Keen: Kotlin Genetic Algorithms Framework

Thesis for the degrees of
Civil Engineer in Computer Science
and
Master of Science in Computing

Ignacio Slater Muñoz
Departamento de Ciencias de la Computación
Facultad de Ciencias Físicas y Matemáticas
Universidad de Chile.
Thesis Author

Nancy Hitschfeld. PhD.
Departamento de Ciencias de la Computación
Facultad de Ciencias Físicas y Matemáticas
Universidad de Chile
Thesis Supervisor

Alexandre Bergel. PhD.
Relational AI
Switzerland
Second Supervisor

Santiago, Chile
2023-08-11
`v0.5.2308-1`

This work, “Keen: Kotlin Genetic Algorithms Framework”, is a derivative of “Simple, Lightweight, and Adaptive TEX” by R8V., used under CC BY 4.0.

Abstract

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Contents

1 Introduction **7**
1.1 Motivation . . . 7
1.2 Hypothesis and Research Questions . . . 8
1.2.1 Hypothesis . . . 8
1.2.2 Research Questions . . . 8
1.3 Objectives . . . 8
1.3.1 Main Objective . . . 8
1.3.2 Specific Objectives . . . 8
1.4 Methodology . . . 9
1.5 Thesis Structure . . . 9

2 Theoretical Framework **11**
2.1 Evolutionary Algorithms . . . 11
2.2 Genetic Algorithms . . . 12
2.2.1 Representation and Evaluation . . . 12
2.2.2 Initialization . . . 13
2.2.3 Selection . . . 14
2.2.4 Variation . . . 15
2.2.5 Termination . . . 19
2.3 Genetic Programming . . . 20
2.3.1 Representation and Evaluation . . . 21
2.3.2 Initialization . . . 26
2.3.3 Selection . . . 29
2.3.4 Variation . . . 29

3 Relevant Work (State of the Art) **35**
3.1 The One Max Problem . . . 35
3.2 Agile Artificial Intelligence in Pharo . . . 36
3.3 Distributed Evolutionary Algorithms in *Python* (DEAP) . . . 37
3.4 Jenetics: Java Genetic Algorithm Library . . . 39
3.5 ECJ: A Java-based Evolutionary Computation Research System . . . 41
3.6 GeneticSharp: A .NET-based Genetic Algorithm Framework . . . 45
3.7 Other Libraries . . . 47

4 The Keen Framework **49**
4.1 Introduction . . . 49
4.2 Architecture . . . 49
4.3 Genetic Algorithms . . . 51
4.3.1 Genetic Material . . . 51
4.3.2 Evolution Engine . . . 52
4.3.3 Evolution Listeners . . . 54
4.4 Genetic Operators . . . 55

4.4.1 Word Guessing Problem . . . 55
4.4.2 Selection . . . 55
4.4.3 Mutation . . . 58
4.4.4 Crossover . . . 60
4.5 Genetic Programming . . . 64
4.5.1 Primitive Set . . . 64
4.5.2 Genetic Operators . . . 65
4.6 Parallelism . . . 66
4.7 Extensibility . . . 67
4.8 Conclusion . . . 68

5 Case Study: Real Function Optimization **69**
5.1 Introduction . . . 69
5.2 Problem Description . . . 69
5.3 Solution . . . 70
5.4 Results . . . 70
5.5 Conclusion . . . 71

6 Case Study: Knapsack Problem **73**
6.1 Introduction . . . 73
6.2 Problem Description . . . 73
6.2.1 0-1 Knapsack Problem . . . 73
6.2.2 Unbounded Knapsack Problem . . . 74
6.3 Solution . . . 74
6.4 Results . . . 75
6.5 Conclusion . . . 75

7 Case Study: Crash Reproduction **77**
7.1 Introduction . . . 77
7.2 Problem Description . . . 78
7.3 Solution . . . 78
7.4 Results . . . 79
7.5 Conclusion . . . 79

8 Conclusions **81**
8.1 Summary . . . 81
8.2 Contributions . . . 81
8.3 Future Work . . . 82
8.3.1 Genetic Algorithms . . . 82
8.3.2 Genetic Programming . . . 83
8.3.3 Multi-Objective Evolution . . . 84
8.3.4 Neuroevolution . . . 85
8.3.5 Coevolution . . . 85
8.3.6 Evolution Strategy . . . 86
8.3.7 Other evolutionary algorithms . . . 87
8.3.8 Crash Reproduction . . . 87

A Glossary **89**

B Test Functions for Optimization **93**
B.1 Ackley Function . . . 93
B.2 Beale Function . . . 94
B.3 Booth Function . . . 94
B.4 Bukin Function N.6 . . . 94
B.5 The Cross-in-Tray Function . . . 95

B.6 Easom Function . . . 96
B.7 Eggholder Function . . . 96
B.8 Goldstein-Price Function . . . 97
B.9 Himmelblau's Function . . . 98
B.10 Hölder Table Function . . . 99
B.11 Lévi Function N.13 . . . 99
B.12 Matyas Function . . . 100
B.13 McCormick Function . . . 101
B.14 Rastrigin Function . . . 101
B.15 The Rosenbrock Function . . . 103
B.16 Schaffer Function N.2 . . . 103
B.17 Schaffer Function N.4 . . . 104
B.18 The Sphere Function . . . 106
B.19 Styblinski-Tang function . . . 106
B.20 Three-Hump Camel Function . . . 107

C Additional Listings **109**

Chapter 1

Introduction

1.1 Motivation

The inception of artificial intelligence (AI) as we perceive it today is deeply rooted in the mid-20th century when Alan Turing proposed the intriguing question, "Can machines think?" [2]. This seminal query catalyzed a remarkable evolution in the field, leading to the emergence of a diverse spectrum of AI applications that persistently extend and advance.

Among the early endeavors to emulate evolution artificially, Nills Aall Barricelli's initiative in 1954 stands out. Barricelli construed evolution as a purely statistical process, setting a novel precedent for the field of evolutionary computation (EC) [4]. This groundbreaking exploration established the parameters for an evolutionary system, which necessitated the system's components to exhibit capabilities of reproduction and mutation, thereby facilitating evolution through natural selection or the survival of the fittest.

Expanding upon these foundational principles, John Holland's influential book, ***Adaptation in Natural and Artificial Systems*** [6], unveiled genetic algorithms (GA). This innovative methodology for addressing optimization problems drew inspiration from natural selection and genetic mechanisms. Holland's work solidified the field of evolutionary algorithms (EA), igniting the development of a plethora of techniques and algorithms, including but not limited to, genetic algorithms, genetic programming, and evolutionary strategies.

Despite the proven efficacy of evolutionary algorithms in resolving intricate optimization problems ranging from scheduling and data mining to machine learning and beyond, their execution can be challenging, frequently demanding extensive, repetitive coding. As the uptake of AI continues to soar, the requirement for streamlined, intuitive libraries and frameworks that empower efficient experimentation and application of these potent algorithms grows concomitantly.

The *Kotlin* programming language emerges as a promising platform for developing such a framework [51]. *Kotlin* distinguishes itself with its lucid syntax, static typing, and flawless interoperability with Java, making it an attractive choice for AI and EC enthusiasts [40, 41, 52]. The significance of *Kotlin* has been formally acknowledged in *Android* app development [39], and it continues to gain popularity among developers, with over 60% of *Android* developers now using it [38]. Furthermore, according to a *GitHub* study, *Kotlin* emerged as the ninth fastest-growing language worldwide from 2021 to 2023, overtaking *Python* by 0.4% [58].

The allure of *Kotlin* is further amplified by its multiplatform feature, which permits code sharing across a wide range of platforms, including JVM, *JavaScript*, *Android*, *iOS*, and native desktop applications. This cross-platform compatibility promotes enhanced productivity and consistency in software development.

Kotlin also supports functional programming (FP) and coroutines, which simplify the development of asynchronous and concurrent applications. This functionality proves particularly beneficial in the realm of evolutionary algorithms that often necessitate the execution of numerous concurrent processes. Moreover, *Kotlin*'s capabilities align well with the design of *Domain Specific Languages* (DSLs) [59], which can be used to concisely and intuitively define the problem domain.

Given these compelling features, this thesis aims to exploit the prowess of *Kotlin* to architect and implement a pioneering genetic algorithms framework, thereby augmenting the ever-growing application of *Kotlin* within the AI community. We expect this

framework to stimulate research and applications in the field, hastening the creation of novel algorithms and techniques.

1.2 Hypothesis and Research Questions

1.2.1 Hypothesis

The *Kotlin* programming language, known for its expressive syntax, simplicity, and cross-platform capabilities, presents an opportunity for enhancing the field of evolutionary algorithms. We hypothesize that the development of an efficient and user-friendly genetic algorithms framework in *Kotlin* could significantly streamline the implementation process of these algorithms, thereby catalyzing innovation and expediting research in this domain.

1.2.2 Research Questions

To assess the validity of our hypothesis and further investigate the potential benefits of the proposed *Kotlin*-based framework, we outline the following pivotal research questions:

1. What constitute the *fundamental requirements* for a versatile, efficient, and user-friendly genetic algorithm framework?
2. How can the *unique language features* of *Kotlin* be leveraged to fulfill these requirements and facilitate the development and use of genetic algorithms?
3. How does the performance and *ease of use* of a *Kotlin*-based genetic algorithms framework compare to that of existing solutions, particularly in terms of syntactical complexity?
4. In what ways can the proposed *Kotlin*-based framework be applied to *address complex real-world problems*, thereby demonstrating its practical viability and potential to contribute to advancements in AI and evolutionary computation?
5. What potential opportunities exist for the *enhancement and expansion* of the proposed *Kotlin*-based genetic algorithm framework in the future?

These research questions serve as guideposts for our exploration into the feasibility and potential impact of using *Kotlin* to develop a novel genetic algorithms framework. Each question targets a specific facet of the project, offering a holistic understanding of the technical intricacies, practical applications, and future prospects of our proposed framework.

1.3 Objectives

The primary objectives of this thesis are set forth to establish a solid foundation for the design, implementation, and evaluation of a novel, efficient, and user-friendly genetic algorithms framework using the *Kotlin* programming language.

1.3.1 Main Objective

The main objective of this study is to develop a *Kotlin*-based genetic algorithms framework that is versatile, efficient, and user-friendly. The framework should be able to support classical GAs, as seen in section 2.2 on page 12, and be easily extensible to support future enhancements and expansions such as new algorithms, like GP, and genetic operators.

1.3.2 Specific Objectives

Building upon the primary objectives, the following specific objectives have been established to guide this research:

1. **Language Feature Utilization:** Examine and utilize *Kotlin*'s unique language features, such as its expressive syntax, static typing, and its ease of developing DSLs, to design and implement a genetic algorithms framework. This exploration should enhance the design and implementation of the genetic algorithms framework, ensuring it is efficient, robust, and user-friendly.
2. **Framework Efficiency:** Design the genetic algorithms framework to optimize computational efficiency. This should involve algorithmic improvements, effective use of *Kotlin*'s language features, and careful resource management.

3. **User-friendliness:** Ensure the framework is accessible and user-friendly. It should present a clear, intuitive API for users to define and manipulate genetic algorithms, easing the learning curve and fostering adoption among AI and EC practitioners.

4. **Comparative Study:** Conduct comparative studies between the *Kotlin*-based genetic algorithms framework and existing solutions, particularly in terms of syntactical complexity, and ease of use. This should provide a benchmark to evaluate the advantages and potential areas of improvement for our framework.

5. **Real-world Applications:** Apply the proposed *Kotlin*-based framework to address complex real-world problems. This will not only demonstrate the practical viability of the framework but also its potential to contribute to advancements in AI and EC.

6. **Future Enhancements and Expansion:** Anticipate and propose potential enhancements and extensions for the *Kotlin*-based genetic algorithms framework. Discuss its adaptability to incorporate future advancements in AI and EC, ensuring its sustainability and continued relevance.

These specific objectives aim to add more granularity to our research focus, ensuring a comprehensive and detailed exploration of the possibilities offered by *Kotlin* for the development of a novel genetic algorithms framework.

1.4 Methodology

Our research methodology employs a multi-faceted approach comprised of distinct phases, each devised to address a specific objective identified in section 1.3 on the preceding page. This approach maintains a balance between theoretical exploration and practical application, with a consistent emphasis on rigorous analysis and evaluation.

1. **Literature Review and Language Exploration:** We initiate our research with an exhaustive study and analysis of the current literature pertaining to genetic algorithm frameworks, thereby discerning their respective merits and shortcomings. This analysis will serve as a cornerstone for defining the quintessential attributes for a versatile, efficient, and user-friendly genetic algorithms framework. Concurrently, we will conduct an in-depth investigation of the unique language features offered by *Kotlin*, discerning their potential contributions to the design and development of the proposed framework.

2. **Framework Design and Development:** Informed by the comprehensive literature review and language exploration, we will transition to the design and implementation phase of the *Kotlin*-based genetic algorithms framework. The design strategy will involve the thoughtful employment of *Kotlin*'s unique features such as its expressive syntax, static typing, and ease of crafting DSLs, thereby ensuring that the resulting framework is efficient, robust, and user-friendly.

3. **Application to Real-world Challenges:** Subsequently, we will utilize the developed *Kotlin*-based framework to address complex real-world problems. This stage will function as a testing ground to demonstrate the practical applicability, potential contributions, and viability of the framework in fostering advancements in the realms of AI and evolutionary computation.

4. **Future Adaptability and Expansion:** In the final phase of our research, we will deliberate upon prospective enhancements and extensions for the *Kotlin*-based genetic algorithms framework. Moreover, this phase will entail an assessment of the framework's adaptability and scalability to accommodate future advancements in AI and evolutionary computation, thereby assuring its long-term sustainability and relevance.

Throughout the implementation of our research methodology, we will uphold an analytical and critical perspective to discern the significance of our findings and their implications. We acknowledge and uphold the ethical principles of transparency and honesty in reporting our results and documenting the progress of our research.

1.5 Thesis Structure

1. **Background (chapter 2 on page 11):** This chapter offers a necessary foundation for understanding the central concepts utilized in this thesis. It encompasses a theoretical analysis of the relevant algorithms, setting the stage for the discussions and investigations that follow.

2. **State of the Art (chapter 3 on page 35):** Here, we present a comprehensive review of the current advancements in the domain of evolutionary computation. We particularly emphasize and examine the prevailing frameworks in this field, facilitating a comparative context for our research.

3. **Framework Design and Implementation (chapter 4 on page 49):** This chapter elucidates the architecture of our framework, detailing the design considerations and the subsequent implementation.

4. **Function Optimization Case Study (chapter 5 on page 69):** In this section, we put our framework to the test by applying it to a real-world function optimization problem. We utilize 20 classical benchmark functions to assess the framework's performance, with the functions elaborately displayed in appendix B on page 93.

5. **Knapsack Problem Case Study (chapter 6 on page 73):** This chapter features another practical application of the framework, in this case, tackling the Knapsack Problem. We test the framework against two different variants of this problem: the 0-1 Knapsack Problem and the Unbounded Knapsack Problem, evaluating its adaptability and robustness.

6. **Crash Reproduction Problem Case Study (chapter 7 on page 77):** In our third case study, we explore the application of the framework to a crash reproduction problem, showcasing its utility in a distinctly different domain.

7. **Conclusion and Future Work (chapter 8 on page 81):** In the concluding chapter, we summarize the breadth of work accomplished in this thesis and propose potential avenues for future investigations and enhancements based on our findings.

This structure facilitates a progressive narrative of our research, allowing for a coherent understanding of our objectives, methodology, and outcomes.

Chapter 2

Theoretical Framework

The objective of this chapter is to provide the reader with the theoretical background necessary to understand the rest of the document.

2.1 Evolutionary Algorithms

In the field of computational intelligence, ***evolutionary algorithms*** (EA) [22] are a family of algorithms inspired by the process of natural selection. They are part of the larger field of *evolutionary computation*,[1] which is a subfield of *metaheuristics*.[2]

EAs are algorithms that perform optimization or learning tasks by evolving solutions to a given problem via emergent intelligence [10]. These tasks may range from function optimization to machine learning or game AI development. EAs have three main characteristics:

- **Population-based:** These algorithms work with a **population of solutions**, allowing them to explore the search space in *parallel*.
- **Fitness-oriented:** The solutions in the population are evaluated using a ***fitness function***, which is a **problem-dependent function** that assigns a value to each solution based on its quality. The goal of the algorithm is to find the solution with the highest[3] fitness.
- **Variation-driven:** The candidate solutions are modified using ***genetic operators***, such as mutation, crossover, and selection, to create new solutions. These operators are usually based on the biological processes of *mutation* and *recombination*.

Remark. *"Parallel" in this context means that the algorithm is exploring multiple points in the search space in the same generation (iteration). It should not be confused with parallel computing, which is a technique used to speed up the execution of the algorithm by running it on multiple processors.*
Even though EAs are parallel in nature, they can be run on a single processor. Nevertheless, this parallel nature makes them a good candidate for parallel computing; in this thesis we will explore the use of parallel computing to speed up the execution of certain stages of the algorithm.

While these principles serve as the foundation for most EAs, it's important to note that some variants may prioritize some principles over others, or introduce new principles. This diversity allows EAs to be adapted to a wide range of problems and scenarios.

[1] See definition A.6 on page 89
[2] See definition A.11 on page 90
[3] In some cases, the goal is to minimize the fitness function, in which case the algorithm will aim to find the solution with the lowest fitness.

2.2 Genetic Algorithms

Genetic Algorithms (GA)[4] [6, 7, 22, 23] are a type of EA where a ***population*** of ***individuals***[5] representing candidate solutions to an optimization problem evolves towards better solutions. Each individual is defined by its location in the search space, known as its ***genotype***[6], and its fitness value, computed by a ***fitness function***. At a high level, GA is an automatic method for problem-solving, starting from a *high-level statement* of the desired outcome, without needing the user to predefine the solution's form or structure.

The classical GA operates as follows:

Listing 2.1– Genetic algorithm

```
var population = initialize population // Creates a random population of individuals
evaluate(population) // Calculates the fitness of each individual
while (termination condition not met) { // Could be a pre-defined number of generations, a
                                        // desired fitness level, etc.
  val survivors = select survivors(population) // Selects individuals for the next generation
  val parents = select parents(population) // Selects a subset of individuals as parents
  val offspring = alter(parents)  // Applies genetic operators to offspring, creating
                                  // variations
  evaluate(offspring) // Calculates the fitness of each new individual
  population = survivors + offspring // Creates the next generation
}
return population.fittest // Returns the most fit individual
```

The exact implementation of each of these steps depends on the specific problem at hand. Factors such as the problem's complexity, the representation of individuals, or even the computational resources available, can greatly influence the choice of methods used for initialization, selection, alteration, and survivor selection.

2.2.1 Representation and Evaluation

One of the most important aspects of a GA is the representation of the individuals. The representation is the encoding of potential solutions to the problem into a form that can be manipulated by the algorithm. This defines the search space of the algorithm, and it is one of the main factors that determines the performance of the algorithm.

The most general representation of an individual is a matrix of genes[7] called the ***genotype***[8] of the individual, where each column of the matrix is called a ***chromosome***[9].

Definition 2.1 (Cardinality of the search space)**.** *The **cardinality of the search space** is the number of different individuals that can be represented by the encoding.*
Formally, given a vector of alphabets $(\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n)$, and a representation $\mathbf{G}$ with n chromosomes of length $(m_1, m_2, \ldots, m_n)$ where each chromosome is encoded using the alphabet $\mathcal{A}_i$, the cardinality of the search space S is defined as:

$$|S| = \prod_{i=1}^{n} |\mathcal{A}_i|^{m_i} \tag{2.1}$$

Note that this definition assumes that the chromosomes are independent, which may not be the same for all evolutionary algorithms.

[4] Also known as Simple Genetic Algorithms (SGA) [22], or Traditional Genetic Algorithms (TGA) [23].
[5] See definition A.10 on page 90.
[6] See definition A.9 on page 90
[7] See definition A.7 on page 90.
[8] See [60].
[9] See definition A.4 on page 89.

Remark. *In the original publication of the GA [6], the genotype was known as the **environment** (E) and the search space was defined as a class $\mathcal{E}$ of all possible environments.*

To illustrate this concept, consider the following problem: given a binary string of length n, find the string that has the most ones.[10] In this case, we can use a single column matrix $\mathbf{G}$ to represent the individual, where each gene $g_i \in \mathcal{A}$ represents the i-th bit of the string, where $\mathcal{A} = \{0, 1\}$ is the alphabet containing the two possible values of a bit.

Then,

$$|S| = \prod_{i=1}^{1} |\mathcal{A}|^n = 2^n$$

Knowing this, we can conclude that an exhaustive search of the search space would require evaluating 2^n individuals, and thus the algorithm would have a time complexity of $\mathcal{O}(2^n)$.

This is a very simple example, but we can see how a naive search algorithm would have a very high time complexity. This would be of the utmost importance in a real world problem, where the search space would be much larger.

With a representation defined, we can now define an evaluation method for the individuals, which is done using a ***fitness function***.

Definition 2.2 (Fitness function)**.** *A **fitness function** is a function $\phi : S \to \mathbb{R}^n$, where S is the search space and n is the number of objectives of the optimization problem, that takes a genotype as input and returns a vector of real numbers representing how close the individual is to the global optimum of each objective.*
The fitness function is usually defined by the user of the algorithm, and it is problem dependent.

Definition 2.3 (Batch fitness function)**.** *A **batch fitness function** $\Phi : \mathbb{P} \to \mathbb{R}^{m \times n}$ is a function that maps a population to a matrix of real numbers, where m is the number of individuals in the population and n is the number of objectives of the optimization problem.*

The one max problem is a maximization problem with a single objective, so the fitness function would be defined as follows:

$$\phi_{\mathbf{G}} = \sum_{i=1}^{n} g_i \tag{2.2}$$

In the ***Representation and Evaluation*** section, we examined the key aspect of a genetic algorithm (GA) — the representation of individuals, their encoding and search space. The performance of a GA significantly depends on how solutions are encoded to form individuals. We use the concepts of genotype, chromosome, and gene to describe the individual's representation. The cardinality of the search space, defined as the total number of different individuals that can be represented, is crucial as it impacts the algorithm's complexity. We introduced the "One Max" problem as an example, using a binary string representation. The fitness function, which evaluates individuals' fitness, plays a critical role in navigating the search for optimal solutions. In the "One Max" problem, the fitness function sums the binary string elements, representing the number of ones in the string.

