$ = 16). . . . 163
B.6 Seed influence for the Sphere function ($\mathfrak{z}$ = 16). . . . 163
B.7 Performance on the Sphere function ($\mathfrak{z}$ = 32). . . . 163
B.8 Seed influence for the Sphere function ($\mathfrak{z}$ = 32). . . . 163
B.9 Performance on the Rastrigin function ($\mathfrak{z}$ = 4, "close" seed). . . . 164
B.10 Seed influence for the Rastrigin function ($\mathfrak{z}$ = 4 "close" seed). . . . 164
B.11 Performance on the Rastrigin function ($\mathfrak{z}$ = 4, "far" seed). . . . 164
B.12 Seed influence for the Rastrigin function ($\mathfrak{z}$ = 4 "far" seed). . . . 164
B.13 Performance on the Rastrigin function ($\mathfrak{z}$ = 8, "close" seed). . . . 165
B.14 Seed influence for the Rastrigin function ($\mathfrak{z}$ = 8 "close" seed). . . . 165
B.15 Performance on the Rastrigin function ($\mathfrak{z}$ = 8, "far" seed). . . . 165
B.16 Seed influence for the Rastrigin function ($\mathfrak{z}$ = 8 "far" seed). . . . 165
B.17 Performance on the Rastrigin function ($\mathfrak{z}$ = 16, "close" seed). . . . 166

B.18 Seed influence for the Rastrigin function ($\mathfrak{z} = 16$ "close" seed). 166
B.19 Performance on the Rastrigin function ($\mathfrak{z} = 16$, "far" seed). 166
B.20 Seed influence for the Rastrigin function ($\mathfrak{z} = 16$ "far" seed). 166
B.21 Performance on the Rastrigin function ($\mathfrak{z} = 32$, "close" seed). 167
B.22 Seed influence for the Rastrigin function ($\mathfrak{z} = 32$ "close" seed). 167
B.23 Performance on the Rastrigin function ($\mathfrak{z} = 32$, "far" seed). 167
B.24 Seed influence for the Rastrigin function ($\mathfrak{z} = 32$ "far" seed). 167
B.25 Performance on the Rosenbrock function ($\mathfrak{z} = 4$, "close" seed). 168
B.26 Seed influence for the Rosenbrock function ($\mathfrak{z} = 4$ "close" seed). 168
B.27 Performance on the Rosenbrock function ($\mathfrak{z} = 4$, "far" seed). 168
B.28 Seed influence for the Rosenbrock function ($\mathfrak{z} = 4$ "far" seed). 168
B.29 Performance on the Rosenbrock function ($\mathfrak{z} = 8$, "close" seed). 169
B.30 Seed influence for the Rosenbrock function ($\mathfrak{z} = 8$ "close" seed). 169
B.31 Performance on the Rosenbrock function ($\mathfrak{z} = 8$, "far" seed). 169
B.32 Seed influence for the Rosenbrock function ($\mathfrak{z} = 8$ "far" seed). 169
B.33 Performance on the Rosenbrock function ($\mathfrak{z} = 16$, "close" seed). 170
B.34 Seed influence for the Rosenbrock function ($\mathfrak{z} = 16$ "close" seed). 170
B.35 Performance on the Rosenbrock function ($\mathfrak{z} = 16$, "far" seed). 170
B.36 Seed influence for the Rosenbrock function ($\mathfrak{z} = 16$ "far" seed). 170
B.37 Performance on the Rosenbrock function ($\mathfrak{z} = 32$, "close" seed). 171
B.38 Seed influence for the Rosenbrock function ($\mathfrak{z} = 32$ "close" seed). 171
B.39 Performance on the Rosenbrock function ($\mathfrak{z} = 32$, "far" seed). 171
B.40 Seed influence for the Rosenbrock function ($\mathfrak{z} = 32$ "far" seed). 171

List of Tables

3.1 Literature overview of population history and dynamics used in the design of EAs. 35
3.2 Literature overview of evaluation methods for population dynamics EAs excluding the contributions of this thesis. 37

6.1 Literature overview of papers evaluating population dynamics of SOEAs, excluding the contributions of this thesis. 65

7.1 Performance for all test configurations, measured as the median for the best solutions found by each algorithm configuration. 74
7.2 The median values of all four impact metrics of population 1. Results are rounded to three decimal places. Statistically significant differences to the I_C metric are marked with a "*". . . 78
7.3 The median values of all four impact metrics of population 2. Results are rounded to three decimal places. Statistically significant differences to the I_C metric are marked with a "*". . . 78
7.4 The median values of all four impact metrics of population 3. Results are rounded to three decimal places. Statistically significant differences to the I_C metric are marked with a "*". . . 78

8.1 Performance comparison between the close (c) vs. far (f) seed individuals. 91
8.2 Influence comparison between the close (c) vs. far (f) seed individuals. 92

9.1 Literature overview of papers evaluating population dynamics of MOEAs, excluding the contributions of this thesis. 100

10.1 The dimensionality in search and objective space of the used test problems 102
10.2 Median GD and IGD values for each problem and population size. 103
10.3 Lowest, median, and highest number of contributors per population size for all problems as absolute and relative values. 104

11.1 Average HV and IGD+ for two-objective problems. 115
11.2 Average HV and IGD+ for three-objective problems. 115

Part IV

Appendix

A

Code and Data

For transparency and reproducibility, we provide an overview of the code and supplementary material published alongside the author's publications during his PhD journey.

Code and Data for the Experiments in this Dissertation

The code and data for all four experimental chapters of this thesis are available:

- *Chapter 7 - Influence of Operators in SOEAs*: This chapter is based on the author's publications [112, 114], but were repeated for this thesis with the newest implementation of the T-EA. Code and data for the experiments in this thesis are available on GitHub[1].
- *Chapter 8 - Influential Individuals in SOEAs*: This chapter is based the author's publication [118]. Code for the original publication [118] is available on GitHub[2]. Code for the extended tests in this thesis is also available on GitHub[3]. Additional visualization for the tests can also be found in the Appendix B.
- *Chapter 10 - Influence of Population Size in MOEA*: This chapter is based on the author's publication [115]. The code is available on GitHub[4].
- *Chapter 11 - Influential Individuals in MOEAs*: This chapter is based on the author's publication [120]. Supplementary material [175] and code [174] are available, as referenced, on Zenodo.

[1] Code for Chapter 7: `https://github.com/tobeneck/soea_operator_influence.git`
[2] Code for [118]: `https://github.com/tobeneck/soea_influential_inds.git`
[3] Code for Chapter 8: `https://github.com/tobeneck/soea_influential_inds_extended.git`
[4] Code for Chapter 10: `https://github.com/tobeneck/moea_popsize.git`

B

Supplementary Material for Chapter 8

This chapter contains additional visualizations for the data presented in Chapter 8. First, the data for the Sphere function is shown, comparing the performance and impact of the seed for the "close" and "far" seeds in ascending search space dimensions. This is followed by the Rastrigin and Rosenbrock results.