2.2.2 Initialization

GA operates on a group of individuals called a ***population***. The algorithm designer must define the size of the population, and how to initialize it. The initialization process is usually random, but it can also be guided by some prior knowledge about the problem being solved. For example, if the problem is to find a solution to a maze, the population could be initialized with individuals that represent paths from the start to the end of the maze. This would speed up the search process, since the algorithm would not have to start from scratch.

[10] This is known as the ***One Max*** problem [53] or ***Ones Counting*** problem [60].

Once the population is initialized, the algorithm performs an evaluation of each individual in the population, and assigns a ***fitness value*** to each individual. This is done in an effort to learn something about the problem, and to guide the search process towards better solutions.

In the case of the ***One Max*** problem, there is no prior knowledge about the problem, so the population via a blind search of the search space (in other words, the initialization is random). This is done by generating a random binary string of length n for each individual in the population.

Let's assume that we have a population of size 4, and that the length of the binary strings is $n = 4$.

The initialization process could generate the following individuals:[11]

Generation 0

Individual	Binary string	Fitness
I_1	1100	2
I_2	0001	1
I_3	0000	0
I_4	0100	1

Table 2.1: Population of individuals in generation 0

	Fitness	Individual
Best	2	I_1
Worst	0	I_3
Average	1	
Standard deviation	0.817	

Table 2.2: Fitness of the individuals in generation 0

In the initialization phase of a genetic algorithm, we define and setup the population of individuals to be used in the search process. This population can be randomly generated or informed by some prior knowledge about the problem at hand. Each individual is evaluated to determine its fitness, guiding the algorithm's search for optimal solutions. In our "One Max" problem example, we initialized a population of four individuals with binary strings of length $n = 4$ and evaluated their fitness. This setup marks the beginning of the evolutionary process, setting the stage for the subsequent stages of selection (section 2.2.3), crossover (section 2.2.4 on page 16), and mutation (section 2.2.4 on page 18).

2.2.3 Selection

Once initialization is complete, the Genetic Algorithm (GA) enters its main loop, where the core evolutionary processes take place. In the GA, a mechanism that simulates natural selection operates, providing fitter individuals with higher chances of survival and breeding opportunities.

Suppose that we have a population P of N individuals, each with a fitness value f_i, where $i \in \{1, \ldots, N\}$. Let σ be the survival rate, a parameter controlling the degree of elitism. This is the proportion of individuals that will survive (unmodified) to the next generation. The GA will then select $\lfloor \sigma N \rfloor$ individuals to survive to the next generation, and $\lceil (1-\sigma)N \rceil$ individuals to be replaced by the offspring.[12]

[11] Since the nature of genetic algorithms is stochastic, the initialization process could generate different individuals each time the algorithm is run. For this example, we selected a specific set of individuals in a way that makes it easier to get a grasp of the algorithm.

[12] The sole purpose of employing both *floor* and *ceiling* functions is to guarantee that the total number of individuals chosen for survival and replacement equals N, which makes these functions interchangeable in this context.

Definition 2.4 (Selection operator)**.** *An operator used to select individuals from a population. Formally, a selection operator is a function*

$$\Sigma : \mathbb{P} \times \mathbb{N} \times \cdots \to \mathbb{P}; (P, n, \dots) \mapsto \Sigma(P, n, \dots)$$

where:

- $\mathbb{P}$ *is the set of populations;*
- $\mathbb{N}$ *is the set of positive natural numbers;*
- P *is a population;*
- n *is the number of individuals to select from* P*;*
- $\Sigma(P, n)$ *is the population of* n *individuals selected from* P*.*

The selection operator is typically implemented as a ***stochastic*** operator, introducing some randomness into the selection process.

As an illustration, consider a *roulette wheel* selection operator[13] applied to a population of four individuals with a survival rate of 0.25. In this selection scheme, each individual is assigned a selection probability proportional to its fitness value (assuming higher fitness is better). The selection probability of an individual i is calculated as follows:

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j} \tag{2.3}$$

In our example, the selection probabilities are detailed in table 2.3.

Individual	Fitness	Selection Probability
I_1	2	50%
I_2	1	25%
I_3	0	0%
I_4	1	25%

Table 2.3: Selection probabilities for the individuals in the example population.

The selection operator then selects individuals at random, each with a probability equal to their selection probability. Suppose I_2 is selected to survive to the next generation, then I_1, I_3, and I_4 will be replaced by the offspring.

This section has introduced the concept of selection in GAs, which will be explored further in section 4.4.2 on page 55. Next, we will delve into variation operators responsible for generating the offspring that will replace the individuals not selected to survive to the next generation.

2.2.4 Variation

Variation is the process of creating new individuals from existing ones in the pursuit of exploring the solution space. This is crucial in a Genetic Algorithm (GA) to avoid premature convergence to sub-optimal solutions. In a GA, variation is achieved by applying ***variation operators*** to the individuals in the population. The most common variation operators are ***crossover*** and ***mutation***, which will be explored in this section.

[13] See section 4.4.2 on page 56 for a detailed description of the roulette wheel selection operator.

Definition 2.5 (Variation operator)**.** *A variation operator is used to create new individuals from existing ones. Formally, it is a variadic function represented as*

$$\varphi : \mathbb{P} \times \mathbb{R} \times \cdots \to \mathbb{P};\ (P, \rho, \dots) \mapsto \varphi(P, \rho, \dots)$$

where:

- $\mathbb{P}$ *is the set of all possible populations,*
- $\mathbb{R}$ *is the set of real numbers,*
- P *is the population to be varied,*
- ρ *is the probability of applying the operator to an individual in the population.*

The additional arguments depend on the specific implementation of the variation operator. The role of these arguments will be clarified in section 4.4 on page 55.

Crossover

The variation operator in genetic algorithms often involves a procedure known as ***crossover***, which emulates the process of genetic recombination observed in nature.[14] This process involves the exchange of genetic material between two individuals to create a new generation.

Definition 2.6 (Crossover operator)**.** *A crossover operator is a variation operator that is used to create new individuals from existing ones by performing a recombination of their genetic material.*
Formally, it is a variadic function represented as

$$X : \mathbb{P} \times \mathbb{R} \times \cdots \to \mathbb{P}; (P, \rho, \dots) \mapsto X(P, \rho, \dots)$$

where:

- $\mathbb{P}$ *is the set of all possible populations,*
- $\mathbb{R}$ *is the set of real numbers,*
- P *is the population to be varied,*
- $\rho \in [0, 1]$ *is the probability of applying the operator to an individual in the population.*

For the problem under consideration, we utilize a simplified form of the ***single-point crossover***[15] operator. This operator selects the first half of the genes from two parent individuals and generates two new offspring by interchanging these selected genes.

For instance, consider two parent individuals selected via the ***roulette wheel*** selector described earlier: $I_1 = 1100$ and $I_2 = 0001$. The single-point crossover operator selects the first half of the genes from each parent, i.e., 11 from I_1 and 00 from I_2, and produces a pair of new chromosomes with the first half, and produces two new offspring by exchanging these selected parts: $O_1 = 1101$ and $O_2 = 0000$ (as illustrated in fig. 2.1 on the facing page).

Following another iteration of the single-point crossover operator, we can generate a result as shown in table 2.4 on the next page, leading to a new population $\mathbf{O} = \{(0000,\ 0),\ (1101,\ 3),\ (0101,\ 2)\}$.

If we now use these offspring as-is to create the next generation, we would obtain the population shown in table 2.5 on the facing page:

	Fitness	**Individual**
Best	3	O_2
Worst	0	O_1
Average	1.25	
Standard deviation	1.291	

Table 2.6: Fitness of the population after applying the single-point crossover operator. "Best" refers to the individual with the highest fitness, and "Worst" refers to the individual with the lowest fitness

[14] This is referred to as ***crossing-over*** in [6].
[15] See section 4.4.4 on page 63.

Figure 2.1: Single-point crossover

Generation 0 → Generation 1

I	$\Phi_{\mathbf{I}}$	**O**	$\Phi_{\mathbf{O}}$
$\begin{bmatrix}1100\\0001\end{bmatrix}$	$\begin{bmatrix}2\\1\end{bmatrix}$	$\begin{bmatrix}0000\\1101\end{bmatrix}$	$\begin{bmatrix}0\\3\end{bmatrix}$
$\begin{bmatrix}0001\\0100\end{bmatrix}$	$\begin{bmatrix}1\\1\end{bmatrix}$	$\begin{bmatrix}0101\\\cdot\end{bmatrix}$	$\begin{bmatrix}2\\\cdot\end{bmatrix}$

Table 2.4: Illustration of the single-point crossover operation. In this procedure, two parent individuals are selected and a cut point is chosen. Each offspring is then formed by combining the genes from the parents: one gets the genes from the first part of the first parent and the second part of the second parent, while the other gets the genes from the first part of the second parent and the second part of the first parent. Here, $\cdot$ represents a "discarded" value (since according to the survival rate, only three offspring need to be produced).

Generation 1

Individual	**Binary String**	**Fitness**
I_2	0001	1
O_1	0000	0
O_2	1101	3
O_3	0101	2

Table 2.5: Population after applying the single-point crossover operator. Note that I_2 is the survivor of the previous generation picked in section 2.2.3 on page 14.

As observed from table 2.6 on the preceding page, the average fitness of the population has increased from 1 to 1.25, and the fitness of the best individual has improved from 2 to 3. This improvement showcases how the crossover operator helps guide the search towards superior solutions.

While the crossover operation has indeed enhanced the average fitness of the population, to further augment genetic diversity within the population and prevent premature convergence to suboptimal solutions (local optima), the introduction of a ***mutation*** operator is often beneficial. This operation will be discussed in the next section.

Mutation

One limitation of the crossover operator is its reliance on existing genetic material in the population.

This constraint can lead to premature convergence, particularly for problems with numerous local optima such as the ***Rastrigin function*** optimization.[16]

To counteract this and introduce *diversity* in the population, we use the *mutation* operator.

This operator alters the genetic material of an individual within the population according to a specific probability.

Definition 2.7 (Mutation operator)**.** *A mutation operator is a function that alters the genetic material of individuals within a population based on a certain probability, thereby producing a new population.*
Formally, a mutation operator is a variadic function

$$M : \mathbb{P} \times \mathbb{R} \times \cdots \to \mathbb{P};\ (P,\ \mu,\ \ldots) \mapsto M(P,\ \mu,\ \ldots)$$

where:

- $\mathbb{P}$ *represents the set of all possible populations;*
- $\mathbb{R}$ *represents the set of real numbers;*
- P *is the current population;*
- μ *represents the mutation rate—the probability that an individual in the population will undergo mutation.*

The other arguments are specific to the mutation operator being used.

For instance, in the "One Max" problem, we can use a ***bit-flip*** mutation.[17] This operator scans each gene in an individual and substitutes it with its complement according to a predetermined probability.

Suppose we set the ***mutation rate*** $\mu = 1$, and apply the mutation operator to the population resulting from the crossover operation described in section 2.2.4 on page 16. As shown in **??** on page ??, the resulting population would be $\mathbf{O} = \{(1111,\ 4), (0010,\ 1), (1010,\ 3)\}$

Generation 0 → Generation 1

$\mathbf{I}$	$\Phi_{\mathbf{I}}$	$\mathbf{O}$	$\Phi_{\mathbf{O}}$
$\begin{bmatrix}0000\\1101\\0101\end{bmatrix}$	$\begin{bmatrix}0\\3\\2\end{bmatrix}$	$\begin{bmatrix}1111\\0010\\1010\end{bmatrix}$	$\begin{bmatrix}4\\1\\2\end{bmatrix}$

Table 2.7: Illustration of the ***bit-flip*** mutation operator applied to the population resulting from the crossover operation in section 2.2.4 on page 16.

If the mutated offspring were used to generate the next population, as shown in table 2.8, you can observe the increased diversity.

Generation 1

Individual	Binary String	Fitness
I_2	0001	1
O'_1	1111	4
O'_2	0010	1
O'_3	1010	2

Table 2.8: Population after applying the ***bit-flip*** mutation operator to the population resulting from the crossover operation in section 2.2.4 on page 16.

[16] See appendix B.14 on page 101.
[17] See section 4.4.3 on page 58.

	Fitness	**Individual**
Best	4	O'_1
Worst	0	(I_2, O'_2)
Average	2	
Standard deviation	1.414	

Table 2.9: Fitness of the population after applying the ***bit-flip*** mutation operator to the population resulting from the crossover operation in section 2.2.4 on page 16.

Clearly, the mutation operator has introduced diversity into the population. In the original population, no individual had a 1[18] in the third position. Therefore, the crossover operator could never produce an individual with a 1 in that position. But the mutation operator has introduced three individuals with a 1 in the third position.[19]

In conclusion, the mutation operator plays a crucial role in genetic algorithms by introducing diversity into the population and preventing premature convergence to local optima. It facilitates a more thorough exploration of the search space, allowing new and potentially beneficial traits to emerge. However, it's essential to note that a high mutation rate might disrupt advantageous traits, while a low rate might not sufficiently prevent premature convergence. The specific mutation operator and the mutation rate used are crucial factors in shaping the genetic algorithm's search process.

With this, the variation process is complete, and we can proceed to the next step of the genetic algorithm.

2.2.5 Termination

After each generation – when a new population is fully created – the genetic algorithm verifies if the termination criteria have been met. If so, the algorithm terminates and returns the best individual identified. Otherwise, the process continues to the next generation.

Consider a scenario where the termination criterion is defined as the discovery of an individual possessing the maximum number of ones, represented as 1111. This would correspond to a condition where $\phi_{\mathbf{G}} = 4$.

Recall that we found the individual 1111 after applying the variation operators to the population. As a result, the termination criterion is met and the genetic algorithm concludes its process.

It's worth noting that not all search space has been explored, as demonstrated in table 2.10. The algorithm's fitness-oriented search strategy means it performs a guided, rather than exhaustive, search. However, the increasing fitness of the population's individuals across generations indicates convergence towards an optimal solution.

	00	01	10	11
00				
01				
10				
11				

Table 2.10: Candidates from the search space that were explored by the genetic algorithm. Cells that are coloured in dark gray represent candidates that were explored by the genetic algorithm. Each individual is defined by the row and column that it occupies in the search space, where the row represents the first 2 bits of the individual and the column represents the last 2 bits of the individual; e.g. the individual 0001 is located in the first row and second column of the table.

For small search spaces like in our example, the distinction between this algorithm and a purely random search may seem minimal. But for larger search spaces, as explored later in this thesis, the difference becomes highly significant.

It's important to underline that genetic algorithms, being stochastic in nature, do not guarantee discovery of the optimal solution. Their effectiveness depends on various factors such as the fitness function, the representation scheme, the variation operators,

[18] $P = \{1100, 0001, 0000, 0100\}$
[19] $P' = \{0001, 1111, 0010, 1010\}$

and the selection strategy. These components and their impact on performance across different problems will be thoroughly examined in this thesis.

In summary, the termination phase of the genetic algorithm represents a crucial step in determining the overall process outcome. By utilizing a targeted termination criterion – such as the discovery of an individual with the highest possible fitness score – the algorithm effectively navigates the search space. While not exhaustive in its exploration, the algorithm uses a fitness-oriented strategy to guide its trajectory towards an optimal solution. It's essential to recognize the inherent limitations of genetic algorithms due to their stochastic nature. Despite these, their potential to outperform random searches, especially in large search spaces, is considerable. However, success relies heavily on choosing appropriate parameters and procedures, a topic to be explored in-depth in subsequent sections of this thesis.

2.3 Genetic Programming

Genetic Programming (GP) [7, 9, 17, 22] is a specialized branch of Evolutionary Algorithms (EA) which focuses on evolving a population of computer programs to solve a given problem. One can perceive GP as an extension of Genetic Algorithms (GA), the key distinction being the problem each approach solves: GA optimizes parameters to enhance a given function, whilst GP induces programs.[20]

Despite these differences, GP and GA share various characteristics such as the utilization of a population of individuals, the employment of a fitness function to evaluate the individuals, and the application of genetic operators to generate new individuals. Notwithstanding, GP adopts a unique representation for the individuals and unique genetic operators.

Remark. *Although GP operates a fitness-guided search in the space of computer programs, it can be deemed as an optimization problem, akin to GA.*

Each individual in a GP population embodies a computer program composed of a set of primitives, referred to as ***functions*** and ***terminals***. An intuitive way to comprehend primitives is by visualizing a composite pattern where the functions equate to composite objects and the terminals to leaf objects (refer to fig. 2.2). An ***abstract syntax tree*** (AST) is an example of a program representation where the functions correspond to the internal nodes, and the terminals to the leaf nodes.

Figure 2.2: The composite structure of a GP individual, illustrating the relationship between functions and terminals

The general GP algorithm is represented in algorithm 1, which closely resembles the GA algorithm with the main disparity being the nature of the *genetic operators* in GP.

Algorithm 1 Outline of the Genetic Programming algorithm, showcasing its structural similarities with the Genetic Algorithm

```
Generate an initial population by recursively building random programs.
Execute each program and assign a fitness value to it.
repeat
    parents ← selectParents(population)                    ▷ Parent selection for reproduction
    alter(offspring)                                       ▷ Apply genetic operators to offspring
    population ← selectSurvivors(population, offspring)    ▷ Survivor selection to form next generation
until termination condition is met  ▷ Termination can be a fixed number of generations, or a satisfactory fitness level, etc.
return fittest(population)                                 ▷ Return the best solution found
```

[20] For a formal definition of program induction, refer to definition A.16 on page 90.

In the ensuing sections, we will delve into the fundamental components of a GP algorithm and elucidate them through an example.

2.3.1 Representation and Evaluation

Representation

As with GAs, the representation of the individuals is one of the most important aspects of a GP. The representation is the encoding of potential solutions to the problem into a form that can be manipulated,[21] executed and evaluated by the algorithm.

Various methods exist for program representation, such as utilizing an abstract syntax tree, a linear sequence of instructions, a stack of instructions, or a combination of these approaches. However, the most common representation is the ***tree representation***, where the program is represented as a composite data structure like the one shown in the introduction to this section.

Let's illustrate this with an example problem: given a set of n points in the plane, find the curve that best fits the points. This is a very common problem in statistics, and it is known as the ***symbolic regression*** problem [7]. In this example, our goal is to use symbolic regression to approximate the function

$$f(x) = 5x^3 - 2x^2 + \sin(x) - 7;\ x \in [-1, 1] \tag{2.4}$$

using this function, we can generate a set of points that lies on the curve as shown in fig. 2.3 and table 2.11 on the following page.

The next step in preparing our GP setup is to define the primitive set, which includes the functions and terminals that the algorithm can use to construct candidate solutions. In this case, we will use the following set of functions and terminals:

- Functions: 1. $+$ (Addition) 2. $-$ (Subtraction) 3. $\times$ (Multiplication) 4. $/$ (Division) 5. sin (Sine) 6. cos (Cosine) 7. pow (Power)
- Terminals: 1. x (The variable x) 2. $\{c \mid c \in [1,\ 7] \wedge c \in \mathbb{Z}\}$ (An ephemeral constant)[22]

Figure 2.3: A set of points generated from the function $5x^3 - 2x^2 + \sin(x) - 7$

Using this set of functions and terminals, we can represent the program as a tree, as shown in fig. 2.4 on the following page.

Note that this definition arises the possibility of having a program that has an infinite number of nodes, as the tree can grow indefinitely. This leads the search to be unsuccessful since the probability of finding a solution is close to zero, for example, the

[21] For example, by applying genetic operators.
[22] See definition A.5 on page 89.

x	y
0.889 160	-12.872 629
0.856 103	-12.358 361
0.821 295	-11.851 004
0.818 193	-11.807 452
0.429 859	-8.183 442
0.352 328	-7.812 033
0.035 776	-7.038 557
0.017 450	-6.983 134
0.529 010	-6.314 804
0.821 101	-4.848 557

Table 2.11: A set of points generated from the function $5x^3 - 2x^2 + \sin(x) - 7$

Figure 2.4: A possible tree representation of the program $5x^3 - 2x^2 + \sin(x) - 7$

probability of finding a solution on the initial population would be: $\lim_{x\to\infty} \frac{1}{x} = 0$. To avoid this issue of potentially infinite trees, we typically impose certain ***constraints*** on the generation of the trees. The most common constraints are the ***maximum height*** of the tree and the ***maximum number of nodes*** in the tree.

Search space

Using this representation, we can define the ***genotype*** of the individuals to contain only one ***chromosome*** which is composed of a single ***gene***[23] that is the tree representation of the program. Recalling the definition of cardinality presented in definition 2.1 on page 12, we can see that the cardinality of the search space will be the number of possible trees that can be generated using the primitive set and the maximum height of the tree.

Lemma 2.1. *Let $\mathbb{T}_H$ be the set of all possible **labeled trees** of height H, with $H \in \mathbb{N}$. Given the sets $\mathcal{T}$ and $\mathcal{F}$ corresponding to the possible labels of terminal nodes (nodes that do not have children) and the possible labels of internal nodes (nodes that have children) respectively, the number of trees in $\mathbb{T}_H$ is given by the following recurrence relation:*

[23] Although the most common representation is to have a single gene referencing the root of the tree, several variations that use multi-gene chromosomes have been proposed, such as Koza's *Automatically Defined Functions* [9], Angeline and Pollack's *Genetic Library Builder* [8, 10], or Rosca and Ballard's *Adaptive Representation* [11].

$$|\mathbb{T}_H(\mathcal{T},\,\mathcal{F})| = \begin{cases} |\mathcal{T}| & \textit{if } H = 0 \\ \sum_{f\in\mathcal{F}} |\mathbb{T}_{H-1}(\mathcal{T},\,\mathcal{F})|^{A(f)} & \textit{if } H > 0 \end{cases} \tag{2.5}$$

where $A(f)$ is the arity of the node f.

Proof. For the proof, we will use induction on the height of the tree. For the sake of brevity, we will use the notation $|\mathbb{T}_H(\mathcal{T},\,\mathcal{F})| = |\mathbb{T}_H|$.

Base case: $H = 0$ If the height of the tree is 0, then the tree is composed of a single node, which is a terminal node. Thus, the number of possible trees is equal to the number of possible terminal nodes, which arises:

$$|\mathbb{T}_0| = |\mathcal{T}|$$

Base case: $H = 1$ If the height of the tree is 1, then the tree is composed of a root node, which is an internal node, and a set of children, which are terminal nodes. Suppose that the root node has the label $f \in \mathcal{F}$ and arity $A(f)$.