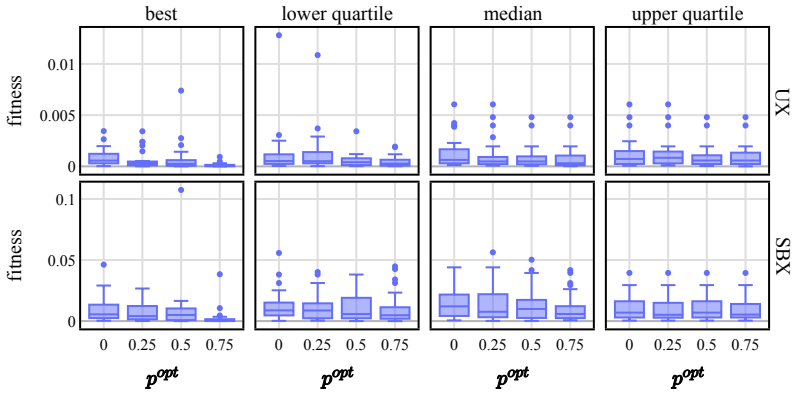


Figure B.1: Performance on the Sphere function ($\flat = 4$).

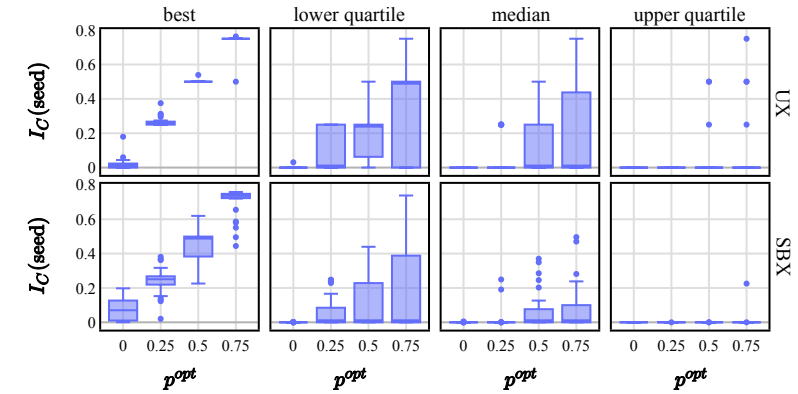


Figure B.2: Seed influence for the Sphere function ($\flat = 4$).

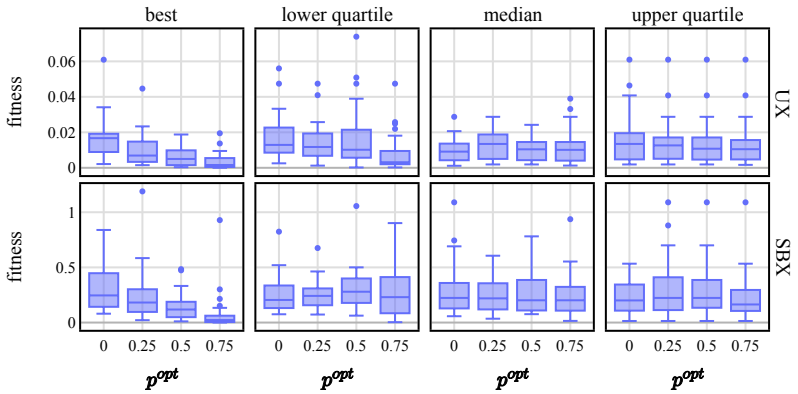


Figure B.3: Performance on the Sphere function ($\flat = 8$).

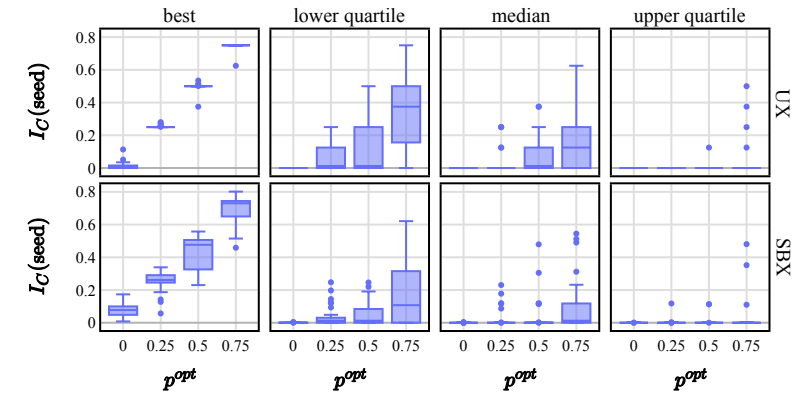


Figure B.4: Seed influence for the Sphere function ($\flat = 8$).

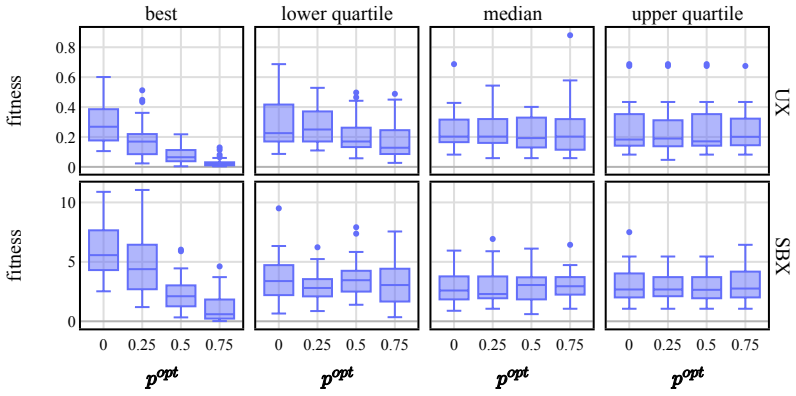


Figure B.5: Performance on the Sphere function ($\flat = 16$).

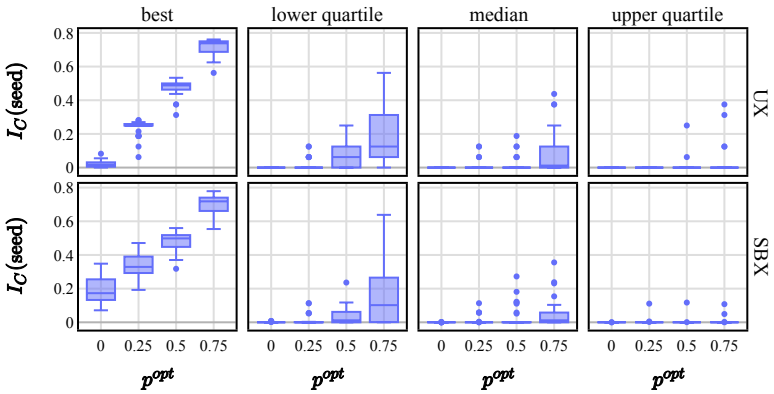


Figure B.6: Seed influence for the Sphere function ($\flat = 16$).

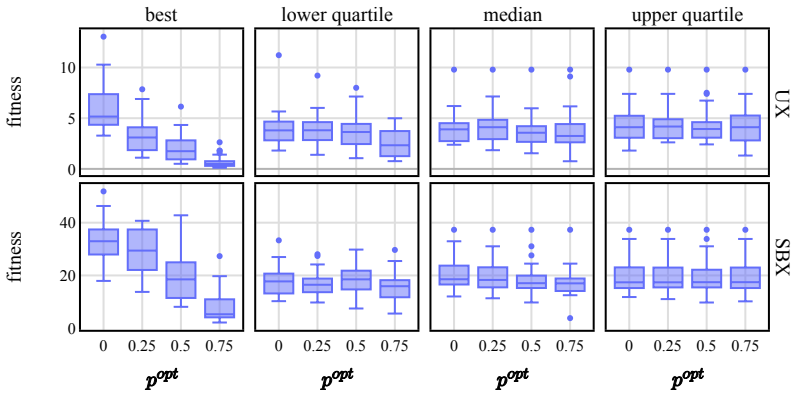


Figure B.7: Performance on the Sphere function ($\flat = 32$).

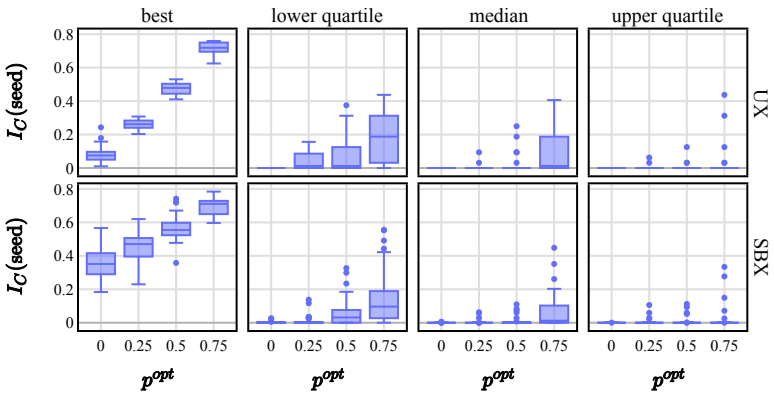


Figure B.8: Seed influence for the Sphere function ($\flat = 32$).

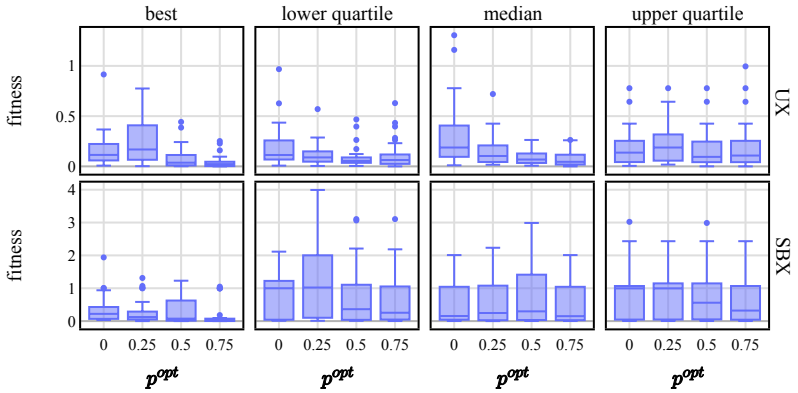


Figure B.9: Performance on the Rastrigin function ($\mathfrak{z} = 4$, "close" seed).

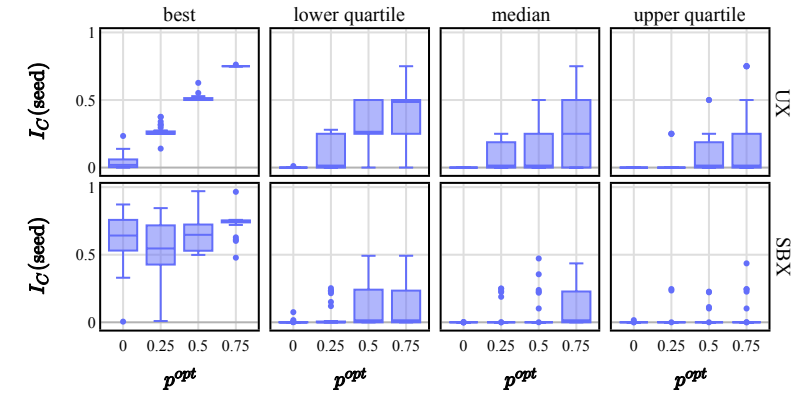


Figure B.10: Seed influence for the Rastrigin function ($\mathfrak{z} = 4$ "close" seed).

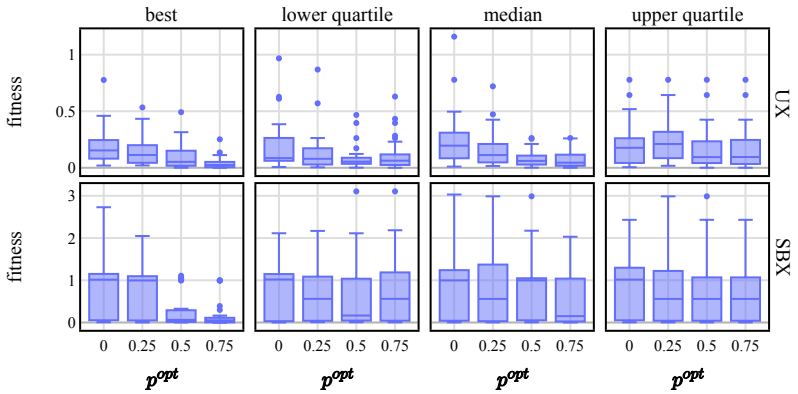


Figure B.11: Performance on the Rastrigin function ($\mathfrak{z} = 4$, "far" seed).