Since the children are terminal nodes, each children can have any of the labels in $\mathcal{T}$. Thus, the number of trees rooted at f is equal to the number of possible combinations of $A(f)$ elements (with repetition and order) from the set $\mathcal{T}$, this is:

$$\prod_{i=1}^{A(f)} |\mathcal{T}| = |\mathcal{T}|^{A(f)}$$

Since the root node can have any of the labels in $\mathcal{F}$, the number of possible trees of height 1 is equal to:

$$|\mathbb{T}_1| = \sum_{f\in\mathcal{F}} |\mathcal{T}|^{A(f)}$$

Inductive step: $H > 1$ Suppose the statement holds true for $H = h$. We aim to prove that the statement also holds true for $H = h + 1$.

Since a terminal node cannot have children,[24] each tree of height $h + 1$ has a root with one of the labels from the set $\mathcal{F}$, and the remaining h layers are fully formed subtrees of height h.

For a given node label $f \in \mathcal{F}$ with arity $A(f)$, each child is the root of a subtree of height h. Given our inductive assumption, there are $|\mathbb{T}_h|$ possible such subtrees.

Since all subtrees are independent, the number of possible trees with the root f is $|\mathbb{T}_h|^{A(f)}$, which is the product of $|\mathbb{T}_h|$ over the arity of f.

We can sum this quantity over all $f \in \mathcal{F}$ to get the total number of possible trees of height $h + 1$:

$$|\mathbb{T}_{h+1}| = \sum_{f\in\mathcal{F}} |\mathbb{T}_h|^{A(f)}$$

□

Lemma 2.2. *Let $\mathbb{T}_{\leq H}$ be the set of all possible* ***labeled trees*** *of height $h \leq H$, with $H \in \mathbb{N}$ and $h \in \mathbb{N}$. Given the sets $\mathcal{T}$ and $\mathcal{F}$ corresponding to the possible labels of terminal nodes and the possible labels of internal nodes respectively, the number of trees in $\mathbb{T}_{\leq H}$ is given by the following recurrence relation:*

$$|\mathbb{T}_{\leq H}(\mathcal{T},\,\mathcal{F})| = \begin{cases} |\mathcal{T}| & \textit{if } H = 0 \\ |\mathbb{T}_H(\mathcal{T},\,\mathcal{F})| + |\mathbb{T}_{\leq H-1}(\mathcal{T},\,\mathcal{F})| & \textit{if } H > 0 \end{cases} \tag{2.6}$$

[24] This could also be interpreted as a terminal node having an arity of 0, or that all terminal nodes are leaves.

Where $\mathbb{T}_H$ is the set of all possible trees of height H.

Proof. For the sake of simplicity, we will use the notation $|\mathbb{T}_{\leq H}| = |\mathbb{T}_{\leq H}(\mathcal{T}, \mathcal{F})|$ and $|\mathbb{T}_H| = |\mathbb{T}_H(\mathcal{T}, \mathcal{F})|$.

The set $\mathbb{T}_{\leq H}$ can be partitioned into two disjoint sets: the set of all possible trees of height H and the set of all possible trees of height $h < H$. Thus we have:

$$|\mathbb{T}_{\leq H}| = |\mathbb{T}_H| + |\mathbb{T}_{\leq H-1}|$$

□

Theorem 2.1. *Let $\mathbb{T}_{\leq H}$ be the set of all possible **labeled trees** of height $h \leq H$, with $H \in \mathbb{N}$ and $h \in \mathbb{N}$. Given the sets $\mathcal{T}$ and $\mathcal{F}$ corresponding to the possible labels of terminal nodes and the possible labels of internal nodes respectively, the number of trees in $\mathbb{T}_{\leq H}$ is given by the following recurrence relation:*

$$|\mathbb{T}_{\leq H}(\mathcal{T}, \mathcal{F})| = \begin{cases} |\mathcal{T}| & \text{if } H = 0 \\ \left(\sum_{h=0}^{H-1} \sum_{f \in \mathcal{F}} |\mathbb{T}_h(\mathcal{T}, \mathcal{F})|^{A(f)} \right) + |\mathcal{T}| & \text{if } H > 0 \end{cases} \tag{2.7}$$

Where $\mathbb{T}_H$ is the set of all possible trees of height H and $A(f)$ is the arity of the node f.

Proof. From lemma 2.2 on the preceding page we know the number of trees of height H or less is given by:

$$|\mathbb{T}_{\leq H}| = |\mathbb{T}_H| + |\mathbb{T}_{\leq H-1}|$$

Then, by applying lemma 2.1 on page 22, we get:

$$|\mathbb{T}_{\leq H}| = \left(\sum_{f \in \mathcal{F}} |\mathbb{T}_{H-1}|^{A(f)} \right) + |\mathbb{T}_{\leq H-1}|$$

By unrolling the recurrence relation, we get:

$$\begin{aligned} |\mathbb{T}_{\leq H}| &= \left(\sum_{f \in \mathcal{F}} |\mathbb{T}_{H-1}|^{A(f)} \right) + |\mathbb{T}_{\leq H-1}| \\ &= \left(\sum_{f \in \mathcal{F}} |\mathbb{T}_{H-1}|^{A(f)} \right) + \left(\sum_{f \in \mathcal{F}} |\mathbb{T}_{H-2}|^{A(f)} \right) + \cdots + \left(\sum_{f \in \mathcal{F}} |\mathbb{T}_1|^{A(f)} \right) + \left(\sum_{f \in \mathcal{F}} |\mathbb{T}_0|^{A(f)} \right) + |\mathbb{T}_0| \\ &= \left(\sum_{h=0}^{H-1} \sum_{f \in \mathcal{F}} \mathbb{T}_h(\mathcal{T}, \mathcal{F})|^{A(f)} \right) + |\mathcal{T}| \end{aligned}$$

□

With this result, we can now calculate the cardinality of the search space of the genetic programming algorithm for a given maximum height H, since a program can be seen as a ***labeled tree***. Then, given the set $\mathcal{T} = \{x, c\}$ and the set $\mathcal{F} = \{+, -, \times, /, \text{sin}, \text{cos}, \text{exp}, \text{log}\}$, where

$$A(f) = \begin{cases} 2 & \text{if } f \in \{+, -, \times, /, \text{pow}\} \\ 1 & \text{if } f \in \{\text{sin}, \text{cos}\} \end{cases}$$

Given that the height of the AST of the target program is 4, we can use 5 as the maximum height of the programs in the search space (to allow a little of "breathing room" to the generated programs). Noting that, since c can be one of 7 possible values, $|\mathcal{T}| = 8$, we can calculate the cardinality of the search space as follows:

$$
\begin{aligned}
|\mathcal{S}| = |\mathbb{T}_{\leq 5}(\mathcal{T}, \mathcal{F})| &= \left(\sum_{h=0}^{4} \sum_{f \in \mathcal{F}} |\mathbb{T}_h(\mathcal{T}, \mathcal{F})|^{A(f)} \right) + 8 \\
&\approx 9.531\,142 \times 10^{37} \\
&\approx 10^{38}
\end{aligned}
$$

Thus, the search space of the genetic programming algorithm is of the order of 10^{38} programs.[25] It should be easy to see that the size of the search space make it unfeasible to perform an exhaustive search. This is why we need to use a heuristic search algorithm, such as genetic programming. In fig. 2.5 we can see how the number of trees of height less or equal to h rapidly increases as h increases.

Figure 2.5: Total number of trees of height less or equal to h for $h \in \{0, \ldots, 5\}$ and $\mathcal{T} = \{x, c\}$ and $\mathcal{F} = \{+, -, \times, /, \sin, \cos, \text{pow}\}$. Note that the Y axis is in logarithmic scale.

Evaluation

Now, we need to define how to evaluate the fitness of a program. Again, there are many ways to do this, but a common way is to use the ***mean squared error*** (MSE) between the points and the program.

Definition 2.8 (Mean Squared Error)**.** *If a vector of n predictions is generated from a sample of n data points on all variables, and $\mathbf{y}_i$ is the i-th observed value and $\hat{\mathbf{y}}_i$ is the i-th prediction, then the MSE of the predictor is a function* MSE : $\mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ *defined as:*

$$
\text{MSE}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{n} \sum_{i=1}^{n} (\mathbf{y}_i - \hat{\mathbf{y}}_i)^2 \tag{2.8}
$$

The MSE is a common measure of the quality of an estimator, used in many *machine learning* problems.

In our case, we will use the MSE to evaluate the fitness of a program. Consider a program P and two sets of points, $\mathbf{x}$ and $\mathbf{y}$, as outlined in table 2.11 on page 22. Suppose also that $\mathsf{P}[\mathbf{x}]$ is the set of points generated by evaluating P on the points of $\mathbf{x}$, and that $\mathsf{P}(x)$ is the result of evaluating P on the point x. Then, we can define the fitness of P as:

[25] This value was computed using the script shown in appendix C on page 109.

$$\phi_{\mathsf{P}} = \text{MSE}(\mathbf{y}, \mathsf{P}[\mathbf{x}]) = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{y}_i - \mathsf{P}(\mathbf{x}_i))^2 \tag{2.9}$$

This section elucidated the pivotal aspects of Genetic Programming (GP), focusing on representation of individuals and their evaluation. Individuals, potential solutions to a problem, can be encoded in several ways including tree representation. A common statistical problem, symbolic regression, was illustrated using a set of points on a curve from a function involving both polynomial and trigonometric elements. By defining a set of functions and terminals, these points were represented as a tree. The fitness of an individual program was assessed using the Mean Squared Error (MSE) between the points on the curve generated by the function and the points produced by the program. Thus, in the context of this problem, the fittest individual is the one that minimizes this error, i.e., best fits the curve. The methods described here underscore the versatility and applicability of GP to various types of problems. This method of representation and evaluation provides a robust basis for generating evolving populations of programs. In the next sections we will discuss the mechanisms for generating these populations and the evolutionary operators that act on them.

2.3.2 Initialization

As with other evolutionary algorithms, the algorithm starts by generating a population of random individuals.

There are many ways to generate random individuals, but a very common (and simple) method is the ***grow method*** [7]. In this method, a maximum height is defined, and the algorithm then creates a random tree with a given minimum height and a maximum height.

Remark. *A tree with a height of 0 is a tree with only one node, the root.*

The grow method then proceeds to recursively generate the trees with random nodes until a terminal node is selected or the maximum height is reached. This method is shown in algorithm 2.

Algorithm 2 The grow method for generating random trees

Require: $l \in \mathbb{N}, h \in \mathbb{N}$
Require: $l \leq h$
Require: a random integer n such that $l \leq n \leq h$
Require: **t** and **f** are the sets of terminal and function nodes respectively, where $\mathbf{t} \neq \emptyset$ and $\mathbf{f} \neq \emptyset$
Ensure: a random tree with a height between l and h

```
function grow(t, f, d)
    c ← ∅
    if d = n ∨ ( d ≥ l ∧ random() < |t| / (|t| + |f|) ) then
        return a random node from t
    else
        f ← a random node from f
        for i in 1 ... arity(f) do
            c_i ← grow(t, f, d + 1)
            c ← c ∪ {c_i}
        end for
        return a tree with root f and children c
    end if
end function
```

With this method, the algorithm can generate trees where the size of the longest path from the root to a leaf is a number $n \in [l, h]$.

Now that the algorithm can generate random trees, it can generate a random population of trees by generating a random tree for each individual in the population. Assuming a population size of $p = 4$, a maximum height of $h = 3$, a minimum

height of $l = 1$, and the primitives set defined in the previous section, the algorithm could generate the population: $\mathbf{P} = \{\mathbf{I}_1, \mathbf{I}_2, \mathbf{I}_3, \mathbf{I}_4\} = \left\{\frac{3}{\sin(2)} \times 5^3,\ 7 - (5 + \sin(x)),\ 7 + 2,\ 5x^2\right\}$, as shown in fig. 2.6.

(a) $\mathbf{I}_1 = \dfrac{3}{\sin(2)} \times 5^3$

(b) $\mathbf{I}_2 = 7 - (5 + \sin(x))$

(c) $\mathbf{I}_3 = 7 + 2$

(d) $\mathbf{I}_4 = 5x^2$

Figure 2.6: A population of random trees.

The next step is to calculate the fitness of each individual in the population. If we recall, the fitness function is the MSE between the expected output and the actual output of the individual.

$$
\begin{aligned}
\mathrm{MSE}(\mathbf{y}, \mathbf{I}_1[\mathbf{x}]) &= \frac{1}{10}\sum_{i=1}^{10}(\mathbf{y}_i - \mathbf{I}_1(\mathbf{x}_i))^2 = \frac{1}{10}\sum_{i=1}^{10}\left(\mathbf{y}_i - \frac{3}{\sin(2)} \cdot 5^3\right)^2 \\
&\approx 177\,596.851\,131 \\
\mathrm{MSE}(\mathbf{y}, \mathbf{I}_2[\mathbf{x}]) &= \frac{1}{10}\sum_{i=1}^{10}(\mathbf{y}_i - \mathbf{I}_2(\mathbf{x}_i))^2 = \frac{1}{10}\sum_{i=1}^{10}(\mathbf{y}_i - 7 - (5 + \sin(x_i)))^2 \\
&\approx 137.398\,836 \\
\mathrm{MSE}(\mathbf{y}, \mathbf{I}_3[\mathbf{x}]) &= \frac{1}{10}\sum_{i=1}^{10}(\mathbf{y}_i - \mathbf{I}_3(\mathbf{x}_i))^2 = \frac{1}{10}\sum_{i=1}^{10}(\mathbf{y}_i - (7 + 2))^2 \\
&\approx 331.924\,267 \\
\mathrm{MSE}(\mathbf{y}, \mathbf{I}_4[\mathbf{x}]) &= \frac{1}{10}\sum_{i=1}^{10}(\mathbf{y}_i - \mathbf{I}_4(\mathbf{x}_i))^2 = \frac{1}{10}\sum_{i=1}^{10}\left(\mathbf{y}_i - 5x^2\right)^2 \\
&\approx 138.079\,865
\end{aligned}
$$

With this, we can assign a fitness to each individual in the population as shown in table 2.12. A summary of the population's fitness is shown in table 2.13.

Generation 0

Individual	Program	Height	Fitness
$\mathbf{I}_1(x)$	$\frac{3}{\sin(2)} \cdot 5^3$	3	177 596.851 131
$\mathbf{I}_2(x)$	$7 - (5 + \sin(x))$	3	137.398 835
$\mathbf{I}_3(x)$	$7 + 2$	1	331.924 267
$\mathbf{I}_4(x)$	$5x^2$	2	138.079 865

Table 2.12: Initial population of the genetic programming algorithm

	Fitness	Individual
Best	137.398 835	$\mathbf{I}_2(x)$
Worst	177 596.851 131	$\mathbf{I}_1(x)$
Average	44 551.063 525	
Standard deviation	76 814.062 197	

Table 2.13: Fitness of the individuals in generation 0

We can observe that the worst individual has an error significantly larger than the best individual. This is to be expected, as the MSE is a measure of the error that escalates exponentially with the difference between the expected and actual output.

A graphical representation of the population is shown in fig. 2.7. It is clear from the figure that the worst individual is $\mathbf{I}_1$.

Figure 2.7: Graphical representation of the population in generation 0 compared to the expected output.

In summary, the initialization stage of the genetic programming algorithm, a population of random individuals, represented as trees, was generated using the "grow method". The height of these trees, signifying the length of the longest path from the root to a leaf, is determined randomly within a specified range. Each tree is recursively populated with random nodes until a terminal node is chosen or the maximum height is reached. Once a random population of trees is created, the fitness of each individual, based on the Mean Squared Error (MSE) between the expected and actual output, is calculated. This results in the assignment of fitness scores to all individuals, facilitating the evaluation and selection process in the succeeding stages of the

algorithm. The output of this process is a random population of trees with fitness scores and height measurements, ready for the next steps of selection, crossover, and mutation.

2.3.3 Selection

The selection process in Genetic Programming (GP) is similar to that of Genetic Algorithms (GA). However, unique modifications have been proposed for the standard process in GP, which consider the semantics, or the functional behavior, of the programs being evolved [25]. This work will not delve into this topic, as it is beyond the scope of this document.

For the problem under study, we use a selection process similar to the one used in the *One Max* problem (a simple optimization problem where the objective is to maximize the number of ones in a binary string). The equation for calculating the selection probability will diverge from eq. (2.3) on page 15, as it presumes the fittest individual is the one with the highest fitness value. In contrast, the symbolic regression problem, which is our focus, considers the individual with the lowest fitness value as the fittest.

For this particular case of the symbolic regression problem, we adjust our approach to selection. We introduce a ***corrected fitness function***, ϕ', defined as:

$$\phi'(I) = \left(\sum \Phi_{\mathbf{P}}\right) - \phi_I \tag{2.10}$$

Here, ϕ_I signifies the fitness of individual I, and $\Phi_{\mathbf{P}}$ represents the *batch fitness function* defined in definition 2.3 on page 13 applied to the population $\mathbf{P}$. We then define the selection probability for an individual $\mathbf{P}_i$ as:

$$p_i = \frac{\phi'(\mathbf{P}_i)}{\sum_{j=1}^{N} \phi'(\mathbf{P}_j)} \tag{2.11}$$

In this equation, N stands for the size of the population.

With this methodology, we calculate the selection probabilities for the population as illustrated in table 2.14. The outcome shows that the individual with the highest error has a considerably low probability of being selected, while the other individuals have roughly equal chances.

Individual	Fitness	Selection probability
$\mathbf{I}_1(x)$	607.402 968	0.113 615%
$\mathbf{I}_2(x)$	178 066.855 263	33.307 633%
$\mathbf{I}_3(x)$	177 872.329 832	33.271 246%
$\mathbf{I}_4(x)$	178 066.174 234	33.307 505%

Table 2.14: Selection probabilities for the symbolic regression problem.

Assuming a ***survival rate*** of 50%, let's consider that the selection process favors I_2 and I_3 as survivors due to their higher selection probabilities (as shown in the previous table). In this scenario, I_1 and I_4 are identified as the ones to be replaced by the offspring in the next generation. A comparison between the survivors and the target function is shown in fig. 2.8 on the next page.

To conclude, this section discussed the selection process in Genetic Programming (GP), noting that it largely mirrors that of Genetic Algorithms (GA) with some differences. These differences primarily focus on the semantics of the programs being evolved. The symbolic regression problem, necessitates a corrected fitness function and an adjusted selection probability equation. We outlined how these modifications are implemented and computed selection probabilities for a hypothetical population. A survival rate of 50% resulted in two individuals being selected as survivors and two being marked for replacement in the next generation. This lays the groundwork for the next phase of the process, which is variation.

2.3.4 Variation

Analogous to GA, the individuals in a GP population undergo a variation process. For the symbolic regression problem under consideration, we introduce variation using two specific operators: ***crossover*** and ***mutation***.

Figure 2.8: Comparison between the survivors and the target function.

Crossover

Variation operators in Genetic Programming (GP) must maintain the syntactic correctness of the programs or individuals. For crossover, this implies that the resultant offspring must be syntactically correct programs.

The crossover operator used in Genetic Algorithms (GAs), described in section 2.2.4 on page 16, is not typically suitable for GP, as it does not guarantee the syntactic correctness of the resulting individuals. Though there could be instances where the crossover operator used in GAs is applicable to GP, as depicted in chapter 7 on page 77, these are not common scenarios.

The choice of operator in GP depends on the representation of the individuals.

For tree-based GP, the fundamental crossover operator is the ***subtree crossover***, referenced in **??** on page ??.

This operator selects a random node from each parent and exchanges the subtrees rooted at these nodes. Usually, a constraint similar to the one used for generating the initial population is applied to this operator to prevent the creation of overly large trees.

Assuming we select two individuals, $\mathbf{I}_1$ and $\mathbf{I}_2$, from the population, the subtree crossover operator chooses a random node from each individual, say $\clubsuit = 7$ from $\mathbf{I}_1$ and $\diamondsuit = x^2$ from $\mathbf{I}_2$. The subtrees rooted at these nodes are then interchanged, as shown below:

$$\begin{aligned} \chi(\mathbf{I}_2,\, \mathbf{I}_4) &= \chi(\clubsuit - (5 + \sin(x)),\, 5\diamondsuit) \\ &= (\diamondsuit - (5 + \sin(x)),\, 5\clubsuit) \\ \Leftrightarrow \quad (\mathbf{O}_1,\, \mathbf{O}_2) &= (x^2 - (5 + \sin(x)),\, 5 \cdot 7) \end{aligned}$$

where χ signifies the subtree crossover operator between two individuals. The crossover of $\mathbf{I}_1$ and $\mathbf{I}_2$ is depicted in fig. 2.9 on the next page.

Following the application of the subtree crossover operator, the fitness of the individuals in the population is evaluated as shown in table 2.15 on the facing page. A summary of the population's fitness is given in table 2.16 on the next page.

Figure 2.9: Crossing over of $\mathbf{I}_2 = 7 - (5 + \sin(x))$ and $\mathbf{I}_4 = 5x^2$, producing $\mathbf{O}_1 = x^2 - (5 + \sin(x))$ and $\mathbf{O}_2 = 5 \cdot 7$.

Generation 1

Individual	Program	Fitness
$\mathbf{I}_2(x)$	$7 - (5 + \sin(x))$	137.398 835
$\mathbf{I}_3(x)$	$7 + 2$	331.924 267
$\mathbf{O}_1(x)$	$5 \cdot 7$	1 944.288 127
$\mathbf{O}_2(x)$	$x^2 - (5 + \sin(x))$	33.740 766

Table 2.15: Population after applying the subtree crossover operator.

	Fitness	Individual
Best	33.740 766	$\mathbf{O}_2(x)$
Worst	1 944.288 127	$\mathbf{O}_1(x)$
Average	611.837 999	
Standard deviation	776.701 997	

Table 2.16: Fitness summary of the population after applying the subtree crossover operator.

A notable improvement in the population's fitness is observed after the application of the subtree crossover operator. Comparing the results from table 2.13 on page 28, we find that the average fitness (or error) has dropped from 44 551.063 525 to 611.837 999, equating to an improvement of approximately 98.627%.

$$\frac{\bar{\Phi}_i - \bar{\Phi}_X}{\bar{\Phi}_i} = \frac{44\,551.063\,525 - 611.837\,999}{44\,551.063\,525} \approx 98.627\%$$

where $\bar{\Phi}_i$ is the average fitness of the population after initialization and $\bar{\Phi}_X$ is the average fitness of the population after applying the subtree crossover operator.

A reduction in the population's fitness standard deviation, from $76\,814.062\,197$ to $776.701\,997$, is also seen. This decrease of around 98.989% indicates that the population's diversity has reduced.

$$\frac{\sigma_i - \sigma_X}{\sigma_i} = \frac{76\,814.062\,197 - 776.701\,997}{76\,814.062\,197} \approx 98.989\%$$

where σ_i is the standard deviation of the fitness of the population after initialization and σ_X is the standard deviation of the fitness of the population after applying the subtree crossover operator.

Diversity here is a measure of population's dispersion, and a decline in diversity may signal that the population is not converging towards a solution (too much diversity) or is converging prematurely (too little diversity).

In the following section, we discuss the mutation operator, which can be used to introduce diversity into the population, thereby preventing premature convergence.

Mutation

Mutation, another crucial genetic operator, introduces new genetic material into the population, thus maintaining genetic diversity and preventing premature convergence to suboptimal solutions. In the context of GP, the mutation operator modifies a program in the population, while ensuring the resultant individual's syntactic correctness.

In tree-based GP, a common form of mutation is the ***point mutation*** [17, 60], which selects a random node from an individual and replaces it with a random primitive with the same arity. This operator is similar to the bit-flip mutation operator used on section 2.2.4 on page 18. As with bit-flip mutation, point mutation can also be applied with a certain probability to each node in an individual, meaning that more than one node can be mutated in a single individual.

Suppose we mutate all the individuals in the population resulting from the crossover operation in section 2.3.4 on page 30, and that exactly one node is mutated in each individual. Let the selected nodes be $\clubsuit = \sin$ in $\mathbf{I}_2$, $\spadesuit = 7$ in $\mathbf{I}_3$, $\heartsuit = 5$ in $\mathbf{O}_1$, and $\diamondsuit = 2$ in $\mathbf{O}_2$. Then, a possible result of applying the point mutation operator can be:

$$M\left(\begin{bmatrix} I_2 \\ I_3 \\ O_1 \\ O_2 \end{bmatrix}\right) = M\left(\begin{bmatrix} 7 - (5 + \clubsuit(x)) \\ \spadesuit + 2 \\ \heartsuit \cdot 7 \\ x^{\diamondsuit} - (5 + \sin(x)) \end{bmatrix}\right) = \left(\begin{bmatrix} 7 - (5 + \clubsuit'(x)) \\ \spadesuit' + 2 \\ \heartsuit' \cdot 7 \\ x^{\diamondsuit'} - (5 + \sin(x)) \end{bmatrix}\right) = \left(\begin{bmatrix} 7 - (5 + \cos(x)) \\ 6 + 2 \\ 6 \cdot 7 \\ x^3 - (5 + \sin(x)) \end{bmatrix}\right)$$

Here, $\clubsuit'$, $\spadesuit'$, $\heartsuit'$, and $\diamondsuit'$ are random primitives with the same arity as $\clubsuit$, $\spadesuit$, $\heartsuit$, and $\diamondsuit$, respectively. That being, $\clubsuit' = \cos$, $\spadesuit' = 6$, $\heartsuit' = 6$, and $\diamondsuit' = 3$.

The fitness of the individuals in the population after applying the subtree mutation operator is then evaluated. The results of this fitness evaluation are shown in **??** on page ??. A summary of the fitness of the population is presented in **??** on page ??.

Generation 1

Individual	Program	Fitness
$M(\mathbf{I}_2(x)$	$7 - (5 + \cos(x))$	112.297 411
$M(\mathbf{I}_3(x)$	$6 + 2$	296.910 273
$M(\mathbf{O}_1(x)$	$6 \cdot 7$	2 609.386 088
$M(\mathbf{O}_2(x)$	$x^3 - (5 + \sin(x))$	15.295 863

Table 2.17: Population after applying the node mutation operator.

	Fitness	Individual
Best	15.295 863	$M(\mathbf{O}_2(x)$
Worst	2 609.386 088	$M(\mathbf{O}_1(x)$
Average	758.472 409	
Standard deviation	1 073.402 832	

Table 2.18: Fitness summary of the population after applying the node mutation operator.

Just like the crossover operator, mutation can also significantly influence the fitness and diversity of the population. By generating new structures in the population, mutation can help prevent stagnation and maintain diversity, thus avoiding premature convergence to suboptimal solutions.

With this, we can conclude that after at the end of the generation, the population shows an improvement of 98.298% in fitness, and 98.603% in standard deviation.

$$\frac{\bar{\Phi}_i - \bar{\Phi}_M}{\bar{\Phi}_i} = \frac{(44\,551.063\,525 - 758.472\,409)}{44\,551.063\,525} \approx 98.298\%$$

$$\frac{\sigma_i - \sigma_M}{\sigma_i} = \frac{(76\,814.062\,197 - 1\,073.402\,832)}{76\,814.062\,197} \approx 98.603\%$$

fig. 2.10 shows the population after applying the node mutation operator. Two points should be clear from this figure. First, all individuals of the population are closer to the target function than the individuals in the initial population. Second, the new fittest individual is $\mathbf{O}_2$, this individual has a shape similar to the target function.

Figure 2.10: Population after applying the node mutation operator

In summary, mutation plays a pivotal role in the genetic programming process. It injects fresh genetic material into the population, thereby promoting diversity and ensuring the continual exploration of the search space. As demonstrated, mutation not only alters the structure of individuals but can also lead to significant enhancements in fitness. By tactfully combining crossover and mutation operations, genetic programming can efficiently traverse the vast landscape of possible solutions. This underscores the importance of maintaining a delicate balance between exploration (introducing new genetic structures) and exploitation (refining existing successful solutions).

Chapter 3

Relevant Work (State of the Art)

In this chapter, we will explore the current state of the art in the field of genetic algorithm frameworks. While the theory of genetic algorithms is not a recent development, the application of these algorithms continues to evolve[1] with the advancement of various programming languages and tools. Therefore, our focus in this chapter will be on current and actively developed frameworks widely used for studying and implementing genetic algorithms.

Specifically, we will look into Agile Artificial Intelligence in Pharo, DEAP, Jenetics, ECJ, and GeneticSharp. Each section will provide a brief overview of the framework along with basic code samples to highlight their syntax and outline their unique features and differences. We believe this approach will provide a comprehensive perspective on the versatility and diversity of tools available in this field.

This state-of-the-art review does not only offer insights into the current trends and tools in genetic algorithms but also sets the stage for our contribution - a genetic algorithm framework in Kotlin.

3.1 The One Max Problem

For the purpose of illustrating the use of the different frameworks, we will use the *One Max problem* introduced on section 2.2.1 on page 12.

The ***One Max problem***, or the Ones Counting problem, is a classic and straightforward optimization problem often used as a benchmark in the study of evolutionary algorithms and other heuristic search methods. It serves as a deceptively simple yet effective test of an algorithm's optimization ability.

Given a binary string of length n, the *One Max problem* is to find a binary string such that the sum of its bits (counting the number of ones) is maximized. In formal terms, if we denote the binary string as $x = (x_1, x_2, ..., x_n)$ where each $x_i \in \{0,\ 1\}$ for all $i = 1,\ 2,\ \ldots,\ n$, the *fitness function* $\phi(x)$ to be maximized can be expressed as:

$$\phi(x) = \sum_{i=1}^{n} x_i \tag{3.1}$$

The function $f(x)$ counts the number of ones in the string x. The maximum possible value of $f(x)$ is n, which is achieved when all bits in the string are one. The *One Max problem* is an instance of a unimodal problem since there's only one local maximum which is also a global maximum.

It's important to note that despite its simplicity, the *One Max problem* does provide a non-trivial task for many search algorithms. For a binary string of length n, there are 2^n possible solutions. For larger n, an exhaustive search of the solution space is not feasible, hence the need for efficient optimization algorithms.

[1] No pun intended.

Due to its characteristics, the *One Max problem* is often used to evaluate the performance of optimization algorithms especially genetic and evolutionary algorithms. It's particularly well-suited for genetic algorithms as the operations of crossover and mutation can directly change the number of ones in a binary string, thus impacting the fitness of a potential solution.

Despite the straightforward objective function, the *One Max problem* is invaluable in the study of heuristic search methods due to its accessibility, simplicity and the vastness of its search space, which permits the analysis and comparison of the performance of different optimization techniques.

3.2 Agile Artificial Intelligence in Pharo

Alexandre Bergel's ***"Agile Artificial Intelligence in Pharo"*** [28] delivers a comprehensive exploration of genetic algorithms, encapsulating their theory and application within the *Pharo* programming environment. This language, dynamic and reflective, finds its roots in *Smalltalk*.

Bergel's tome not only introduces readers to genetic algorithms but also to neural networks and the concept of neuroevolution. The framework's design and the reasoning behind it are meticulously detailed, offering readers insights into its three main components: ***genetic operators***, ***selection operators***, and the ***evolution engine***. Accompanying this detailed framework are test cases. These serve dual roles: exemplifying the framework's application and guiding enthusiasts in developing their own robust and extensible frameworks.

The book also provides utilities for the visualization of genetic algorithms and neural networks. These are implemented using *Roassal*, a visualization engine for *Pharo*. The engine is capable of rendering a variety of graphs, including directed and undirected graphs, trees, and more relevant to this document, a summary of the evolution fitness over time.

To showcase the framework's utility, the OMP problem resolution via a genetic algorithm is provided:

Listing 3.1: A simple genetic algorithm using Bergel's framework.

```
1  engine := GAEngine new.
2  engine random: (Random seed: 11).
3  engine populationSize: 20.
4  engine numberOfGenes: 20.
5  engine createGeneBlock: [ :rand :index :ind |
6    (0 to: 1) atRandom: rand ].
7  engine fitnessBlock: [ :ind | ind count: [ :each | each = 1 ] ].
8  engine endIfFitnessIsAbove: 20.
9  engine run.
10 engine trace: 'Target fitness reached at generation '.
11 engine traceCr: engine logs last generationNumber.
12 engine trace: 'Best individual is: '.
13 engine traceCr: engine logs last fittestIndividual genes .
14 engine trace: 'with fitness: '.
15 engine traceCr: engine logs last bestFitness .
```

Breaking it down:

1 A new genetic algorithm engine is created.

2 The random seed is set to 11.

3-4 The population and number of genes are both set to 20.

5-6 The gene creation block is outlined for initial population generation, returning a random digit either 0 or 1.

7 The fitness block is then detailed, here counting the number of 1s in an individual.

8 The algorithm's termination condition is set, halting if an individual's fitness surpasses 20.

9 The algorithm is then executed.

10-18 The algorithm's logs are then printed to the console. Note that the engine keeps a log of the evolution process.

While Bergel's framework, as presented in the book, stands as a clear and potent genetic algorithm tool for *Pharo*, its primary intent is instructional. As such, it may not rival the robustness of seasoned frameworks like DEAP or *Jenetics*.

3.3 Distributed Evolutionary Algorithms in *Python* (DEAP)

Distributed Evolutionary Algorithms in Python (DEAP) [43] is a powerful evolutionary computation framework designed for rapid prototyping and validation of concepts. It stands apart from many other evolutionary computation libraries due to its significant modularity and versatility, which enables the construction of a broad range of evolutionary algorithms, genetic algorithms, and even hybrid algorithms. DEAP is open-source and available under the *GNU Lesser General Public License v3.0* (LGPL-3.0) [48].

DEAP is structured around two primary components: the `Creators` and the `Toolbox`. The `Creators` module facilitates the generation of new classes integral to the genetic algorithm, such as individuals and populations. In contrast, the `Toolbox` serves as a comprehensive repository for various operators necessary in evolutionary algorithms, including the evaluation, selection, mutation, and crossover functions.

The following illustrates the use of DEAP to solve the OMP problem using a genetic algorithm:

Listing 3.2: A simple genetic algorithm using DEAP.

```
creator.create("FitnessMax", base.Fitness, weights=(1.0,))
creator.create("Individual", list, fitness=creator.FitnessMax)
toolbox = base.Toolbox()
toolbox.register("attr_bool", random.randint, 0, 1)
toolbox.register("individual", tools.initRepeat, creator.Individual, toolbox.attr_bool, n=20)
toolbox.register("population", tools.initRepeat, list, toolbox.individual)
toolbox.register("evaluate", lambda i: (sum(i),))
toolbox.register("mate", tools.cxTwoPoint)
toolbox.register("mutate", tools.mutFlipBit, indpb=0.05)
toolbox.register("select", tools.selTournament, tournsize=3)
TARGET_FITNESS = 20
if __name__ == "__main__":
    random.seed(11)
    pop = toolbox.population(n=20)
    hof = tools.HallOfFame(1)
    stats = tools.Statistics(lambda i: i.fitness.values)
    stats.register("max", max)
    gen = 0
    while True:
        gen += 1
        offspring = algorithms.varAnd(pop, toolbox, cxpb=0.5, mutpb=0.2)
        fits = toolbox.map(toolbox.evaluate, offspring)
        for fit, ind in zip(fits, offspring):
            ind.fitness.values = fit
        pop = toolbox.select(offspring, k=len(pop))
        hof.update(pop)
        record = stats.compile(pop)
        if record['max'][0] >= TARGET_FITNESS:
            break
    print(f"Target fitness reached at generation {gen}.")
    print(f"Best individual is: {''.join(map(str, hof[0]))}")
    print(f"with fitness: {hof[0].fitness.values[0]}")
```

When executed, the program outputs the following:

```
Target fitness reached at generation 12.
Best individual is: 11111111111111111111
with fitness: 20.0
```

This example demonstrates the use of DEAP to implement a simple genetic algorithm, with the added functionality to stop evolution once a target fitness is achieved.

Here is a breakdown of the code:

1-2 DEAP uses a meta-factory to *dynamically* create user-defined classes. Here, we create two classes: `FitnessMax` for the fitness (single-objective, to be maximized) and `Individual` to represent an individual in the population. The second argument to `create` is the base class, meaning that `FitnessMax` is a subclass of `base.Fitness` and `Individual` is a subclass of `list`.

3 The `Toolbox` is created, it is used to store most of the functions and arguments required to perform the genetic algorithm.

4-5 We register two functions in the toolbox: `attr_bool` which generates a random binary number (0 or 1), and `individual` which creates a new individual using the `attr_bool` function. The individual consists of 20 binary numbers (genes).

6-7 We then register two more functions: `population` which creates a population of individuals, and `evaluate` which evaluates an individual's fitness as the sum of its genes.

8-10 We register three more functions in the toolbox: `mate` for performing *two-point crossover*, `mutate` with a *bit-flip mutator* with a 5% probability, and `select` for performing tournament selection with a tournament size of 3.

11 We set the target fitness to 20 (since we aim to maximize the sum of the genes, the maximum possible fitness is 20)

12-32 In the main section of the code, we seed the random number generator to ensure reproducibility, create a population of 20 individuals, and set up the *Hall of Fame* and a statistics object to keep track of the maximum fitness in the population.

19-29 A while loop is used to run the genetic algorithm until the target fitness is reached. Inside the loop:

- 21-22 We apply crossover and mutation to the population using the `varAnd` function,[2] and then evaluate the fitness of the offspring
- 23-24 We assign the newly computed fitness values to the individuals.
- 25 We replace the old population with the selected individuals from the offspring.
- 26-27 We update the Hall of Fame and compile the statistics.
- 28-29 We check if the maximum fitness has reached the target fitness. If it has, we break the loop and the algorithm stops.

30-32 Finally, we print the results.

This example illustrates the key aspects of using DEAP: creating custom classes, setting up a toolbox, defining and registering functions, and manually controlling the loop of the genetic algorithm.

DEAP provides robust support for multi-objective algorithms and parallelization, which are common requirements in complex optimization problems. The library also includes a set of benchmark functions and examples to help users understand the various algorithms' behavior and performance.

DEAP is widely regarded as one of the most comprehensive and cutting-edge genetic algorithm frameworks currently available, encompassing a wide range of algorithms, including GAs, GP, Evolutionary Strategies (ES), and more. It boasts a wealth of documentation and a robust community of users and contributors. However, certain facets of DEAP may pose challenges for users, as reflected in the preceding code example.

- DEAP's code can be quite verbose, and the utilization of a toolbox might initially confuse newcomers, given that it's not a standard convention in evolutionary algorithms. This characteristic becomes more evident when juxtaposed with the code syntax of other frameworks.

[2] This function simply applies the variation operators to the population.

- A trade-off exists between the framework's flexibility and ease of use. While DEAP's malleability is commendable, it heavily relies on code injection. This reliance complicates static analysis, resulting in warnings such as "Unresolved attribute reference" in *PyCharm* and "`reportGeneralTypeIssues`" in *Visual Studio Code* with *Pylance* (on lines 2, 5, 6, 14, 22, and 25), which can inhibit certain IDE capabilities, including auto-completion and refactoring.
- DEAP's dependence on code injection can potentially introduce risks in production environments. Even though code injection is not inherently unsafe, if handled carelessly, it can result in system vulnerabilities.
- *Python*'s dynamic typing can be a double-edged sword. While it offers flexibility, type errors will not be detected until runtime, which can potentially lead to bugs and render the code more challenging to comprehend.
- For efficiency, DEAP necessitates the *NumPy* library, which isn't part of the standard *Python* library. Although this is not inherently negative, it does introduce an extra dependency that needs to be managed.
- Enhancing DEAP's functionality mandates an in-depth comprehension of the library's architecture, and in many instances, necessitates manual modifications to the algorithm's execution loop. This is evident in the previous example, where manual control of the algorithm's loop is required to halt upon reaching the target fitness.
- The library's tendency to abbreviate variable names can render the code difficult to interpret. Without referring to the documentation, understanding the meaning of variables such as `cxpb` and `mutpb` can be challenging. The problem can be exacerbated by *Python*'s keyword arguments (`kwargs`), as they are not explicitly declared, preventing IDEs from providing valuable information about them.

Despite this, DEAP's flexibility and modularity have made it a popular choice among researchers and practitioners in the field of evolutionary computation.

3.4 Jenetics: Java Genetic Algorithm Library

Jenetics [60] is a robust EC library developed in *Java*. It is a comprehensive framework that provides a *wide range* of genetic algorithms and operators. The library is open-source and available under the *MIT* license [16].

Jenetics structure revolves around two main concepts: `Phenotype`[3] and `Engine`. The `Phenotype` is a representation of a single candidate solution to a given optimization problem. Meanwhile, the `Engine` is the core of the library, which controls the evolution process. It is responsible for the initialization of the population, the execution of the evolution, and the termination of the evolution once the termination condition is met (this uses a very similar approach to the one we presented in section 3.2 on page 36).

Onr thing that sets *Jenetics* apart from other libraries is the use of the *Java's Stream API* [57] to make a seamless integration between it and the `EvolutionStream` class used by the `Engine`. This means that the evolution has access to all the features provided by the Stream API, such as parallelization, filtering, and mapping.

Below, we demonstrate the use of *Jenetics* to solve the OMP with a genetic algorithm:

```
public class OneMax {
  public static void main(String[] args) {
    RandomRegistry.random(new Random(11));
    final var engine =
        Engine.builder((Genotype<BitGene> gt) ->
                          gt.chromosome().as(BitChromosome.class).bitCount(),
                       BitChromosome.of(20, 0.5))
              .maximizing()
              .populationSize(20)
              .alterers(new Mutator<>(0.05), new SinglePointCrossover<>(0.5))
              .build();
    Phenotype<BitGene, Integer> best = engine.stream()
                                            .limit(Limits.byFitnessThreshold(19))
                                            .collect(EvolutionResult.toBestPhenotype());
```

[3] See definition A.14 on page 90

```
    System.out.println("Target fitness reached at generation: " + best.generation());
    System.out.println("Best individual is: " + best.genotype());
    System.out.println("with fitness: " + best.fitness());
  }
}
```

When executed, the program outputs the following:

```
Target fitness reached at generation: 40
Best individual is: [00001111|11111111|11111111]
with fitness: 20
```

This example demonstrates *Jenetics*' usage for implementing a straightforward genetic algorithm, with the added ability to stop evolution once steady fitness is achieved.

Here is an explanation of the code:

1-2 First, a class containing a `main` method is established. This forms the launching point of the program.

3 The `RandomRegistry` is then utilized to preset the random seed, a step that ensures consistency and replicability of program outputs.

4-11 The core configuration of the evolutionary engine occurs next, tailored specifically to the problem at hand.

5-7 An `Engine` is instantiated using the `Engine.builder` method, requiring two arguments: a `Function` and a `Genotype`. The `Function` assesses the fitness of a `Genotype` by tallying the quantity of `1`s in the `Genotype`. A `Genotype` is created via the `BitChromosome.of` method, which mandates the `length` of the chromosome and the `probability` of a `1` appearing within the chromosome. Here, `BitChromosome` is a distinct version of the `Chromosome` interface, used to depict a `Genotype` composed of `BitGenes`.

8 The `maximizing` method signals that the objective is to boost fitness to its maximum possible value.

9 The `populationSize` method sets the size of the evolutionary population.

10 The `alterers` method sets the genetic operations to be used in the evolution. Here, the `Mutator` and `SinglePointCrossover` are selected. The `Mutator` behaves similarly to a *bit-flip mutation* operator in this context.

11 Finally, the `build` method assembles the functional `Engine`.

12-14 Execution of the evolution process follows.

12 The `stream` method generates an `EvolutionStream` from the `Engine`, allowing for control over the evolution process.

13 The `limit` method sets the termination condition. The evolution halts once a `Phenotype` possessing a fitness **exceeding** 19 is located.

14 Lastly, the `collect` method gathers the evolution's output, in this case, capturing the optimal `Phenotype` discovered.

15-17 The final stage involves printing the evolution results to the console for review and analysis.

This example showcases the key aspects of using *Jenetics*: creating genotypes, setting up an engine, and controlling the evolution process.

Jenetics provides extensive support for multi-objective algorithms and parallel execution, which are common necessities in intricate optimization problems.

Recognized as one of the most comprehensive and cutting-edge genetic algorithm libraries today, *Jenetics* receives wide-ranging acclaim. However, certain aspects of *Jenetics* could be challenging for its users.