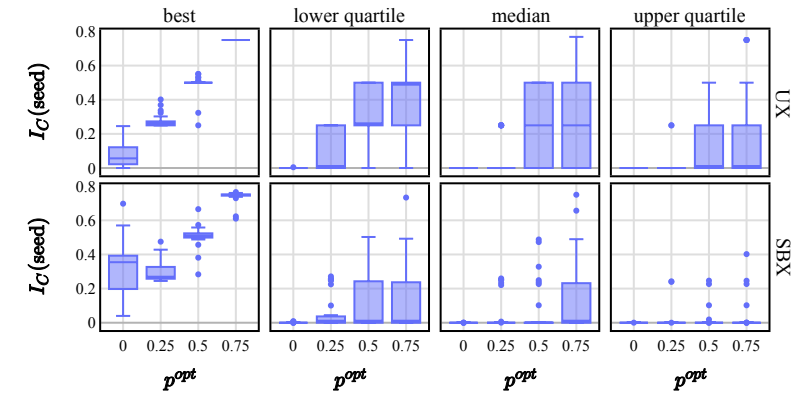


Figure B.12: Seed influence for the Rastrigin function ($\mathfrak{z} = 4$ "far" seed).

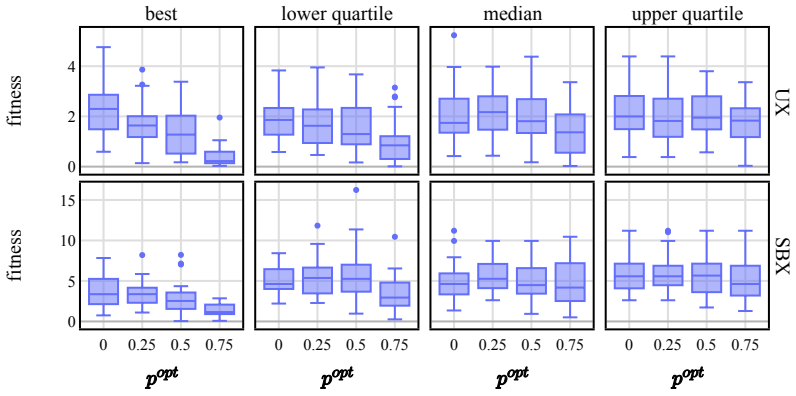


Figure B.13: Performance on the Rastrigin function ($\mathfrak{z} = 8$, "close" seed).

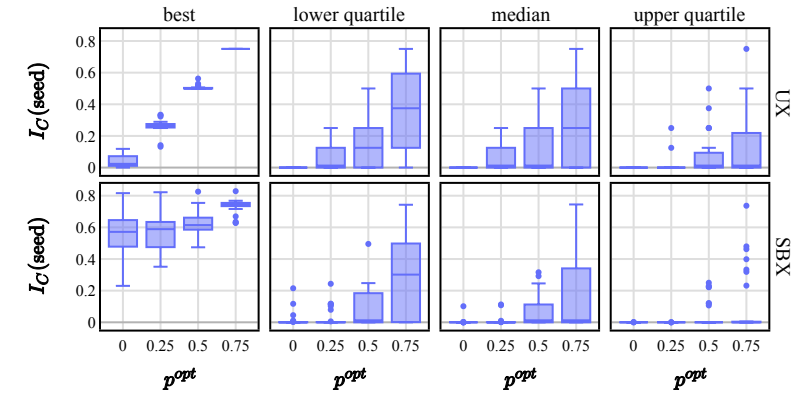


Figure B.14: Seed influence for the Rastrigin function ($\mathfrak{z} = 8$ "close" seed).

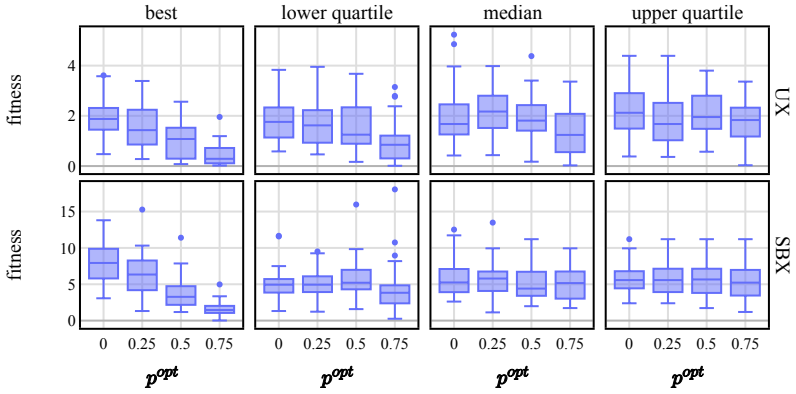


Figure B.15: Performance on the Rastrigin function ($\mathfrak{z} = 8$, "far" seed).

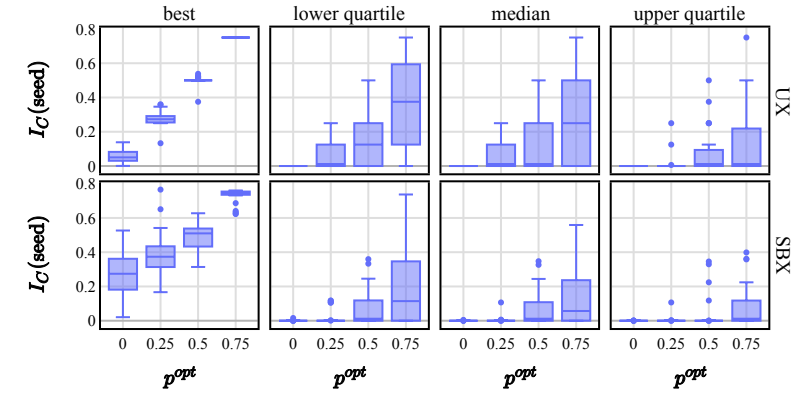


Figure B.16: Seed influence for the Rastrigin function ($\mathfrak{z} = 8$ "far" seed).

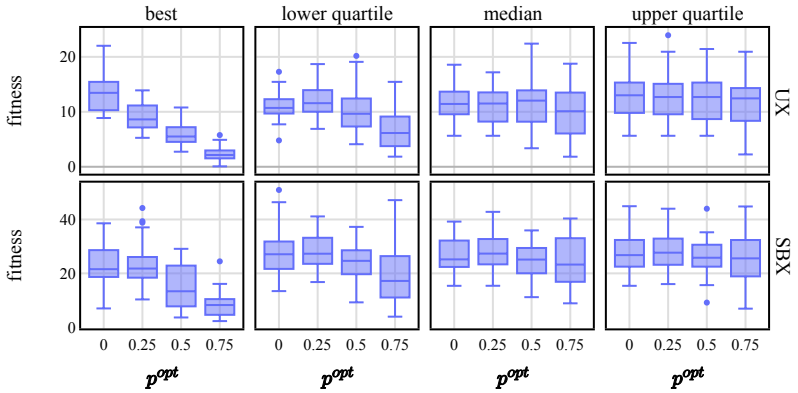


Figure B.17: Performance on the Rastrigin function ($\mathfrak{z}$ = 16, "close" seed).

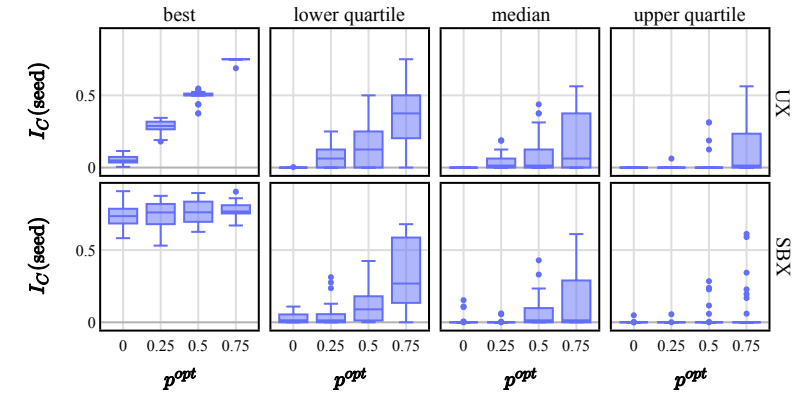


Figure B.18: Seed influence for the Rastrigin function ($\mathfrak{z}$ = 16 "close" seed).

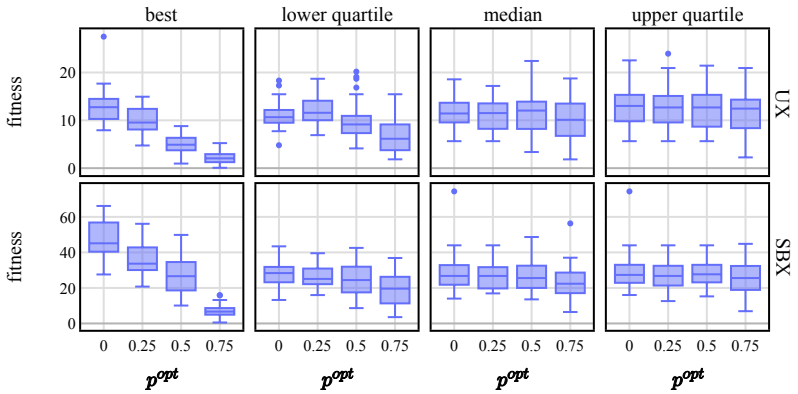


Figure B.19: Performance on the Rastrigin function ($\mathfrak{z}$ = 16, "far" seed).

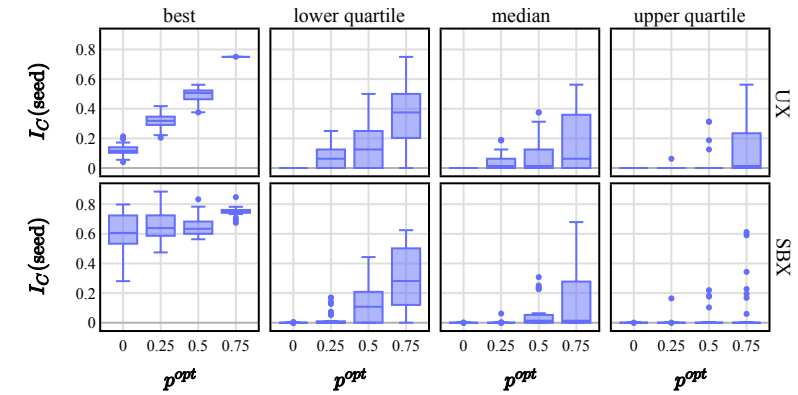


Figure B.20: Seed influence for the Rastrigin function ($\mathfrak{z}$ = 16 "far" seed).

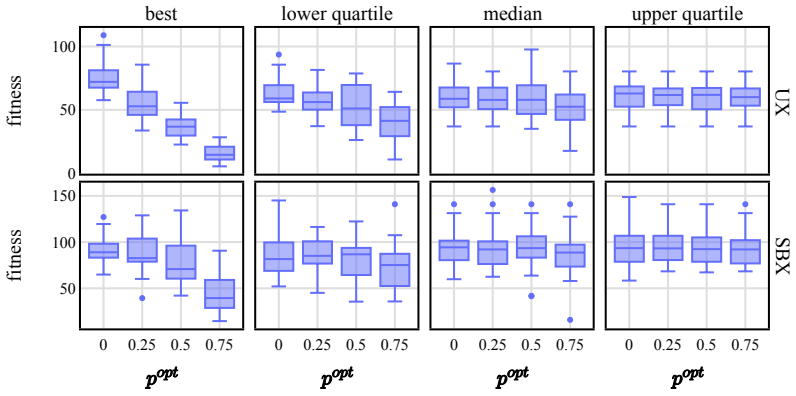


Figure B.21: Performance on the Rastrigin function ($\mathfrak{z}$ = 32, "close" seed).

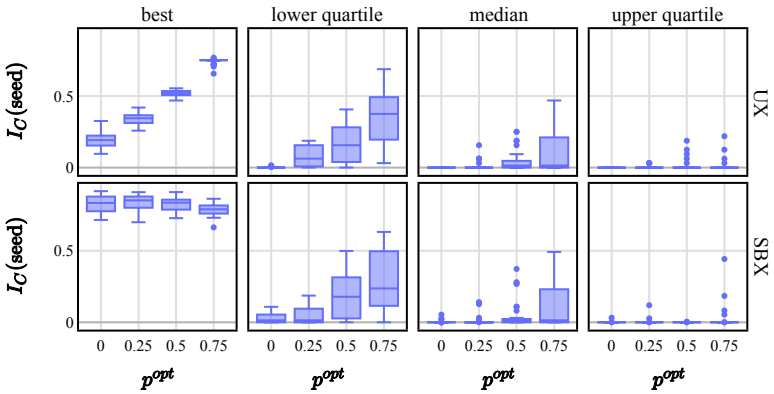


Figure B.22: Seed influence for the Rastrigin function ($\mathfrak{z}$ = 32 "close" seed).

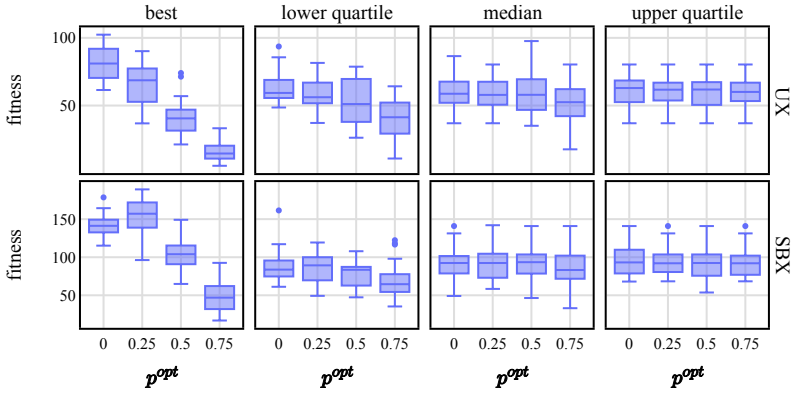


Figure B.23: Performance on the Rastrigin function ($\mathfrak{z}$ = 32, "far" seed).