- As a Java library, *Jenetics* unavoidably incorporates some of Java's limitations, notably its verbosity and complexity.

- The library's documentation could be more comprehensive. Currently, some examples are outdated or have redundant implementations.
- The insufficient documentation is most evident in the use of the `RandomRegistry` to set the random seed, as seen in example line 3. The documentation claims this ensures the replicability of the program's output, but practical execution proves otherwise. More complete code samples and concrete use cases would ameliorate this issue.
- There is a delicate balance between the framework's flexibility and its usability. *Jenetics*' notable adaptability leans heavily on builders and factories, which can occasionally obfuscate the code.
- Java's lack of robust type inference can lead to verbosity, particularly evident when using classes like `Phenotype` requiring multiple type parameters.
- *Jenetics* relies on the Java Stream API for parallelization to maintain efficiency. However, this approach is not typical in the field of evolutionary computation, making it challenging for beginners. Furthermore, the use of the Stream API can result in difficult-to-debug code due to the execution mainly handled by Java's standard library.
- By default, *Jenetics* operates in parallel. This can detrimentally affect program performance when working on small populations or problems with fewer variables due to parallelization's overhead.
- Complex problem-solving with *Jenetics* can quickly complicate the code, introducing *codecs*, *proxies*, and the `Problem` interface.
- *Jenetics* extensively employs features not found in Java's standard library, particularly immutable data, thereby steepening the learning curve for beginners.
- *Jenetics* is compatible with newer versions of Java, potentially problematic for users unable to update their Java version. This issue is especially relevant for most Android devices, which do not support newer Java versions.
- There's a lack of a robust mechanism for visualizing the evolution process, a feature commonly found in other evolutionary computation libraries.
- *Jenetics* isn't entirely type-safe, necessitating explicit casts via the `as` method.

Despite these shortcomings, *Jenetics* is a powerful and flexible library that can be used to solve a wide variety of optimization problems. The library is actively maintained, and it is one of the most popular evolutionary computation libraries available today.

3.5 ECJ: A Java-based Evolutionary Computation Research System

The ***Evolutionary Computation in Java*** (ECJ) [45] is a powerful and flexible framework for conducting research and experiments in the field of evolutionary computation. ECJ has been under active development for more than two decades, and during this period, it has grown into one of the most comprehensive open-source[4] libraries for evolutionary computation.

What sets ECJ apart from other libraries is its ability to handle a broad range of evolutionary computation paradigms. It offers support for genetic algorithms, genetic programming, multi-objective optimization, co-evolution, and many more. It also comes with built-in functionality for distributed computing, allowing researchers to harness the power of large-scale computing clusters for their experiments.

The ECJ library is designed with flexibility and extensibility in mind. It employs a design pattern that uses parameter files for configuration, thus enabling researchers to customize the algorithms according to their specific needs without having to modify the source code. The library is highly modular, and its components can be easily replaced or extended.

The following code sample demonstrates how to solve the *OneMax* problem using ECJ:[5]

```
1 public class OneMax extends Problem implements SimpleProblemForm {
2   @Override
3   public void evaluate(final EvolutionState state,
4                        final Individual ind,
```

[4] No license is provided with the ECJ source code.
[5] We couldn't find a way to set the random seed in ECJ, so the results are not reproducible.

```
5                       final int subpopulation,
6                       final int threads) {
7     if (ind.evaluated) {
8       return;
9     }
10    if (!(ind instanceof BitVectorIndividual)) {
11      state.output.fatal("Individual must be a BitVectorIndividual")
12    }
13    int sum = 0;
14    // This is necessary to safe-cast ind to BitVectorIndividual
15    BitVectorIndividual ind2 = null;
16    if (ind instanceof BitVectorIndividual) {
17      ind2 = (BitVectorIndividual) ind;
18    }
19    if (ind2 != null) {
20      for (int x = 0; x < ind2.genome.length; x++) {
21        sum += (ind2.genome[x] ? 1 : 0);
22      }
23
24      if (!(ind2.fitness instanceof SimpleFitness)) {
25        state.output.fatal("Fitness must be a SimpleFitness")
26      }
27      ((SimpleFitness) ind2.fitness)
28          .setFitness(state,
29              /* fitness = */ sum / (double) ind2.genome.length,
30              /* isIdeal = */ sum == ind2.genome.length);
31      ind2.evaluated = true;
32    }
33  }
34 }
```

An explanation of the code:

1 The class `OneMax` extends `Problem` and implements `SimpleProblemForm`, which is ECJ's interface for problems that can be solved with generational evolutionary algorithms.

2-6 The `evaluate` method is overridden from the `Problem` class. It is responsible for evaluating the fitness of an `Individual` (a candidate solution in the population). The arguments include the current state of the evolution, the individual to be evaluated, the subpopulation to which the individual belongs, and the number of the thread executing this method (useful in a multithreaded setting).

7-9 The method checks if the individual has already been evaluated. If it has, then the method immediately returns to avoid unnecessary computation.

10-12 If the individual is not an instance of `BitVectorIndividual` (which represents individuals whose genome consists of bits), a fatal error is reported. The OneMax problem assumes a bit string representation of individuals.

13-18 Variable `sum` is initialized to keep track of the total number of 1s in the individual's bit string representation. The individual is then safe-casted to `BitVectorIndividual`. This is necessary because the `ind` argument is of type `Individual`, but we need to work with its `BitVectorIndividual` specifics (such as its `genome` property).

20-22 If the casting is successful, a loop iterates through the genome (bit string) of the individual, incrementing `sum` for every bit set to 1.

24-26 If the fitness of the individual is not an instance of `SimpleFitness` (which represents a single floating-point fitness value), a fatal error is reported.

27-30 The individual's fitness is then set to the proportion of bits set to 1 in the bit string (`sum / (double) ind2.genome.length`).

If all bits are set to 1 (i.e., `sum` equals the length of the genome), then the individual is marked as ideal.

31 Finally, `ind2.evaluated` is set to `true`, indicating that the individual has been evaluated.

To run the OneMax problem, we need to create a parameter file that specifies the problem, the evolutionary algorithm, and the parameters of the algorithm. The following is a sample parameter file for the OneMax problem:

```
1  breedthreads = 1
2  evalthreads = 1
3  seed.0 = 11
4  eval.problem = ec.app.onemax.OneMax
5  pop.subpop.0.species.ind = ec.vector.BooleanVectorIndividual
6  pop.subpop.0.species.genome-size = 20
7  pop.subpop.0.species.fitness = ec.FitnessSimple
8  pop.subpop.0.size = 20
9  pop.subpop.0.species.pipe = ec.vector.breed.VectorMutationPipeline
10 pop.subpop.0.species.pipe.source.0 = ec.vector.breed.VectorCrossoverPipeline
11 pop.subpop.0.species.pipe.source.0.prob = 0.5
12 pop.subpop.0.species.pipe.source.0.source.0 = ec.select.TournamentSelection
13 pop.subpop.0.species.pipe.source.0.source.1 = ec.select.TournamentSelection
14 eval.problem.terminateFitness = 20
```

This is a parameter file for ECJ, which specifies various configuration settings for an evolutionary computation run.

1-2 `breedthreads` and `evalthreads` determine the number of threads to be used for breeding and evaluation operations, respectively. Both are set to 1, which means a single-threaded operation.

3 `seed.0` sets the initial seed for the random number generator. Different seeds will lead to different runs, even with identical configuration parameters.

4 `eval.problem` sets the problem to be solved, in this case, `ec.app.onemax.OneMax`, which refers to the OneMax problem class implemented earlier.

5-6 `pop.subpop.0.species.ind` sets the type of individual in the population. `ec.vector.BooleanVectorIndividual` means that each individual will be a Boolean vector, which is appropriate for the OneMax problem. `pop.subpop.0.species.genome-s` sets the length of the Boolean vector to 20.

7-8 `pop.subpop.0.species.fitness` sets the type of fitness to be used. `ec.FitnessSimple` represents a single floating-point fitness value. `pop.subpop.0.size` sets the population size to 20.

9 `pop.subpop.0.species.pipe` defines the pipeline for breeding new individuals. `ec.vector.breed.VectorMutationPipeline` means a mutation operation will be applied.

10 `pop.subpop.0.species.pipe.source.0` sets the source of individuals for the mutation operation to be a crossover operation (`ec.vector.breed.VectorCrossoverPipeline`), with a crossover probability of 0.5.

12-13 `pop.subpop.0.species.pipe.source.0.source.0` and `pop.subpop.0.species.pipe.source.0.source.1` set the source of individuals for the crossover operation to be tournament selection (`ec.select.TournamentSelection`).

14 `eval.problem.terminateFitness` sets the fitness level at which the evolutionary run should terminate. When an individual reaches a fitness of 20, the run will stop, implying that a perfect solution (all bits set to 1) has been found for the OneMax problem.

To run the OneMax problem, we need to execute the following command:

```
java -cp .
ec.Evolve -file .\ec\app\onemax\one_max.properties
```

This command will run the OneMax problem with the parameter file `one_max.properties` in the package `ec.app.onemax`. According to the official ECJ documentation, this command should be able to solve the OneMax problem, however, we were unable to get it to work. The following is the output of the command:

```
# Code omitted for brevity
Exception in thread "main" ec.util.ParamClassLoadException:
No class name provided.
PARAMETER: state
        at ec.util.ParameterDatabase.getInstanceForParameter(ParameterDatabase.java:493)
        at ec.Evolve.initialize(Evolve.java:479)
        at ec.Evolve.initialize(Evolve.java:412)
        at ec.Evolve.main(Evolve.java:758)
```

The error message indicates that the parameter `state` is missing. This parameter is required by ECJ, but is not specified in the parameter file. We tried to add the parameter to the file, but the error persisted. We also tried to run the OneMax problem with the parameter file provided by the ECJ documentation, but the error persisted. We were unable to find a solution to this problem.

While ECJ stands as one of the most comprehensive and sophisticated Evolutionary Computation (EC) frameworks available, it presents users with a daunting learning curve. We've identified several areas of concern that contribute to this complexity:

- **Verbose and obscure code:** The framework's verbosity hinders code comprehension. To illustrate, the `eval` method in the `OneMax` class spans 31 lines, yet only 3 of these lines (20-22) actually pertain to the OneMax problem itself—the rest are framework-related.
- **Complex configuration:** Setting up the framework involves numerous layers of boilerplate code, necessitating a deep understanding of the underlying concepts and terminology. Lines such as `pop.subpop.0.species.pipe.source.0.source.0` can be particularly opaque to newcomers.
- **Outdated documentation and implementation:** The `README.md` file has not been updated in 4 years, and the OneMax class equivalent was last refreshed 5 years ago. Even the most recent version (27) of the framework dates back 4 years, rendering much of the documentation and implementation obsolete.
- **Legacy Java syntax:** ECJ's reliance on older Java syntax, specifically the lack of generics, results in frequent casting and `instanceof` checks. This not only makes the code more brittle but also complicates understanding.
- **Exclusion from popular package managers:** The absence of ECJ from widely-used package managers like Maven or JitPack imposes additional burdens on integration with other projects, as manual download and compilation become necessary.
- **Limited IDE compatibility:** The design of ECJ emphasizes command line compilation and execution, undermining the utility of modern Integrated Development Environments (IDEs) and complicating debugging efforts.
- **Inconvenience for integration:** Rather than being crafted for use as a library, the framework is designed with the expectation of direct source code inclusion. This requirement to import source code into users' own projects increases the difficulty of integration.
- **Generic name complicates research:** The term 'ECJ' is sufficiently common to cause confusion when seeking information on the framework. Queries such as "ECJ" or "ECJ Java" yield results concerning the European Court of Justice or the Eclipse Compiler for Java, respectively, rather than the intended framework.
- **Limited community and resources:** As of 2023, ECJ lacks a robust community of developers or users. The dearth of online resources like tutorials or Stack Overflow discussions compounds the challenges for new users seeking to leverage the framework effectively.
- **Non-utilization of modern Java features:** Beyond the lack of generics, the framework fails to harness modern Java features like lambda expressions, lists, or Optional types. These features, by enabling more concise and readable code, can significantly enhance error prevention.

In conclusion, the Evolutionary Computation in Java (ECJ) framework, while extensive and capable, poses considerable barriers to entry for users due to its complexity, outdated documentation, and heavy reliance on legacy Java syntax. Furthermore, despite its versatility in supporting various evolutionary computation paradigms and distributed computing, the steep learning curve, absence from popular package managers, and limited compatibility with modern IDEs make it challenging for newcomers and experienced developers alike. The verbose code, intricate configuration requirements, and limited use of modern Java

features also contribute to the difficulty in understanding and integrating the ECJ into projects. Our exploration further exposed issues with the execution of sample problems, highlighting the need for updated documentation and maintenance. However, ECJ's modular design, extensibility, and breadth of capabilities maintain its status as a valuable tool for research in the field of evolutionary computation. Future efforts could focus on modernizing the framework, improving documentation, and fostering an active user community to mitigate the identified issues and enhance its usability.

3.6 GeneticSharp: A .NET-based Genetic Algorithm Framework

GeneticSharp [32] is a fast, extensible, multi-platform and multithreading *C#* Genetic Algorithm library that **simplifies the development of applications** using genetic algorithms. It is open-source and licensed under the *MIT License* [16].

The central theme of *GeneticSharp* revolves around three main concepts: `Chromosome`, `Population`, and `GeneticAlgorithm`. The `Chromosome` class is the base for all types of chromosomes, storing genes and managing genetic operators. The `Population` class holds a set of chromosomes which are evolved over time. The `GeneticAlgorithm` is the central point of execution and contains the genetic algorithm's main loop.

GeneticSharp stands out with its support for parallelism, which greatly speeds up the execution of fitness evaluation, crossover, and mutation. It also introduces the concept of extensions, providing a simple way to add new operators and customization to the genetic algorithm.

Here's an example of how to solve the OMP using *GeneticSharp*:

```
public sealed class BinaryChromosome : BinaryChromosomeBase {
    public BinaryChromosome(int length) : base(length) {
        CreateGenes();
    }
    public override IChromosome CreateNew() {
        return new BinaryChromosome(Length);
    }
}
public class OneMaxFitness : IFitness {
    public double Evaluate(IChromosome chromosome) {
        return chromosome.GetGenes().Count(gene => (int)gene.Value == 1);
    }
}
public class ReproducibleRandom : RandomizationBase {
    private static readonly object GlobalLock = new();
    private static readonly ThreadLocal<Random?> ThreadRandom = new(NewRandom);
    private static Random? Instance => ThreadRandom.Value;
    private static Random NewRandom() {
        lock (GlobalLock) {
            return new Random(11);
        }
    }
    public override int GetInt(int min, int max) {
        Debug.Assert(Instance != null, nameof(Instance) + " != null");
        return Instance.Next(min, max);
    }
    public override float GetFloat() {
        Debug.Assert(Instance != null, "ReproducibleRandom.Instance != null");
        return (float)Instance.NextDouble();
    }
    public override double GetDouble() {
        Debug.Assert(Instance != null, "ReproducibleRandom.Instance != null");
        return Instance.NextDouble();
```

```
34     }
35 }
36 const int chromosomeLength = 20;
37 const int populationSize = 20;
38 RandomizationProvider.Current = new ReproducibleRandom();
39 var selection = new EliteSelection();
40 var crossover = new UniformCrossover();
41 var mutation = new FlipBitMutation();
42 var fitness = new OneMaxFitness();
43 var chromosome = new BinaryChromosome(chromosomeLength);
44 var population = new Population(populationSize, populationSize, chromosome);
45 var ga = new GeneticAlgorithm(population, fitness, selection, crossover, mutation) {
46     Termination = new FitnessThresholdTermination(chromosomeLength)
47 };
48 ga.Start();
49 Console.WriteLine($"Target fitness reached at generation: {ga.GenerationsNumber}");
50 var bestChromosome = ga.BestChromosome as BinaryChromosome;
51 Console.WriteLine($"Best individual is: {bestChromosome}");
52 Console.WriteLine($"with fitness: {bestChromosome?.Fitness}");
```

Here is an explanation of the code:

1-8 The `BinaryChromosome` class, which is sealed to prevent further subclassing, represents a chromosome with binary genes. Sealing the class is a best practice when a class has a constructor that calls a virtual method (in this case, the `CreateGenes`) By sealing the class, we prevent any confusion that could arise if a subclass were to override the virtual method.

2-4 This is the constructor which takes a length as an argument and passes it to the base constructor. Note that the base constructor might call a virtual method, and in order to prevent the virtual method from being overridden by a subclass (which could lead to unexpected behavior if the subclass isn't aware that the method is called in the constructor), we seal the class.

4-7 The `CreateNew` method returns a new instance of `BinaryChromosome` with the same length.

9-13 The `OneMaxFitness` class calculates the fitness of a chromosome.

10-12 The `Evaluate` method counts the number of genes with value 1 (representing binary `true`) in the chromosome. This is the measure of fitness in the OneMax problem.

14-35 The `ReproducibleRandom` class is used for generating random numbers in a reproducible way. This is especially important to ensure that the results of the genetic algorithm are reproducible across multiple runs.

15-17 A global lock and a thread-local `Random` instance (`ThreadRandom`) are declared. These ensure that each thread gets its own `Random` instance, but they're all seeded in a synchronized way to provide reproducibility across multiple threads.

18-22 The `NewRandom` method is used to create a new `Random` object with a seed of 11. The method is thread-safe due to the use of the `lock` keyword.

23-34 The `GetInt`, `GetFloat`, and `GetDouble` methods are overridden to return values from the `Random` instance.

36-37 Here, we define some constants for the genetic algorithm, including the length of the chromosomes and the size of the population.

38 This line sets the current randomization provider to our `ReproducibleRandom` class.

39-44 These lines set up the different components of the genetic algorithm, including the selection method (elite selection), crossover method (uniform crossover), mutation method (flip bit mutation), and fitness function (`OneMaxFitness`).

45-47 These lines create the initial population and the genetic algorithm instance.

46 This line sets the termination condition for the genetic algorithm to be when a chromosome achieves the maximum possible fitness (equal to the chromosome length).

48 This line starts the genetic algorithm.

49-52 These lines print the number of generations it took to reach the target fitness, the best chromosome, and its fitness.

GeneticSharp's strengths lie in its simplicity and ease of use. It presents a straightforward and flexible API that allows the creation of custom chromosomes, fitness functions, and genetic operators, and the framework's design allows easy parallelization of computations.

While *GeneticSharp* offers numerous benefits, it also possesses certain drawbacks.

- Its API's verbosity may be challenging, particularly when contrasted with more succinct frameworks like Jenetics. This is evident in our previous example where defining classes for the chromosome, fitness function, and randomization provider was necessary.
- *GeneticSharp*'s documentation lacks depth in comparison to its counterparts (in particular DEAP), which coupled with the shortage of available examples, may make it less user-friendly, especially for newcomers.
- Supporting parallelization within *GeneticSharp* may introduce complexities when implementing custom genetic operators. A case in point is the `ReproducibleRandom` class in our example. It necessitates synchronization techniques to ensure that each thread possesses its own `Random` instance. This forces users to familiarize themselves with the framework's threading model.
- The `BinaryChromosome` class is marked as `sealed`, an advanced feature of *C#* which could be perplexing for novices.
- *GeneticSharp* is designed primarily for application development, thereby potentially limiting its appeal to researchers aiming to experiment with various genetic algorithms.
- As of 2023, the *GeneticSharp* community remains relatively small compared to communities dedicated to more established machine learning or AI libraries. This may result in fewer resources, less frequent updates, and limited community support.
- In its current version (as of 2023), *GeneticSharp* exclusively supports classic genetic algorithms, lacking support for other evolutionary algorithms like genetic programming.

In conclusion, *GeneticSharp* provides a robust, multi-threaded platform for implementing genetic algorithms using *C#*. Its support for parallelism and customizable genetic operators highlight its adaptability and utility. With its emphasis on application development, it provides an intuitive platform for developers to integrate genetic algorithms into their software.

However, its merits must be balanced against its potential drawbacks, including a more verbose API compared to some other frameworks, limited documentation and examples, complexities introduced by parallelization support, and the use of advanced features that may not be beginner-friendly. As of 2023, the community around *GeneticSharp* is smaller, which might lead to fewer resources, less frequent updates, and limited support. The framework currently only supports classic genetic algorithms, which may restrict those wishing to explore other evolutionary algorithms.

Overall, *GeneticSharp* is a powerful tool in the *.NET* ecosystem for developers needing to leverage genetic algorithms in their applications. As with any tool, it is essential to understand its capabilities and limitations to make the most effective use of it

3.7 Other Libraries

The preceding sections elucidated a select set of leading frameworks in EC. Yet, our research ambit encompassed a broader array of libraries. This thorough examination was pivotal in sculpting our proposed *Kotlin*-based EC framework. By discerning the strengths and nuances of each library, we endeavored to assimilate their best features, circumventing common shortcomings.

Below is a succinct overview of additional frameworks:

- **EvolvingObjects** (EO) [46] - A template-based *C++* framework, it encompasses a myriad of evolutionary algorithms and operators, such as GAs, ES, and PSO. At present, certain features like GP are not supported.

- **Inspyred** [49] - A Python-oriented framework, Inspyred, inspired by De Jong [15], distinctly separates algorithm-specific computations from problem-centric ones. It encompasses a diverse array of evolutionary algorithms, including GAs and ES. Currently, it does not support GP.
- **Pyevolve** - This Python-centric framework provides multiple evolutionary algorithms, inclusive of GAs and GP. However, it's worth noting that it's no longer under active maintenance.
- **PGAPack** [54] - Crafted in *C*, this parallel genetic algorithm library boasts of features like interoperability with *Fortran*, *C*, and *C++*, compatibility across varied parallel architectures, and a *Python* interface [55].
- **pagmo** [35] - A *C++* library, pagmo prioritizes massively parallel optimization, offering a unified interface for diverse optimization algorithms, including evolutionary ones. It also features a *Python* interface [37].
- **easy_ga** [44] - Based in *Rust*, easy_ga facilitates swift GA prototyping, predominantly supporting traditional GAs.
- **genevo** [47] - Another *Rust*-oriented framework, genevo also chiefly supports classic GAs.
- **Evolutionary Computation Framework** (ECF) [31] - ECF, written in *C++*, offers a rich set of evolutionary algorithms, rivalling frameworks like DEAP and ECJ in terms of comprehensiveness. However, it grapples with issues akin to ECJ, such as limited documentation and an inherent steep learning curve.