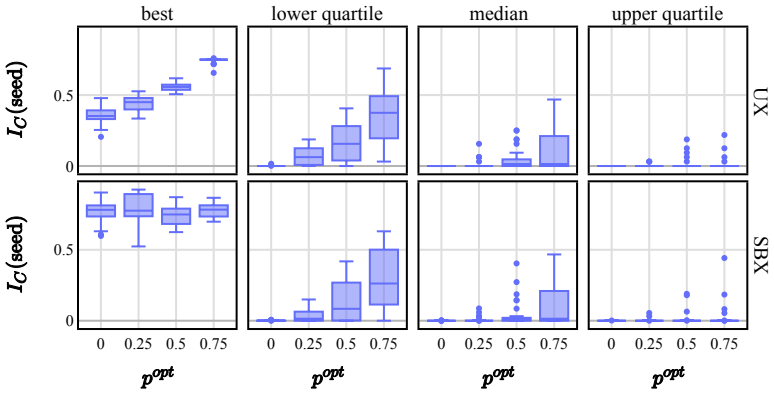


Figure B.24: Seed influence for the Rastrigin function ($\mathfrak{z}$ = 32 "far" seed).

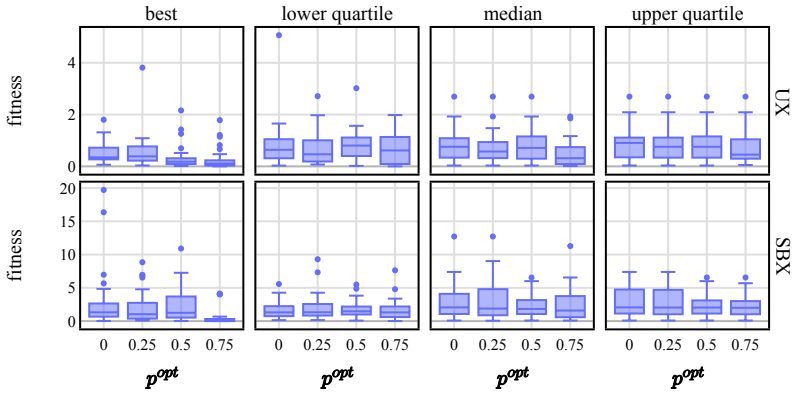


Figure B.25: Performance on the Rosenbrock function ($\mathfrak{z} = 4$, "close" seed).

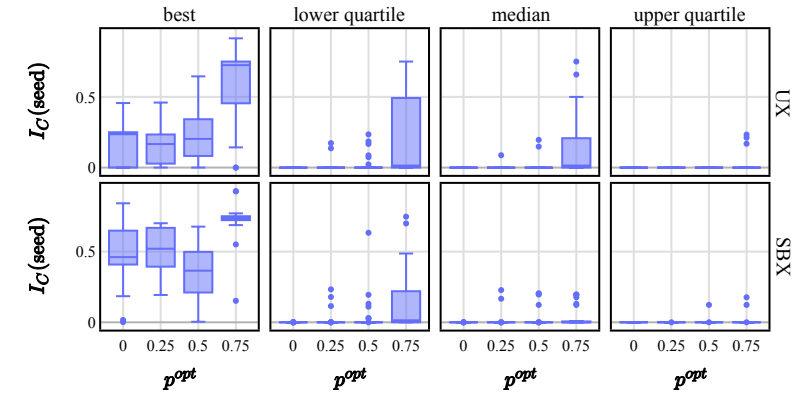


Figure B.26: Seed influence for the Rosenbrock function ($\mathfrak{z} = 4$ "close" seed).

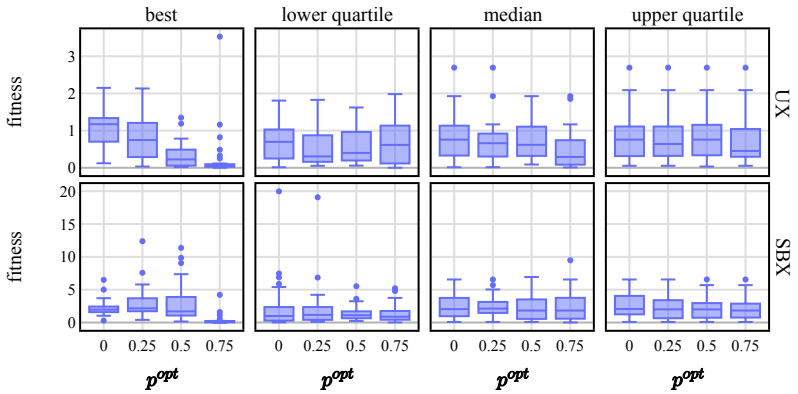


Figure B.27: Performance on the Rosenbrock function ($\mathfrak{z} = 4$, "far" seed).

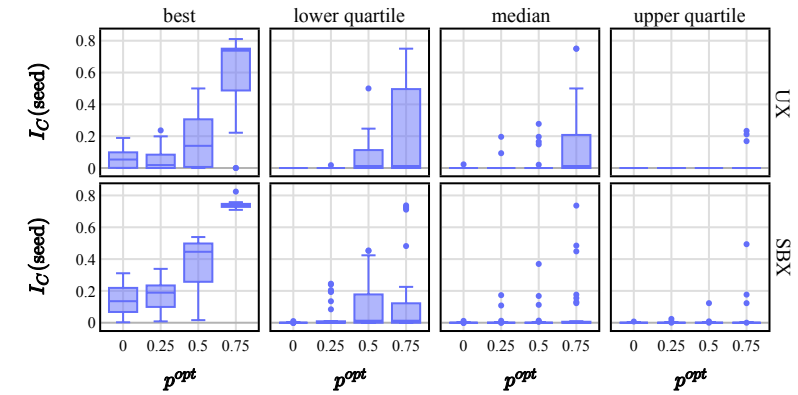


Figure B.28: Seed influence for the Rosenbrock function ($\mathfrak{z} = 4$ "far" seed).

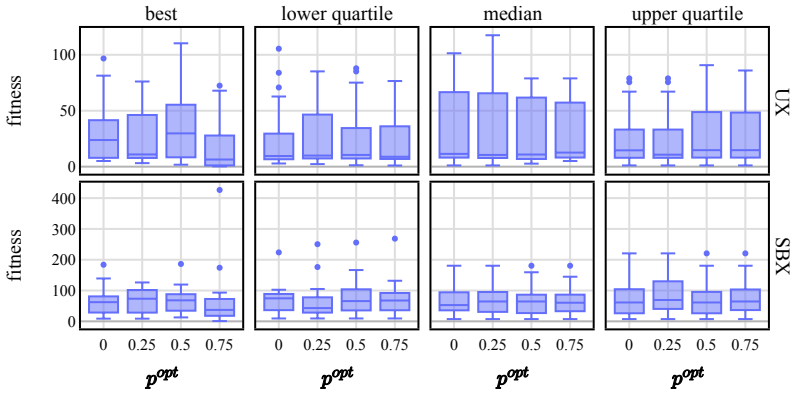


Figure B.29: Performance on the Rosenbrock function ($\flat = 8$, "close" seed).