The multifariousness and capabilities intrinsic to these libraries underscore the expansive potential within the realm of genetic algorithms. Our deep dive into each has equipped us to craft a *Kotlin*-based framework that endeavors to synergize the strengths of each while introducing innovative attributes.

Chapter 4

The Keen Framework

4.1 Introduction

Within the expansive landscape of *evolutionary computation* (EC), a multitude of methodologies and algorithms have emerged, each demonstrating significant success in addressing complex optimization problems. Nevertheless, the practical application of these techniques often presents its own challenges, most notably, the extensive repetitive coding required and the absence of an intuitive, user-friendly platform. To address these challenges, this chapter introduces ***Keen***, an innovative genetic algorithms framework developed using the *Kotlin* programming language.

As a comprehensive and adaptable framework, Keen embarks on an ambitious journey to promote the application and acceptance of evolutionary algorithms across various domains. This well-structured framework is currently designed to simplify the implementation process of *genetic algorithms* (GAs) and *genetic programming* (GP), with an emphasis on allowing extension to other evolutionary algorithms, thereby accelerating the pace of research and development in the EC field.

This chapter provides an in-depth examination of the *Keen framework*, illuminating its architecture, fundamental components, and the array of functionalities it offers. We explore the range of genetic operators within Keen, including various selection, mutation, and crossover techniques. A comprehensive account of the GAs and GP methods within *Keen*'s repertoire will be provided, underscoring their distinctive attributes and potential use-cases. In addition, the chapter delves into the framework's parallelism capabilities, a feature that significantly boosts computational efficiency and scalability.

Beyond its existing features, the extensibility of the Keen framework will also be examined, demonstrating its readiness to incorporate novel algorithms and techniques. Keen is designed with a view to facilitate easy expansion and modification of its components, encouraging innovation and continuous progression, in tandem with advancements in the EC field.

The introduction of *Keen* marks a valuable contribution to both the *Kotlin* and AI communities, offering a robust, user-friendly platform for the development, testing, and deployment of evolutionary algorithms. We envision that this pioneering framework will encourage broader application of evolutionary algorithms, thus pushing the boundaries of artificial intelligence research and application.

4.2 Architecture

Keen, our sophisticated and flexible genetic programming library, organizes its architectural blueprint into five distinguishable modules: *Evolution Engine*, *Genetic Material*, *Genetic Operators*, *Utility*, and *Programs*.

The *Utility* module serves as the library's toolbox, providing various auxiliary classes and functions that the remaining modules can utilize. As its content is primarily supplementary, it won't be thoroughly examined within this document.

The *Genetic Material* module encapsulates the foundational elements defining the characteristics of our evolving entities or `Phenotypes`. Its core constituents are the `Phenotype` and `Genotype` classes, accompanied by the encompassing `Chromosome` and `Gene` hierarchies.

Within the *Genetic Operators* module, we define the different genetic operators that the *Evolution Engine* leverages to stimulate the evolutionary process. This module accommodates classes embodying crossover, mutation, and selection operators, systematically grouped into corresponding categories: **Crossover**, **Mutation**, and **Selection**.

The *Evolution Engine* module houses classes that construct the engine of evolution itself, along with essential classes promoting the smooth progression of the evolution process.

Finally, the nascent *Programs* module showcases genetic programming paradigms through a collection of programs developed under this concept. However, as it is in its preliminary stages, this document will not cover it in detail.

A majority of the *Keen* classes are parameterized by the type of data they handle and the gene type composing the population individuals, like an integer or an integer gene. This design choice lends *Keen* a wide range of flexibility, enabling users to tailor the genetic material for the evolution engine. Moreover, thanks to *Kotlin*'s proficient type inference, we can generally omit type parameters, thereby boosting code legibility.

Keen upholds the principle of immutability, rendering most of its classes immutable. This provides an assurance of robustness and predictability, as objects remain unaltered post-creation, enhancing parallelization by allowing shared memory use without the need for synchronization mechanisms.

Furthermore, *Keen* offers factory methods for most classes, fostering an intuitive and straightforward programming experience. It employs language-oriented programming techniques to form a domain-specific language (DSL), streamlining object creation with more readable code. This approach is evident in libraries such as *Kotlinx.html* [34], *Kotlin Telegram Bot* [33], and *Kotest* [50]. While we promote factory methods as the primary mode of object creation, the design caters to manual object creation, granting users greater control over the process when needed.

Here's an illustrative DSL example, where we create a `Genotype` for the *Room Scheduling* problem. Here, each `Chromosome` represents a meeting, composed of a single integer gene indicating the assigned room. The full code snippet can be found in listing C.1 on page 110.

Listing 4.1– Usage of the *Keen* Domain Specific Configuration Language (KDSCL) to create a `Genotype` for the *Room Scheduling* problem.

```
genotype {                  // We are creating a Genotype
    repeat(meetings.size) { // We create a Chromosome for each meeting
        chromosome {        //
            ints {          // We create a single integer gene
                size = 1    // The gene is a single integer
                // The gene can take values from 0 to the number of meetings
                range = meetings.indices.first to meetings.indices.last
            }
        }
    }
}
```

The approach we see here closely mirrors the use of configuration files in ECJ. However, as our DSL is defined entirely within *Kotlin*, it offers significant advantages. These include type safety, robust Integrated Development Environment (IDE) support, and the ability to leverage the full capabilities of the language in defining our genotypes.

In the given example, we employed the standard `repeat` function from the *Kotlin* library to create a number of chromosomes equal to the number of meetings. It is essential to note that this function is not part of our DSL but a standard feature provided by the language.

Furthermore, our approach enhances the readability of the code. Compared to using ECJ's configuration files, the DSL presents a more intelligible and cleaner interface. By defining the configurations in the same language and even within the same files as the rest of the code, we significantly improve code readability. This consolidation offers a more streamlined, unified development environment that facilitates easier understanding and manipulation of the codebase.

In the following sections, we will delve deeper into the pivotal classes within these modules and examine their functions and significance.

4.3 Genetic Algorithms

As we delve deeper into the intricate workings of the framework, it becomes imperative to explore its foundational components. While the theoretical underpinnings of Genetic Algorithms (GAs) have been exhaustively discussed in section 2.2 on page 12, this section aims to shed light on the actual implementation and design specifics of the GAs within our framework. Our exploration will be segmented into three primary components: the representation and structuring of the ***genetic material***, the core mechanics of the ***evolution engine***, and the crucial role of the ***evolution listeners***. Each subsection will provide a granular understanding of how these elements function and interconnect, ensuring that the GAs are optimally tailored to meet the design goals of the framework.

4.3.1 Genetic Material

The genetic material stands at the heart of GAs, representing the problem domain's primary aspect. Consequently, it's essential to structure this material efficiently and flexibly.

Our approach is "gene-centric". Rather than relying on indirect representations, the genetic material is depicted as a collection of genes. Such a representation bolsters the flexibility of the API design. For instance, the genetic material can manifest as a single gene, a collection of genes (referred to as a chromosome), or even as multiple collections of genes (genotype).

Furthermore, the genetic material is immutable. This immutability ensures thread safety and adheres to the "copy-on-write" principle, both of which are indispensable for crafting robust GAs.

Gene The gene stands as the fundamental unit of genetic material within our framework, representing the minutest unit that can undergo manipulation.

Fundamentally, each gene must be capable of: 1. *mutation*, 2. *reproduction*, and 3. *information storage*. To promote extensibility, genes are designed to handle minimal operations, passing most tasks to operators (refer to section 4.4 on page 55). This approach ensures the framework's adaptability across diverse genetic materials, including binary, integer, real, and custom types.

Thus, we suggest the following interface for gene implementation:

Listing 4.2– Gene interface

```
interface Gene<DNA, G: Gene<DNA, G>> : GeneticMaterial<DNA, G>,
                                       SelfReferential<G> {
    val dna: DNA
    fun mutate(): G
    fun withDna(dna: DNA): G
}
```

Chromosome A chromosome is defined as an iterable collection of genes. Its primary role is to offer a streamlined way to manipulate a group of genes. It's crucial to keep the chromosome's responsibilities minimal to foster adaptability across diverse genetic materials.

To this end, we suggest the following interface for each chromosome:

Listing 4.3– Chromosome interface

```
interface Chromosome<DNA, G : Gene<DNA, G>> : GeneticMaterial<DNA, G>,
                                              Iterable<G> {
    val genes: List<G>
    fun withGenes(genes: List<G>): Chromosome<DNA, G>

    interface Factory<DNA, G : Gene<DNA, G>> {
        fun make(): Chromosome<DNA, G>
    }
}
```

Incorporated within is the `Factory` interface, dedicated to the creation of new chromosomes. This is harnessed by the framework whenever there's a need to generate new chromosomes.

Genotype Similar to the chromosome, the genotype serves as an iterable collection of chromosomes. Though they share analogous responsibilities, the genotype operates on a broader spectrum. While many frameworks tend to merge the definitions of chromosomes and genotypes, distinguishing between them is paramount to our framework's adaptability. This distinction facilitates the simultaneous manipulation of multiple chromosomes, paving the way for more intricate genetic materials.

Utilizing parametric polymorphism, we negate the necessity for a hierarchical type relationship between genotypes and chromosomes. Consequently, the genotype can be represented by a concrete class, as demonstrated below:

Listing 4.4– Genotype class

```
class Genotype<DNA, G : Gene<DNA, G>>(val chromosomes: List<Chromosome<DNA, G>>) :
    GeneticMaterial<DNA, G>, Iterable<Chromosome<DNA, G>> {

    class Factory<DNA, G : Gene<DNA, G>> {
        fun make(): Genotype<DNA, G> { ... }
    }
}
```

Embedded within is the `Factory` class, designated for the generation of new genotypes.

Phenotype The phenotype epitomizes the expression of the genetic material. Serving as the evaluated manifestation of the genotype, it embodies the candidate solution. Fundamentally, the phenotype hinges on:

1. A *fitness* value, and
2. The associated *genotype*.

In representing individual entities within a population, it's imperative that a phenotype can be compared with its peers. In light of this, the structure for the phenotype is delineated as:

Listing 4.5– Phenotype class

```
class Phenotype<DNA, G : Gene<DNA, G>>(
    val genotype: Genotype<DNA, G>,
    val fitness: Double = Double.NaN
) : GeneticMaterial<DNA, G>, Comparable<Phenotype<DNA, G>> { ... }
```

4.3.2 Evolution Engine

The ***evolution engine*** serves as the framework's nucleus, orchestrating the evolution of the population. It encapsulates all functions essential for the evolutionary process. This engine integrates the fundamental structure of the evolutionary algorithms, invoking relevant functions from associated entities like the ***genetic material*** and ***genetic operators***.

Specifically, the engine manages:

1. Population initialization;
2. Population evaluation;
3. Selection of individuals for the forthcoming generation;
4. Genetic operator application on chosen individuals;
5. Evaluation of the newly formed individuals;
6. Supplanting the old population with the new;

7. Iteration from step 3 until meeting termination criteria.

Furthermore, the engine offers hooks for both pre and post-processing steps related to the population and individuals. These hooks facilitate user-driven customization of the evolution trajectory. Although this feature remains outside this work's scope, its availability merits acknowledgment.

The engine's architecture is depicted in **??** on page ??.

Listing 4.6– Evolution engine structure

```
class Engine<DNA, G : Gene<DNA, G>>(...) : Evolver<DNA, G> {
    override fun evolve(): EvolutionResult<DNA, G> {
        var evolution = EvolutionState.empty<DNA, G>()
        var result = EvolutionResult(optimizer, evolution.population, generation)
        while (limits.none { it(this) }) {
            result = evolve(evolution)
            evolution = result.next()
        }
        return result
    }
    fun evolve(start: EvolutionState<DNA, G>) = runBlocking {
        val interceptedStart = interceptor.before(start)
        val evolution = startEvolution(interceptedStart)
        val evaluatedPopulation = evaluate(evolution)
        val offspring = selectOffspring(evaluatedPopulation)
        val survivors = selectSurvivors(evaluatedPopulation)
        val alteredOffspring = alter(offspring, evolution)
        val nextPopulation = survivors + alteredOffspring.population
        val pop = evaluate(EvolutionState(nextPopulation, generation), true)
        evolutionResult = EvolutionResult(optimizer, pop, ++generation)
        val afterResult = interceptor.after(evolutionResult)
        afterResult
    }

    class Builder<DNA, G : Gene<DNA, G>>(
        private val fitnessFunction: (Genotype<DNA, G>) -> Double,
        private val genotype: Genotype.Factory<DNA, G>,
    ) {
        fun build() = Engine(...)
        ...
    }
}
```

In the provided code, the function `runBlocking` harnesses *coroutines* to allow parallel computation.

It's worth noting that two distinct `evolve` functions exist:

- `evolve()` oversees the entire evolutionary procedure.
- `evolve(start: EvolutionState<DNA, G>)` orchestrates a singular iteration of the evolutionary process, commencing from a specified `EvolutionState`.

The supplementary `Builder` class streamlines the engine configuration process. Although its utilization is recommended for engine instantiation, it remains optional, ensuring flexibility for users.

Finally, the `Evolver` interface is designed to accommodate varying engine implementations, with the `Engine` class being the exclusive existing implementation to date.

Termination Criteria The algorithm employs a set of `Limit` objects to define its termination criteria. These objects contain the logic to determine when the algorithm should halt, thus permitting multiple criteria definitions without requiring engine-specific awareness of their details.

The structure of the `Limit` interface is shown in **??** on page ??.

Listing 4.7– Limit interface

```
interface Limit {
    operator fun invoke(engine: Engine<*, *>): Boolean
}
```

Within this interface, the exclusive function is the `invoke` operator. This design lets the `Limit` object function as an invokable entity, as demonstrated in line 5 of **??** on page ??.

Currently, the framework incorporates four predefined termination criteria: 1. `GenerationCount`: halts after a designated number of generations. 2. `SteadyGenerations`: halts after a certain count of non-improving generations. 3. `TargetFitness`: ceases once a defined fitness value is attained. 4. `Matcher`: stops when a predetermined predicate is met.[1]

4.3.3 Evolution Listeners

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

[1] Note that all criteria fundamentally use the `Matcher` class.

4.4 Genetic Operators

Genetic operators are the driving force behind the evolution process in any genetic algorithm or genetic programming paradigm. These operators stimulate change in the population and guide the search towards optimal solutions. This section illuminates the fundamental genetic operators incorporated within *Keen*, categorized into *Selection*, *Mutation*, and *Crossover*.

Selection operators guide the evolutionary process by determining which individuals from the population will contribute to the next generation. These mechanisms often favor individuals that exhibit superior fitness, thereby promoting the inheritance of advantageous traits.

Mutation operators introduce randomness into the population by modifying individual genes or gene sequences. This mechanism ensures genetic diversity, aiding the population to explore a broader search space and escape local optima.

Crossover, also known as recombination, is another crucial genetic operator. It combines the genetic material from a set of parent individuals to generate offspring, simulating the biological process of sexual reproduction.

Each of these categories encompasses several techniques, providing *Keen* with a versatile repertoire of genetic operators. The subsequent subsections provide a comprehensive discussion on each of these operators, elucidating their working principles and application contexts.

4.4.1 Word Guessing Problem

The ***Word Guessing Problem*** is a simple example that illustrates the application of genetic operators. The problem statement is as follows: given a target word, the objective is to evolve a population of strings that match the target word. The fitness of each individual is determined by the number of characters that match the target word. The operators will be tested against a set of randomly generated target words, each with a length n between 1 and 100 characters.

4.4.2 Selection

Building upon the concepts presented in definition 2.4 on page 15, the ***selector operator*** is mathematically represented as $\Sigma : \mathbb{P} \times \mathbb{N} \times \cdots \rightarrow \mathbb{P}$. At its core, the selector operator processes a given population $\mathbb{P}$, selects n individuals based on specific criteria, and outputs a subset of the initial population. To ensure adaptability, the design challenge centers around establishing a versatile interface to cater to diverse selection strategies.

A promising approach consists of introducing an interface encompassing a method that requires the population and desired number of individuals as inputs. We can incorporate any supplementary parameters as properties of a specific selector object. This conceptualization can be defined as:

Listing 4.1: Selector interface

```
interface Selector<DNA, G : Gene<DNA, G>> {
  operator fun invoke(
      population: Population<DNA, G>,
      count: Int,
      optimizer: PhenotypeOptimizer<DNA, G>
  ): Population<DNA, G>
}
```

In this model, the `Selector` interface gets parameterized according to the `DNA` and `Gene` types. The incorporated `invoke` method explicitly mentions the population, the individual count, and an instance of the `PhenotypeOptimizer` class. Thanks to Kotlin's capacity for operator overloading, this framework allows direct function invocation through the selector object, depicted as `selector(...)`.[2]

[2] Syntactic sugar for `selector.invoke(...)`.

We introduce an additional parameter, `PhenotypeOptimizer`, which signifies the optimization strategy employed to facilitate selection, be it maximization or minimization.[3]

> **Remark.** *To add depth to our architecture, the `Selector` interface seamlessly integrates with the `AbstractSelector` abstract class. This integration ensures foundational functionalities for all selector types, while also allowing developers to design specialized selectors by extending this class.*

Random Selector

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Roulette Wheel Selector

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense

[3] Both these strategies - maximization and minimization - are integral components of the *Utility* module in *Keen*. Additionally, the `PhenotypeOptimizer` interface can be modified to support bespoke optimization techniques.

like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Tournament Selector

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters

are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

4.4.3 Mutation

Bit Flip Mutator

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Random Mutator

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Swap Mutator

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Inversion Mutator

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

4.4.4 Crossover

Combine Crossover

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text

should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Mean Crossover

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Ordered Crossover (OX)

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text

should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Partially Mapped Crossover (PMX)

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Position Based Crossover (PBX)

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Single-Point Crossover

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

4.5 Genetic Programming

4.5.1 Primitive Set

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

4.5.2 Genetic Operators

Subtree Crossover

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Point Mutation

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text

should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Subtree Mutation

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

4.6 Parallelism

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind

text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

4.7 Extensibility

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text

should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

4.8 Conclusion

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Chapter 5

Case Study: Real Function Optimization

5.1 Introduction

Real function optimization is a prevalent task in numerous fields, from *data science* and *machine learning* (ML) to *operations research* and *engineering*. It presents a common class of challenges that can be effectively addressed using GAs.

This chapter embarks on a practical exploration of *Keen*. We aim to illustrate the robust capabilities of *Keen* by employing it to solve various classic optimization problems, the details of which are provided in appendix B on page 93.

Please note that this chapter doesn't intend to delve into the formulation of these optimization problems; the discussion is primarily centered around how *Keen* can be leveraged as a solution tool. For a comprehensive understanding of *Keen*, its design, and its features, refer to chapter 4 on page 49.

The ensuing sections will navigate through the problem descriptions, the solutions employed using *Keen*, the corresponding results, and the consequent analysis. In the following sections, we will systematically explore and demonstrate the efficiency and versatility of *Keen* in tackling a range of optimization problems.

5.2 Problem Description

The primary task in this chapter revolves around the optimization of several real functions. These functions, which can be found in appendix B on page 93, are selected based on their complexity and their widespread use in the field of optimization. We aim to find the **global minimum** for each of these functions, a problem known as global optimization.

In the context of this case study, the search space is defined as the set of all possible solutions to the optimization problem. Since we are dealing with real-valued functions, the search space will be a subset of the set of real numbers, $\mathbb{R}$. The exact subset that forms the search space can vary depending on the function we are optimizing. For example, the *Cross-in-Tray* function has a search space of $[-10,\ 10]^2$, whereas the *Easom* function has a search space of $[-100,\ 100]^2$.

Given the nature of our functions and their respective search spaces, it is vital to understand some fundamental concepts related to sets, cardinalities, and functions. These concepts will not only help us better comprehend the vastness and complexity of the search space but will also lay the groundwork for the use of *Keen* in this context.

The following definitions, theorems, and corollaries will elucidate the necessary background.

Definition 5.1 (Set cardinality inequality)**.** *For any two sets A and B, if there exists an injective function $f : A \to B$, then $|A| \leq |B|$.*

Theorem 5.1 (Schröder–Bernstein theorem)**.** *Given two sets A and B, if there exist two injective functions $f : A \to B$ and $g : B \to A$, then there exists a bijective function $h : A \to B$.*

The proof for theorem 5.1 is attributed to J. König [1].

Corollary 5.1.1. *For any two sets A and B, if $|A| \leq |B|$ and $|B| \leq |A|$, then $|A| = |B|$.*

Proof. The conditions $|A| \leq |B|$ and $|B| \leq |A|$ imply that there exist injective functions $f : A \to B$ and $g : B \to A$, respectively. By the Schröder–Bernstein Theorem (theorem 5.1 on the preceding page), the existence of these injective functions guarantees a bijective function $h : A \to B$.

A bijective function is one that is both injective and surjective. This means that every element of A is mapped to a unique element in B and every element of B is the image of some element in A.

Therefore, there can't be more elements in A than in B (or vice versa), as this would contradict the surjectivity or the injectivity of the function h. Thus, by the definition of set cardinality (definition 5.1 on the previous page), we conclude that $|A| = |B|$. □

Theorem 5.2. *Let $(x, y) \in \mathbb{R}^2$ with $x \neq y$. Then, the cardinality of the interval $[x, y]$, denoted as $|[x, y]|$, is the same as the cardinality of $\mathbb{R}$. Thus, $|[x, y]|$ is uncountable.*

Proof. Let us define a function $f : [x, y] \to \mathbb{R}$ defined as $f(z) = \tan\left(\frac{\pi(z-x)}{y-x} - \frac{\pi}{2}\right)$ which maps the interval $[x, y]$ bijectively onto $\mathbb{R}$.

This means that there is an injective function $f : [x, y] \to \mathbb{R}$ and an injective function $g : \mathbb{R} \to [x, y]$.

By the definition of cardinality inequality (definition 5.1 on the preceding page), we conclude that $|[x, y]| \leq |\mathbb{R}|$ and $|\mathbb{R}| \leq |[x, y]|$.

By theorem 5.1 on the previous page, we conclude that $|[x, y]| = |\mathbb{R}|$. □

These results make it clear that, in most cases, the search space for our real function optimization problems will have an uncountable number of potential solutions. This vastness and complexity underline the necessity for robust and efficient optimization methods like those provided by *Keen*.

5.3 Solution

To solve this problem, we can use a GA with a double-valued chromosome. The chromosome's genes will represent the x_i values, and the fitness function will be the function we're trying to optimize.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

5.4 Results

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense

like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

5.5 Conclusion

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Chapter 6

Case Study: Knapsack Problem

6.1 Introduction

The knapsack problem stands as a classic optimization dilemma, posing the task of determining the optimal combination of items with given weights and values while adhering to a weight constraint. Its practical relevance spans diverse fields, making it an essential study in the realm of optimization.

This chapter delves into the two prominent variants of the knapsack problem: the ***unbounded*** and the ***0-1*** knapsack. The former allows an infinite number of each item, whereas the latter restricts each item to a singular instance. Both versions are analyzed in depth, elucidating their inherent complexities and solutions.

6.2 Problem Description

The ***Knapsack Problem*** is a canonical combinatorial optimization problem. It involves selecting items, each characterized by a specific weight and value, to maximize the total value while adhering to a weight constraint.

The goal is to ascertain the optimal subset of items (maximize profit) for the knapsack, ensuring the total weight remains under the knapsack's capacity.

Two primary variants of the Knapsack Problem exist: ***0-1 Knapsack*** ($K_{0/1}$)[5, 28, 30] and ***Unbounded Knapsack*** (K_*) [12, 28].

6.2.1 0-1 Knapsack Problem

In this variant, each item can be taken once or disregarded. This constraint can be expressed as $x_i \in \{0, 1\}$. The mathematical formulation is:

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{n} v_i x_i \\ \text{subject to} \quad & \sum_{i=1}^{n} w_i x_i \leq W \\ & x_i \in \{0, 1\} \quad \forall i \in \{1, 2, \ldots, n\} \end{aligned} \tag{6.1}$$

Analyzing the problem's search space, we note that the potential combinations of selected items equate to a size of 2^n. Given that the ***Subset Sum Problem***, to which this problem is reducible, is NP-hard, $K_{0/1}$ inherits this complexity.

Example: Consider 8 items $(w,\, v) = \{(11,\, 1),\, (21,\, 11),\, (31,\, 21),\, (33,\, 23),\, (43,\, 33),\, (53,\, 43),\, (55,\, 45),\, (65,\, 55)\}$. For a knapsack with a capacity $W = 110$, the optimal selection consists of items 1, 3, 4, 5, and 6. This results in a cumulative value of 159 and a total weight of 109.

6.2.2 Unbounded Knapsack Problem

Here, there's no limit to the number of instances of each item that can be selected.

Formally, the problem can be expressed as:

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{n} v_i x_i \\ \text{subject to} \quad & \sum_{i=1}^{n} w_i x_i \leq W \\ & x_i \in \mathbb{N} \quad \forall i \in \{1, 2, \ldots, n\} \end{aligned} \tag{6.2}$$

It's easy to see that the search space is infinite, since there's no limit on the number of each item that can be selected. The problem is still NP-hard.

To evaluate the performance of the solution, we are going to use the benchmarks from [12].

6.3 Solution

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

6.4 Results

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

6.5 Conclusion

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Chapter 7

Case Study: Crash Reproduction

7.1 Introduction

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

7.2 Problem Description

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

7.3 Solution

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

7.4 Results

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

7.5 Conclusion

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text

should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Chapter 8

Conclusions

8.1 Summary

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

8.2 Contributions

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between

this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

8.3 Future Work

8.3.1 Genetic Algorithms

Genetic Operators

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

8.3.2 Genetic Programming

Primitive Set

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Genetic Operators

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters

are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

8.3.3 Multi-Objective Evolution

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

8.3.4 Neuroevolution

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

8.3.5 Coevolution

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

8.3.6 Evolution Strategy

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

8.3.7 Other evolutionary algorithms

Ant Colony Optimization

Artificial Immune Systems

Differential Evolution

Estimation of Distribution Algorithms

Evolutionary Programming

Grammatical Evolution

Particle Swarm Optimization

8.3.8 Crash Reproduction

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Appendix A

Glossary

In this appendix we present a glossary of terms used throughout this document that may be unfamiliar to the reader and are not defined in the main text.

A

Definition A.1 (Alteration)**.** *See section 2.2.4 on page 15*

Definition A.2 (Alterer)**.** *See definition 2.5 on page 16.*

Definition A.3 (Arity)**.** *The number of arguments a function takes, or the number of children a node has in a tree.*

C

Definition A.4 (Chromosome)**.** *Representation of a single column of genetic information of a candidate solution to a given optimization problem.*
Formally, a chromosome is a vector $\mathbf{c} = (g_1, g_2, \ldots, g_n)$*, where* g_i *is a gene (see definition A.7 on the following page).*

E

Definition A.5 (Ephemeral Constant)**.** *A constant that is randomly generated at the start of the program and remains constant throughout the execution of the program.*
This is used to represent constant values in the program.

Definition A.6 (Evolutionary computation)**.** *Family of algorithms for global optimization inspired by the process of natural selection.*
This typically involves processes mimicking natural selection, mutation, recombination, and survival of the fittest. The solutions to a problem are encoded as a set of "individuals" in a "population". Over multiple generations, these individuals are selected and modified (via genetic operators like crossover and mutation) in order to find better solutions.

G

Definition A.7 (Gene). *Representation of a single component of a candidate solution to a given optimization problem. Formally, for a multi-dimensional function f, a gene is an element g in the domain of f.*

Definition A.8 (Generation). *Number of iterations performed by an evolutionary algorithm.*

Definition A.9 (Genotype). *Representation of the full genetic information of a candidate solution to a given optimization problem.*
Formally, a genotype is a matrix $\mathbf{G} = (\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_n)$, where $\boldsymbol{c}_i$ is a chromosome (see definition A.4 on the previous page).

I

Definition A.10 (Individual). *A candidate solution to a given optimization problem.*
Formally, an individual is a pair $(\mathbf{G}, \mathbf{f})$, where $\mathbf{G}$ is the genotype (see definition A.9) and $\mathbf{f}$ is the fitness value of the individual.

M

Definition A.11 (Metaheuristics). *Problem-independent algorithmic method that yields a sufficiently good solution within reasonable time for an optimization problem, especially for complex problems where an exact solution is not crucial.*

Definition A.12 (Mutator). *See definition 2.7 on page 18.*

P

Definition A.13 (Parameter optimization). *Optimization problem where the solution is a set of parameters that optimize a given function.*

Definition A.14 (Phenotype). *Same as definition A.10.*

Definition A.15 (Population). *Set of candidate solutions to a given optimization problem.*

Definition A.16 (Program induction). *Inference of an algorithm or program featuring recursive calls or repetition control structures, starting from information that is known to be incomplete, called the evidence, such as positive and negative I/O examples or clausal constraints.*

S

Definition A.17 (Search space). *Set of all candidate solutions to a given optimization problem.*

Definition A.18 (Selector). *See definition 2.4 on page 15.*

V

Definition A.19 (Variadic function)**.** *Function that accepts a variable number of arguments.*

Appendix B

Test Functions for Optimization

This appendix contains the test functions used in the numerical experiments in chapter 5 on page 69.

B.1 Ackley Function

In the field of optimization, especially within evolutionary algorithms and swarm intelligence, the ***Ackley function*** serves as a prevalent benchmark function. Named after *David H. Ackley*, who introduced it during his research,[1] this function is particularly challenging for optimization algorithms due to its property of possessing a large number of local minima, despite having a single global minimum. This trait can often cause such algorithms to become entrapped in local minima.

Definition B.1 (Ackley Function)**.** *The **Ackley Function**, denoted as $f : \mathbb{R}^2 \to \mathbb{R}$, is mathematically expressed as:*

$$f(x,\, y) = -20\, e^{-0.2\sqrt{0.5(x^2+y^2)}} - e^{0.5\,[\cos(2\pi x)+\cos(2\pi y)]} + e + 20 \tag{B.1}$$

This function is evaluated within the range $x,\, y \in [-5,\, 5]$.

The global minimum of the Ackley function is located at $f(0,\, 0) = 0$. Visualizations of the Ackley function are depicted as a contour plot and a surface plot in fig. B.1.

(a) Contour plot of the Ackley Function

(b) Surface plot of the Ackley Function

Figure B.1: Illustrations of the Ackley Function with the global minimum indicated by a red dot.

[1] Ackley, D. H. (1987) "A connectionist machine for genetic hillclimbing", Kluwer Academic Publishers, Boston MA.

B.2 Beale Function

Unveiled by Beale in 1958[2], the Beale function is recognized for its multimodal characteristics and sharp peaks that define the domain's corners.

Definition B.2 (Beale Function)**.** *The **Beale Function**, symbolized as $f : \mathbb{R}^2 \to \mathbb{R}$, is expressed by the equation:*

$$f(x,\, y) = (1.5 - x + xy)^2 + (2.25 - x + xy^2)^2 + (2.625 - x + xy^3)^2 \tag{B.2}$$

It's evaluated within the domain $x,\, y \in [-4.5,\, 4.5]$.

The global minimum of the Beale function is found at $f(3,\, 0.5) = 0$. Both contour and surface plots of the Beale function are depicted in fig. B.2.

(a) Contour plot of the Beale Function

(b) Surface plot of the Beale Function

Figure B.2: Visual representation of the Beale Function

B.3 Booth Function

The Booth function is a quadratic test problem used in the optimization field, specifically tailored for algorithms that handle two-dimensional search spaces.

Definition B.3 (Booth Function)**.** *The **Booth function**, $f : \mathbb{R}^2 \to \mathbb{R}$, is defined as:*

$$f(x,\, y) = (x + 2y - 7)^2 + (2x + y - 5)^2 \tag{B.3}$$

where:

- *$x,\, y \in \mathbb{R}$ represent the decision variables.*

The global minimum of the Booth function is $f(1, 3) = 0$. The function is usually constrained to the square $[-10, 10]^2$. A contour plot and a surface plot of the Booth function are illustrated in fig. B.3 on the next page.

B.4 Bukin Function N.6

The ***Bukin function N.6***, a two-dimensional benchmark problem, is renowned for its inherent complexity and frequent utilization in assessing optimization algorithms. Notable for a sharply defined, deep valley, the Bukin function N.6 presents distinct hurdles for optimization techniques owing to its abrupt discontinuity and non-differentiability at $x = 0$.

[2] Beale, E. M. (1958). "On an Iterative Method for Finding a Local Minimum of a Function of More than One Variable".

Figure B.3: Booth Function

Definition B.4 (Bukin Function N.6). *The **Bukin function N.6**, given by $f : \mathbb{R}^2 \to \mathbb{R}$, is mathematically represented as:*

$$f(x, y) = 100\sqrt{|y - 0.01x^2|} + 0.01|x + 10| \tag{B.4}$$

The decision variables, $x, y \in \mathbb{R}$, typically have the prescribed domains: $-15 \leq x \leq -5$ and $-3 \leq y \leq 3$ respectively.

The function attains its global minimum at $(x, y) = (-10, 1)$, yielding a value of zero. The Bukin function N.6, due to its unique characteristics, offers a striking visualization. A profound ridge, extending diagonally across the domain, forms a distinguishing feature. The contour and surface plots illustrating the Bukin function N.6 are presented in fig. B.4.

(a) Contour plot of Bukin function N.6

(b) Surface plot of Bukin function N.6

Figure B.4: Contour and Surface Plots of Bukin Function N.6

B.5 The Cross-in-Tray Function

The Cross-in-Tray function, known for its utility in testing optimization algorithms, poses a challenge due to its numerous local minima and four identical global minima, thereby making it an effective benchmark for evaluating an algorithm's capability to escape local optima and locate global optima.

Definition B.5 (Cross-in-Tray Function)**.** *The **Cross-in-Tray Function**, denoted as a mapping from $\mathbb{R}^2 \to \mathbb{R}$, is formally defined as:*

$$f(x,\, y) = -0.0001 \left[\left| \sin(x)\sin(y)\exp\left(\left| 100 - \frac{\sqrt{x^2+y^2}}{\pi} \right| \right) \right| + 1 \right]^{0.1} \tag{B.5}$$

This function is usually evaluated over the domain $-10 \leq x, y \leq 10$.

This function exhibits four identical global minima located at $(\pm 1.349\,41,\, \pm 1.349\,41)$, with each of these points having the function value of $f(x,\, y) = -2.062\,61$.

In fig. B.5, contour and surface plots are presented to provide a visual understanding of the function's intricate topology.

(a) Contour plot showcasing the complex landscape of the Cross-in-Tray function. The red dot denotes the location of the global minimum.

(b) Surface plot providing a 3D representation of the Cross-in-Tray function, enhancing the visualization of its global and local minima.

Figure B.5: Contour and surface plots illustrating the topological complexity of the Cross-in-Tray function

B.6 Easom Function

The ***Easom function*** is a well-known unimodal benchmark function employed in the evaluation of optimization algorithms. It gains its name from Charles Easom and is distinctively recognized by its 'needle'-like global minimum. This feature presents a demanding task for optimization algorithms due to its confined optimal search space.

Definition B.6 (Easom Function)**.** *The **Easom function**, defined for $f : [-100,\, 100]^2 \to \mathbb{R}$, is formally expressed as:*

$$f(x,\, y) = -\cos(x)\cos(y)\exp\left(-((x-\pi)^2 + (y-\pi)^2)\right) \tag{B.6}$$

Here, x and y constitute the decision variables.

The global minimum of the Easom function is situated at the coordinates $f(\pi,\, \pi) = -1$.

The intricate structure and narrow optimal space of the Easom function are clearly revealed in its contour and surface plots.

B.7 Eggholder Function

The ***Eggholder function*** is a widely-used non-convex function in the field of optimization, particularly for benchmarking optimization algorithms. The function is notorious for its multitude of local minima, presenting a complex search space that challenges the robustness and capability of an optimization algorithm to locate the global minimum.

(a) Contour plot of the Easom function with the global minimum represented by the red dot

(b) Surface plot of the Easom function

(c) Contour plot of the Easom function in the vicinity of the global minimum, $[0,\ 2\pi]$.

(d) Surface plot of the Easom function in the vicinity of the global minimum, $[0,\ 2\pi]$.

Figure B.6: Contour and surface visualizations of the Easom function

Definition B.7 (Eggholder Function)**.** *Formally, the* ***Eggholder Function*** *is represented as a mapping* $\mathbb{R}^2 \to \mathbb{R}$*, and is mathematically defined as:*

$$f(x,\ y) = -(y+47)\sin\left(\sqrt{\left|\frac{x}{2}+(y+47)\right|}\right) - x\sin\left(\sqrt{|x-(y+47)|}\right) \tag{B.7}$$

where the typical domain for evaluation spans $-512 \leq x, y \leq 512$*.*

The Eggholder function's global minimum resides at $(512,\ 404.2319)$, delivering a function value of $f(x,\ y) = -959.6407$.

Visual representations of the function can enhance understanding of its complexity. Figure fig. B.7 on the following page offers both contour and surface plots of the Eggholder function.

B.8 Goldstein-Price Function

The ***Goldstein-Price function***, believed to be proposed by individuals named Goldstein and Price, is a challenging multimodal function recognized for its landscape densely populated with local minima. This function serves as a standard benchmark in the field of optimization, testing the efficacy of various algorithms. The precise origins of this function, however, remain elusive in academic literature.

(a) A contour plot of the Eggholder function, displaying its intricate landscape. The red dot signifies the location of the global minimum.

(b) A 3D surface plot of the Eggholder function, allowing a more comprehensive visualization of its local and global minima.

Figure B.7: Contour and surface visualizations of the Eggholder function, showcasing its intricate topology

Definition B.8 (Goldstein-Price Function)**.** *The **Goldstein-Price function**, denoted as $f : \mathbb{R}^2 \to \mathbb{R}$, is formally articulated as follows:*

$$\begin{aligned} f(x,\, y) = &\left[1 + (x + y + 1)^2 \cdot (19 - 14x + 3x^2 - 14y + 6xy + 3y^2)\right] \\ &\cdot \left[30 + (2x - 3y)^2 \cdot (18 - 32x + 12x^2 + 48y - 36xy + 27y^2)\right] \end{aligned} \tag{B.8}$$

The function's global minimum is located at $f(x^*,\, y^*) = 3$ with $(x^*,\, y^*) = (0,\, -1)$. Evaluations of the Goldstein-Price function are typically performed within the range $x,\, y \in [-2,\, 2]$.

(a) Contour plot.

(b) Surface plot.

Figure B.8: Visual representations of the Goldstein-Price function

B.9 Himmelblau's Function

The ***Himmelblau's function***, attributed to David Mautner Himmelblau,[3] is a significant benchmark function in the realm of optimization, renowned for its complex multi-modal nature. Himmelblau, an American engineer, made substantial contributions to systems engineering and optimization theory.

[3] Himmelblau, D. (1972). "Applied Nonlinear Programming". McGraw-Hill. ISBN 0-07-028921-2.

Definition B.9 (Himmelblau's Function). *The **Himmelblau's function**, symbolized as $f : \mathbb{R}^2 \to \mathbb{R}$, is formally described as:*

$$f(x,\, y) = (x^2 + y - 11)^2 + (x + y^2 - 7)^2 \tag{B.9}$$

Here, $x,\, y \in \mathbb{R}$ are the decision variables, with the domains $\{x \mid -10 \leq x \leq 10\}$ and $\{y \mid -10 \leq y \leq 10\}$.

Distinguished by its four minima, located at $(3,\, 2)$, $(-2.805118,\, 3.131312)$, $(-3.779310,\, -3.283186)$, and $(3.584428,\, -1.848126)$, all roughly equating to zero, this function showcases its complexity.

Figure B.9 portrays the multi-modal landscape of Himmelblau's function through contour and surface plots, underlining its inherent intricacy, and as such, its utility in the evaluation of optimization techniques.

(a) Contour plot of Himmelblau's function, the minima are signified by the red dots

(b) Surface plot of Himmelblau's function

Figure B.9: The detailed multi-modal structure of Himmelblau's function illustrated through contour and surface plots

B.10 Hölder Table Function

The ***Hölder Table function*** is a two-dimensional real-valued function employed in various branches of mathematical analysis. Due to its intriguing properties and complex topography, it serves as a valuable tool for function approximation and numerical analysis studies.

Definition B.10 (Hölder Table function). *The **Hölder Table function**, designated as $f : \mathbb{R}^2 \to \mathbb{R}$, is described by the equation*

$$f(x,\, y) = -\left|\sin(x)\cos(y)\exp\left(\left|1 - \frac{\sqrt{x^2 + y^2}}{\pi}\right|\right)\right| \tag{B.10}$$

This function is defined for $-10 \leq x,\, y \leq 10$.

The Hölder Table function peaks globally at $f(x^*,\, y^*) = 19.208\,5$ for $x^* = \pm 8.05502$ and $y^* = \pm 9.66459$. Its intricately structured landscape can be vividly illustrated through contour and surface plots, as depicted in fig. B.10 on the next page.

B.11 Lévi Function N.13

The ***Lévi function N.13*** is a noteworthy two-dimensional function frequently employed in the field of optimization algorithms for performance testing. Its complex, sinuous landscape, teeming with numerous local minima, presents a significant challenge to optimization procedures.

(a) Contour visualization of the Hölder Table function. The global minimum points are denoted by red dots.

(b) Three-dimensional surface representation of the Hölder Table function.

Figure B.10: The Complex Topography of the Hölder Table Function: Contour and Surface Representations

Definition B.11 (Lévi Function N.13)**.** *The **Lévi function N.13**, denoted as $f : \mathbb{R}^2 \to \mathbb{R}$, is formally defined as follows:*

$$f(x, y) = \sin^2(3\pi x) + (x-1)^2 \cdot (1 + \sin^2(3\pi y)) + (y-1)^2 \cdot (1 + \sin^2(2\pi y)) \tag{B.11}$$

where $x, y \in \mathbb{R}$ are the decision variables.

The Lévi function N.13 finds its global minimum at $f(1, 1) = 0$. The complex topology of this function is visually captured in the contour and surface plots shown in Figure B.11.

(a) Contour plot of the Lévi function N.13

(b) Surface plot of the Lévi function N.13

Figure B.11: Contour and Surface Representations of the Lévi Function N.13

B.12 Matyas Function

The ***Matyas function***, known for its simplicity and convex nature, is a standard test problem in the field of optimization algorithms. Despite its apparent simplicity, it provides invaluable insights into an algorithm's performance and behavior.

Definition B.12 (Matyas Function)**.** *The **Matyas function**, denoted as $f : \mathbb{R}^2 \to \mathbb{R}$, is formulated as:*

$$f(x,\, y) = 0.26(x^2 + y^2) - 0.48xy \tag{B.12}$$

where:

- $x,\, y \in \mathbb{R}$ *denote the decision variables.*

The Matyas function reaches its global minimum at the origin, with $f(0, 0) = 0$. The contour and surface visualizations of the Matyas function, offering perspectives on its topographical attributes, are presented in fig. B.12.

(a) Contour plot of the Matyas function with a red dot indicating the global minimum

(b) Surface plot of the Matyas function

Figure B.12: Contour and Surface Visualizations of the Matyas Function

B.13 McCormick Function

The field of optimization research frequently employs the ***McCormick function*** as an insightful benchmark function. This function owes its name to the researcher, Garth P. McCormick, who first employed it in his seminal study on factorable nonconvex programs.[4] The McCormick function is renowned for its oscillatory properties and its complex landscape featuring a single global minimum amidst numerous local minima. This complex landscape can pose a significant challenge to optimization algorithms, as they risk being ensnared in the local minima.

Definition B.13 (McCormick Function)**.** *The **McCormick Function**, symbolized as $f : \mathbb{R}^2 \to \mathbb{R}$, is expressed mathematically as follows:*

$$f(x,\, y) = \sin(x + y) + (x - y)^2 - 1.5x + 2.5y + 1 \tag{B.13}$$

The McCormick function is typically evaluated within the range $x,\, y \in [-1.5,\, 4]$ and $[-3,\, 4]$ respectively.

The global minimum of the McCormick function is found at $f(-0.54719,\, -1.54719) \approx -1.9133$. Visualizations of the McCormick function in the form of a contour plot and a surface plot are provided in fig. B.13 on the next page.

B.14 Rastrigin Function

The Rastrigin function, first proposed by Rastrigin in 1974[5], is a prominent non-convex function utilized as a benchmark for optimization algorithms. This function is a classic example of non-linear multimodal optimization problems, known for their

[4] McCormick, Garth P. "Computability of Global Solutions to Factorable Nonconvex Programs: Part I — Convex Underestimating Problems." Mathematical Programming 10, no. 1 (December 1976): 147–75. https://doi.org/10.1007/BF01580665.