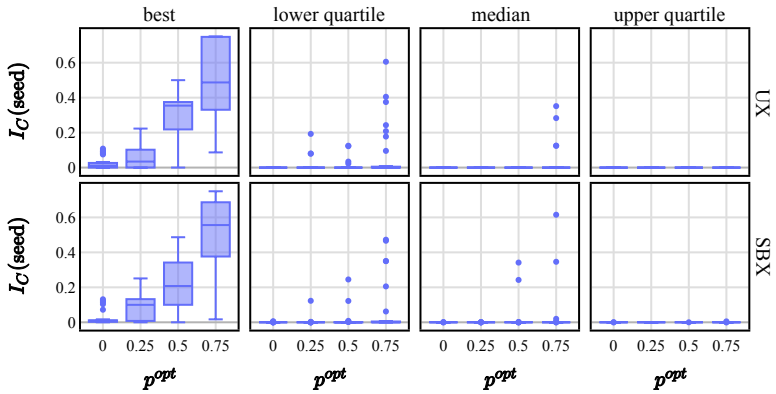


Figure B.30: Seed influence for the Rosenbrock function ($\flat = 8$ "close" seed).

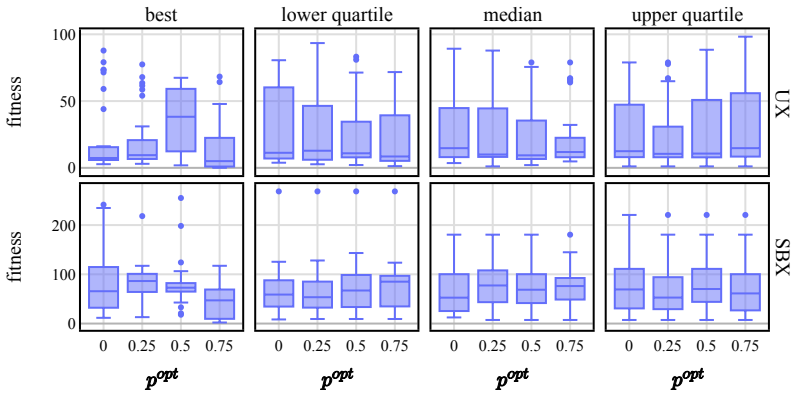


Figure B.31: Performance on the Rosenbrock function ($\flat = 8$, "far" seed).

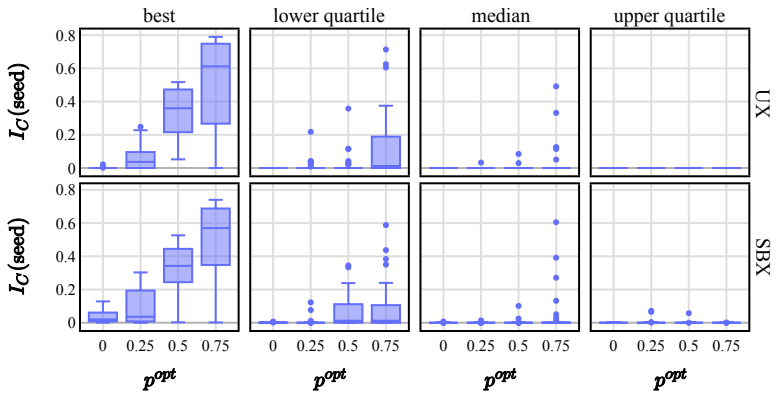


Figure B.32: Seed influence for the Rosenbrock function ($\flat = 8$ "far" seed).

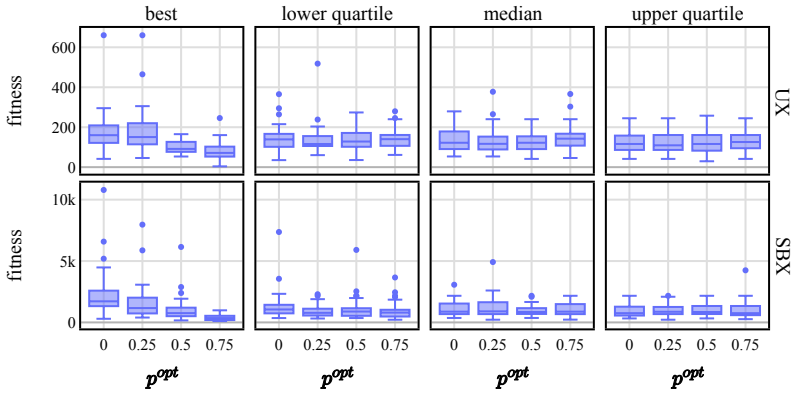


Figure B.33: Performance on the Rosenbrock function ($\mathfrak{z}$ = 16, "close" seed).

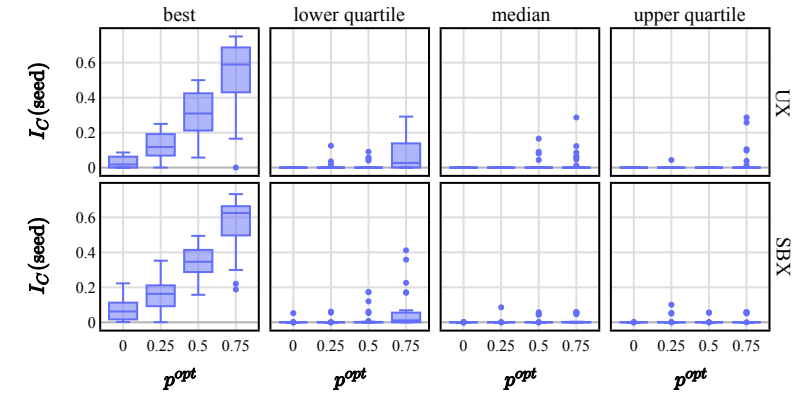


Figure B.34: Seed influence for the Rosenbrock function ($\mathfrak{z}$ = 16 "close" seed).

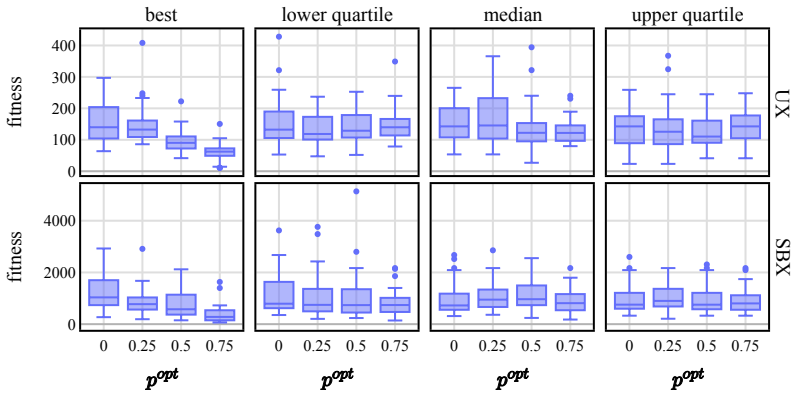


Figure B.35: Performance on the Rosenbrock function ($\mathfrak{z}$ = 16, "far" seed).

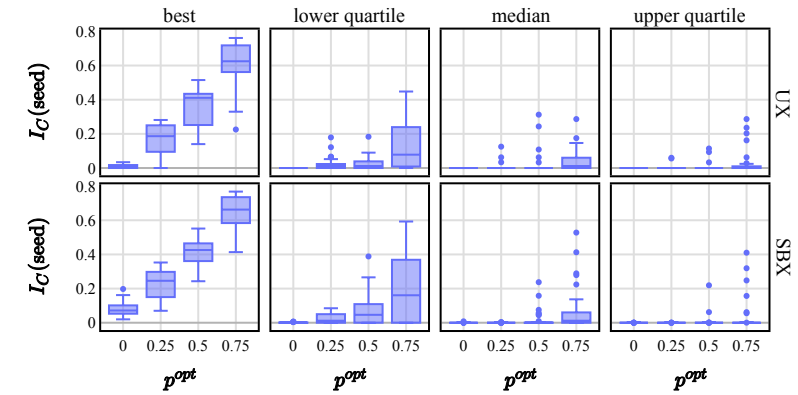


Figure B.36: Seed influence for the Rosenbrock function ($\mathfrak{z}$ = 16 "far" seed).

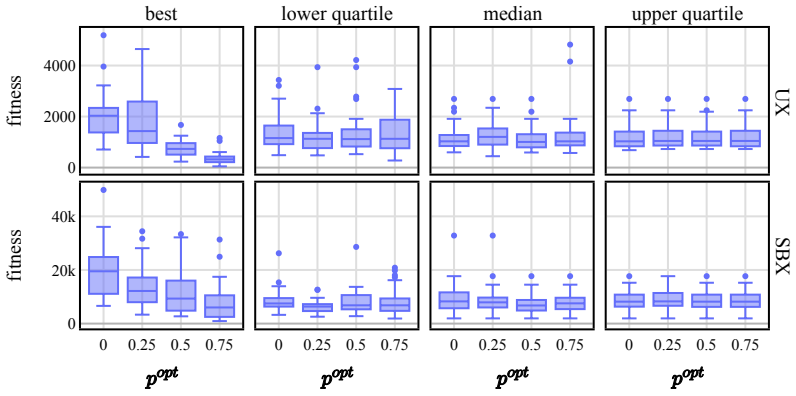


Figure B.37: Performance on the Rosenbrock function (δ = 32, "close" seed).

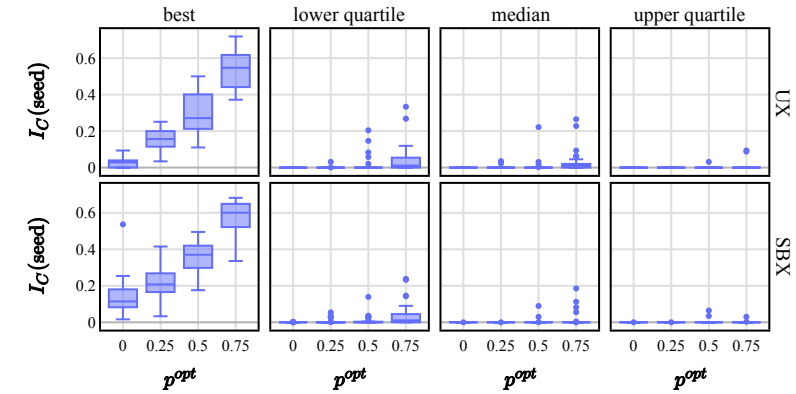


Figure B.38: Seed influence for the Rosenbrock function (δ = 32 "close" seed).

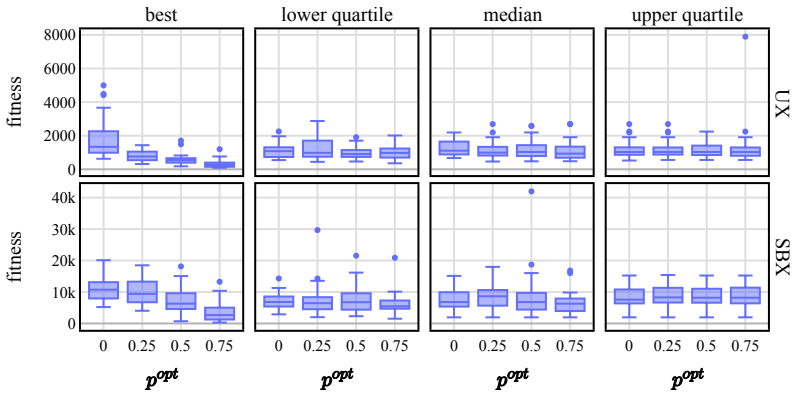


Figure B.39: Performance on the Rosenbrock function (δ = 32, "far" seed).

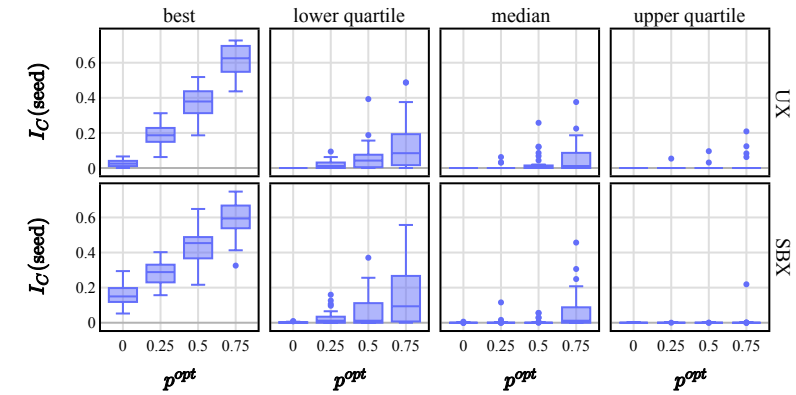


Figure B.40: Seed influence for the Rosenbrock function (δ = 32 "far" seed).