[5] Rastrigin, L. A. (1974). "Systems of extremal control." Mir, Moscow.

(a) Contour plot of the McCormick Function. The location of the global minimum is designated by a red dot.

(b) Surface plot of the McCormick Function

Figure B.13: Visual representations of the McCormick Function.

complexity due to the abundance of local minima.

Definition B.14 (Rastrigin Function)**.** *The **Rastrigin function**, denoted as $f : \mathbb{R}^n \to \mathbb{R}$, is formulated as:*

$$f(\mathbf{x}) = An + \sum_{i=1}^{n} \left[\mathbf{x}_i^2 - A\cos(2\pi\mathbf{x}_i)\right] \tag{B.14}$$

Here,

- *n is the number of dimensions.*
- *$\mathbf{x}$ is a vector composed of n real numbers.*
- *A is a constant, typically set to 10.*

The global minimum of the Rastrigin function is situated at $\mathbf{x}^* = (0, \ldots, 0)$, with $f(\mathbf{x}^*) = 0$. It's particularly challenging due to its numerous local minima, distributed regularly throughout the search space. The function is predominantly evaluated within the hypercube $\mathbf{x} \in [-5.12,\ 5.12]^n$.

Visualizations of the Rastrigin function for $n = 2$ are presented in fig. B.14, showcasing both the contour plot and the 3D surface plot.

(a) Contour plot of the Rastrigin Function.

(b) 3D surface plot of the Rastrigin Function.

Figure B.14: Visualizations of the Rastrigin Function for $n = 2$, with the global minimum marked by a red dot.

B.15 The Rosenbrock Function

Introduced by Howard H. Rosenbrock in 1960[6], the Rosenbrock function is a non-convex function that serves as a benchmark for optimization algorithms. Despite its simplicity, it poses a challenge due to its characteristic landscape, a long, narrow, parabolic-shaped flat valley. While finding the valley is relatively simple, converging to the global minimum within it is notably difficult.

Definition B.15 (Rosenbrock function)**.** *The **Rosenbrock function**, denoted as $f : \mathbb{R}^n \to \mathbb{R}$, is defined by the following equation:*

$$f(\mathbf{x}) = \sum_{i=1}^{n-1} \left[100(\mathbf{x}_{i+1} - \mathbf{x}_i^2)^2 + (1 - \mathbf{x}_i)^2\right] \tag{B.15}$$

In this equation:

- *n is the number of dimensions.*
- *$\mathbf{x}$ is a vector composed of n real numbers.*

The Rosenbrock function's global minimum is at $f(\mathbf{x}^*) = 0$, corresponding to the point $\mathbf{x}^* = (1, \ldots, 1)$. To illustrate the function's behavior for $n = 2$, fig. B.15 presents a contour plot and a surface plot.

(a) Contour plot of the Rosenbrock function.

(b) Surface plot of the Rosenbrock function.

Figure B.15: Rosenbrock Function for $n = 2$, the red dot indicates the global minimum.

B.16 Schaffer Function N.2

The ***Schaffer Function N.2*** is a well-established mathematical function frequently used in the realm of algorithm testing. It serves as a performance benchmark for a wide array of optimization algorithms, particularly those grounded in evolutionary computation and swarm intelligence principles.

Definition B.16 (Schaffer Function N.2)**.** *The Schaffer Function N.2 is defined over a two-dimensional domain, mapping $\mathbb{R}^2$ to $\mathbb{R}$. The function, denoted as $f(\mathbf{x})$, is expressed as follows:*

$$f(\mathbf{x}) = 0.5 + \frac{\sin^2(\sqrt{x_1^2 + x_2^2}) - 0.5}{(1.0 + 0.001 \cdot (x_1^2 + x_2^2))^2}$$

Here, $\mathbf{x} = (x_1, x_2)$ represents a point in the two-dimensional domain, with x_1, x_2 each ranging within the interval [-100, 100].

[6] Rosenbrock, H.H. (1960). "An automatic method for finding the greatest or least value of a function". The Computer Journal. 3 (3): 175-184. doi:10.1093/comjnl/3.3.175. ISSN 0010-4620.

The Schaffer Function N.2 achieves its global minimum at $f(0,\ 0) = 0$. A visual exploration of this function can be enhanced by both contour and surface plots, providing a clearer understanding of its global minimum and topological characteristics.

fig. B.17 illustrate the *Schaffer Function N.2*.

(a) Contour plot of the Schaffer Function N.2

(b) Surface plot of the Schaffer Function N.2

Figure B.16: a

(a) Close-up contour plot of the Schaffer Function N.2 in the vicinity of the global minimum.

(b) Close-up surface plot of the Schaffer Function N.2 in the vicinity of the global minimum.

Figure B.17: Visualization of the Schaffer Function N.2. The contour and surface plots illustrate the function's topology, with the global minimum denoted by a red dot. The close-up contour and surface plots provide a more precise view of the global minimum and the immediate surroundings.

B.17 Schaffer Function N.4

The ***Schaffer Function N.4*** is a mathematical function used as a performance test problem for optimization algorithms. It is particularly suited for testing algorithms that need to optimize complex, non-linear problems with many local minima.

Definition B.17 (Schaffer Function N.4)**.** *Schaffer Function N.4 is defined as follows:*
For $x \in \mathbb{R}^2$, the function is given by:

$$f(x) = 0.5 + \frac{\cos^2\left(\sin\left[\left|x^2 - y^2\right|\right]\right) - 0.5}{\left[1 + 0.001\left(x^2 + y^2\right)\right]^2}$$

Where:

- *x and y are decision variables.*
- *The function has four global minima of 0.292579, located at $0,\ \pm 1.25313$ and $\pm 1.25313,\ 0$.*

The characteristic feature of Schaffer N.4 is its landscape with several local minima and four global minima. This structure poses a significant challenge to optimization algorithms, especially those prone to being trapped in local minima.

Figure B.18 provides a visualization of the Schaffer Function N.4. The contour and surface plots show the function's topology, with the red dots indicating the locations of the global minima. The close-up contour and surface plots offer a detailed view of the global minima and their immediate surroundings, highlighting the intricacy of the function's landscape.

(a) Contour plot of the Schaffer Function N.4

(b) Surface plot of the Schaffer Function N.4

(c) Close-up contour plot of the Schaffer Function N.4 in the vicinity of the global minimum.

(d) Close-up surface plot of the Schaffer Function N.4 in the vicinity of the global minimum.

Figure B.18: Visualization of the Schaffer Function N.4. The contour and surface plots illustrate the function's topology, with the global minimum denoted by a red dot. The close-up contour and surface plots provide a more precise view of the global minimum and the immediate surroundings.

B.18 The Sphere Function

The sphere function serves as a prominent benchmark problem in the realm of optimization algorithms. This convex function's simplicity and well-defined nature make it ideal for gauging the performance of such algorithms.

Definition B.18 (Sphere Function)**.** *The **sphere function**, denoted as $f : \mathbb{R}^n \to \mathbb{R}$, is formally described as:*

$$f(\mathbf{x}) = \sum_{i=1}^{n} \mathbf{x}_i^2 \tag{B.16}$$

Herein:

- $n \in \mathbb{N}$ *signifies the number of dimensions.*
- $\mathbf{x}_i \in \mathbb{R}$ *represents the i-th element of the vector* $\mathbf{x}$.

The global minimum of the sphere function occurs at $f(\mathbf{x}^*) = 0$ with the input vector $\mathbf{x}^* = (0, \ldots, 0)$. The contour and surface plots showcasing the behavior of the sphere function for a two-dimensional input ($n = 2$) are illustrated in fig. B.19.

(a) Contour plot of the Sphere Function.

(b) Surface plot of the Sphere Function.

Figure B.19: Visual Representation of the Sphere Function with $n = 2$

B.19 Styblinski-Tang function

The ***Styblinski-Tang function*** is a non-convex function used as a performance test problem for optimization algorithms. This function is known for its complex and intricate landscape, presenting numerous local minima that pose challenges to optimization algorithms.

Definition B.19 (Styblinski-Tang function)**.** *The Styblinski-Tang function is defined in two dimensions, with the variables x and y.*

$$f(\mathbf{x}) = \frac{1}{2} \sum_{i=1}^{n} \left(\mathbf{x}_i^4 - 16\mathbf{x}_i^2 + 5\mathbf{x}_i \right) \tag{B.17}$$

where x and y are any real numbers.

The intriguing properties of this function have made it a common benchmark in the study of algorithm performance, particularly in the field of evolutionary computation and swarm intelligence.

(a) Contour plot of the Styblinski-Tang function. The intricate patterns show the complex landscape of the function. The red dot signifies the global minimum.

(b) Three-dimensional surface plot of the Styblinski-Tang function. The plot illustrates the numerous local minima and the complex topography of the function.

Figure B.20: Visualizations of the Styblinski-Tang function.

B.20 Three-Hump Camel Function

The ***Three-Hump Camel function***, colloquially termed as the camel-back function, serves as a standard benchmark in the optimization algorithms testing landscape. This two-dimensional function earns its name from the characteristic tri-modal hump visual pattern it presents in a three-dimensional space, bearing resemblance to a camel's back.

Definition B.20 (Three-Hump Camel Function)**.** *The **Three-Hump Camel function**, depicted as $f : \mathbb{R}^2 \to \mathbb{R}$, is formally expressed as:*

$$f(x, y) = 2x^2 - 1.05x^4 + \frac{x^6}{6} + xy + y^2 \tag{B.18}$$

where $-5 \leq x \leq 5$ and $-5 \leq y \leq 5$.

The Three-Hump Camel function exhibits its global minimum at the origin, $f(0, 0) = 0$. Although its form appears straightforward, the function's multiple local optima pose considerable challenges for optimization algorithms, making it an excellent test case.

(a) Contour plot of the Three-Hump Camel function. The red dot signifies the global minimum.

(b) Surface plot showcasing the characteristic tri-modal humps of the Three-Hump Camel function.

Figure B.21: Contour and surface visualizations of the Three-Hump Camel function

Appendix C

Additional Listings

This appendix contains additional listings of the source code used in this thesis that are not essential to the understanding of the thesis. The listings are included here for completeness.

Calculation of $|\mathbb{T}_{\leq 5}(\mathcal{T}, \mathcal{F})|$ for $\mathcal{T} = \{x, c\}$ and the set $\mathcal{F} = \{+, -, \times, /, \sin, \cos, \text{pow}\}$ using the *Julia* programming language.

```
arities = [2, 2, 2, 2, 1, 1, 2] # A({+, −, ×, /, sin, cos, pow}) = {2, 2, 2, 2, 1, 1, 2}
terminals_size = 8  # |T| = |{x, 1, 2, 3, 4, 5, 6, 7}| = 8
# |T_{≤h}|
t_leq(h::Int)::Int128 = if h == 0 # |T| if h = 0
  terminals_size
else  # (Σ_{h=0}^{h−1} Σ_{f∈F} |T_h|^{A(f)}) + |T| if h > 0
  c_sum = terminals_size
  for i = 0:h - 1
    c_sum = c_sum + sum(t(i) .^ arities)
  end
  c_sum
end
# |T_h|
t(h::Int)::Int128 = if h == 0 # |T| if h = 0
  terminals_size
else  # Σ_{f∈F} |T_{h−1}|^{A(f)} if h > 0
  sum(t(h - 1) .^ arities)
end
res = t_leq(5)  # |T_{≤5}(T, F)|
```

Minimal implementation of the *One Max* problem using the *Keen* framework.

```
fun main() {
    Core.random = Random(11)
    val result = engine(
        { gt: Genotype<Boolean, BoolGene> -> gt.flatten().count { it }.toDouble() },
        genotype { chromosome { booleans { size = 20; truesProbability = 0.5 } } }) {
        populationSize = 20
        selector = TournamentSelector(sampleSize = 3)
        alterers = listOf(
          BitFlipMutator(probability = 0.03),
          SinglePointCrossover(probability = 0.2)
```

```
        )
        limits = listOf(TargetFitness(20.0))
        }.evolve()
    println("Target fitness reached at generation ${result.generation}")
    println("Best individual is ${result.best.genotype}")
    println("with fitness ${result.best.fitness}")
}
```

Listing C.1: *Room Scheduling* problem using the *Keen* framework.

```
data class Meeting(val start: Int, val end: Int)

private val meetings =
    listOf(
        Meeting(1, 3),
        Meeting(2, 3),
        Meeting(5, 6),
        Meeting(7, 9),
        Meeting(4, 7),
        Meeting(8, 10),
        Meeting(2, 7),
        Meeting(3, 4),
        Meeting(1, 5),
        Meeting(3, 6),
        Meeting(4, 5)
    )

private fun fitnessFn(genotype: Genotype<Int, IntGene>): Double {
    // Create a map to represent the rooms, where the key is the room number and the value is
    // a list of meetings in that room.
    val rooms = meetings.groupBy { genotype.chromosomes[meetings.indexOf(it)].genes[0].dna }
    // Calculate the number of conflicts in each room.
    val conflicts = rooms.values.sumOf { meetingList ->
        val table = IntArray(10) // Create an array to represent the time slots in a day.
        meetingList.forEach { meeting ->
            // Increment the time slots for each meeting.
            for (i in meeting.start until meeting.end) {
                table[i]++
            }
        }
        // Count the number of time slots with more than one meeting.
        table.count { it > 1 }
    }
    // The fitness score is the number of rooms used plus the number of conflicts.
    // We add 1 to the number of rooms used to avoid a fitness score of 0.
    return rooms.size.toDouble() + conflicts
}

fun main() {
    // Create a genetic algorithm engine with the fitness function and genotype for the problem.
    val engine = engine(::fitnessFn, genotype {
        repeat(meetings.size) {
            chromosome {
```

```
                ints { size = 1; range = meetings.indices.first to meetings.indices.last }
            }
        }
    }) {
        // Set the parameters for the genetic algorithm.
        populationSize = 100
        optimizer = FitnessMinimizer()
        alterers = listOf(Mutator(0.06), SinglePointCrossover(0.2))
        limits = listOf(SteadyGenerations(20), GenerationCount(100))
        listeners = listOf(EvolutionSummary(), EvolutionPlotter())
    }
    // Evolve the population and get the best result.
    val result = engine.evolve()
    // Print the statistics of the genetic algorithm.
    println(engine.listeners.first())
    // Create a schedule based on the best genotype.
    val schedule = MutableList(result.best.genotype.size) { mutableListOf<Meeting>() }
    meetings.forEachIndexed { index, meeting ->
        val room = result.best.genotype.chromosomes[index].genes[0].dna
        schedule[room].add(meeting)
    }
    // Print the schedule for each room.
    schedule.forEachIndexed { index, room ->
        println("Room $index: $room")
    }
    // Display a plot of the fitness values over time.
    (engine.listeners.last() as EvolutionPlotter).displayFitness()
}
```

Bibliography

[1] J. König. "Sur La Théorie Des Ensembles". In: ***Comptes Rendus Hebdomadaires des Séances de l'Académie des Sciences.*** 143 (1906), pp. 110–112.

[2] A. M. Turing. "I.—COMPUTING MACHINERY AND INTELLIGENCE". In: ***Mind*** LIX.236 (Oct. 1, 1950), pp. 433–460. ISSN: 1460-2113, 0026-4423. DOI: 10.1093/mind/LIX.236.433. URL: https://academic.oup.com/mind/article/LIX/236/433/986238 (visited on 06/10/2023).

[3] H. H. Rosenbrock. "An Automatic Method for Finding the Greatest or Least Value of a Function". In: ***The Computer Journal*** 3.3 (Mar. 1, 1960), pp. 175–184. ISSN: 0010-4620, 1460-2067. DOI: 10.1093/comjnl/3.3.175. URL: https://academic.oup.com/comjnl/article-lookup/doi/10.1093/comjnl/3.3.175 (visited on 07/04/2023).

[4] Nils Aall Barricelli. "Numerical Testing of Evolution Theories: Part I Theoretical Introduction and Basic Tests". In: ***Acta Biotheoretica*** 16.1-2 (Mar. 1962), pp. 69–98. ISSN: 0001-5342, 1572-8358. DOI: 10.1007/BF01556771. URL: http://link.springer.com/10.1007/BF01556771 (visited on 06/10/2023).

[5] Sartaj Sahni. "Approximate Algorithms for the 0/1 Knapsack Problem". In: ***Journal of the ACM (JACM)*** 22.1 (1975), pp. 115–124.

[6] John H. Holland. ***Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence***. 1st MIT Press ed. Complex Adaptive Systems. Cambridge, Mass: MIT Press, 1992. 211 pp. ISBN: 978-0-262-08213-6 978-0-262-58111-0.

[7] John R. Koza. ***Genetic Programming. 1: On the Programming of Computers by Means of Natural Selection***. Complex Adaptive Systems. Cambridge, Mass.: MIT Press, 1992. 819 pp. ISBN: 978-0-262-52791-0.

[8] Peter J Angeline and Jordan B Pollack. "The Evolutionary Induction of Subroutines". In: ***Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society***. Bloomington, Indiana, 1992, pp. 236–241.

[9] John R. Koza. ***Genetic Programming II: Automatic Discovery of Reusable Programs***. Complex Adaptive Systems. Cambridge, Mass: MIT Press, 1994. 746 pp. ISBN: 978-0-262-11189-8.

[10] Peter J. Angeline. "Genetic Programming and Emergent Intelligence". In: ***Advances in Genetic Programming***. Advances in Genetic Programming 1. Cambridge (Mass.) London: MIT press, 1994, pp. 75–97. ISBN: 978-0-262-11188-1.

[11] J.P. Rosca and D.H. Ballard. "Learning by Adapting Representations in Genetic Programming". In: ***Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence***. First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence. Orlando, FL, USA: IEEE, 1994, pp. 407–412. ISBN: 978-0-7803-1899-1. DOI: 10.1109/ICEC.1994.349916. URL: http://ieeexplore.ieee.org/document/349916/ (visited on 06/23/2023).

[12] R. Andonov, V. Poirriez, and S. Rajopadhye. "Unbounded Knapsack Problem: Dynamic Programming Revisited". In: ***European Journal of Operational Research*** 123.2 (June 2000), pp. 394–407. ISSN: 03772217. DOI: 10.1016/S0377-2217(99)00265-9. URL: https://linkinghub.elsevier.com/retrieve/pii/S0377221799002659 (visited on 08/09/2023).

[13] W. B. Langdon and Riccardo Poli. ***Foundations of Genetic Programming***. Berlin ; New York: Springer, 2002. 260 pp. ISBN: 978-3-540-42451-2.

[14] Kalyanmoy Deb, Pawan Zope, and Abhishek Jain. "Distributed Computing of Pareto-Optimal Solutions with Evolutionary Algorithms". In: ***Evolutionary Multi-Criterion Optimization***. Ed. by Carlos M. Fonseca et al. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2003, pp. 534–549. ISBN: 978-3-540-36970-7. DOI: 10.1007/3-540-36970-8_38.

[15] Kenneth A. De Jong. ***Evolutionary Computation: A Unified Approach***. 2006. ISBN: 978-0-262-25598-1. URL: https://ieeexplore.ieee.org/servlet/opac?bknumber=6267245.

[16] ***The MIT License***. Open Source Initiative. Oct. 31, 2006. URL: https://opensource.org/license/mit/ (visited on 07/26/2023).

[17] Riccardo Poli et al. ***A Field Guide to Genetic Programming***. [Morrisville, NC: Lulu Press], 2008. 233 pp. ISBN: 978-1-4092-0073-4.

[18] Lucas de P. Veronese and Renato A. Krohling. "Differential Evolution Algorithm on the GPU with C-CUDA". In: ***IEEE Congress on Evolutionary Computation***. IEEE Congress on Evolutionary Computation. July 2010, pp. 1–7. DOI: 10.1109/CEC.2010.5586219.

[19] Travis Desell et al. "An Analysis of Massively Distributed Evolutionary Algorithms". In: ***IEEE Congress on Evolutionary Computation***. IEEE Congress on Evolutionary Computation. July 2010, pp. 1–8. DOI: 10.1109/CEC.2010.5586073.

[20] Thé Van Luong, Nouredine Melab, and El-Ghazali Talbi. "GPU-based Island Model for Evolutionary Algorithms". In: ***Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation***. GECCO '10. New York, NY, USA: Association for Computing Machinery, July 7, 2010, pp. 1089–1096. ISBN: 978-1-4503-0072-8. DOI: 10.1145/1830483.1830685. URL: https://doi.org/10.1145/1830483.1830685 (visited on 08/08/2023).

[21] Thé Van Luong, Nouredine Melab, and El-Ghazali Talbi. "Parallel Hybrid Evolutionary Algorithms on GPU". In: ***IEEE Congress on Evolutionary Computation***. IEEE Congress on Evolutionary Computation. July 2010, pp. 1–8. DOI: 10.1109/CEC.2010.5586403.

[22] Xinjie Yu and Mitsuo Gen. ***Introduction to Evolutionary Algorithms***. Decision Engineering. London ; New York: Springer, 2010. 418 pp. ISBN: 978-1-84996-128-8 978-1-84996-129-5.

[23] Daniel Shiffman. ***The Nature of Code***. Version 1.0, generated December 6, 2012. s.l.: Selbstverl., 2012. 498 pp. ISBN: 978-0-9859308-0-6.

[24] Mateusz Guzek et al. "Multi-Objective Evolutionary Algorithms for Energy-Aware Scheduling on Distributed Computing Systems". In: ***Applied Soft Computing*** 24 (Nov. 1, 2014), pp. 432–446. ISSN: 1568-4946. DOI: 10.1016/j.asoc.2014.07.010. URL: https://www.sciencedirect.com/science/article/pii/S1568494614003408 (visited on 08/08/2023).

[25] Pawel Liskowski et al. "Comparison of Semantic-aware Selection Methods in Genetic Programming". In: ***Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation***. GECCO '15: Genetic and Evolutionary Computation Conference. Madrid Spain: ACM, July 11, 2015, pp. 1301–1307. ISBN: 978-1-4503-3488-4. DOI: 10.1145/2739482.2768505. URL: https://dl.acm.org/doi/10.1145/2739482.2768505 (visited on 06/21/2023).

[26] Clara Schmitt et al. "Half a Billion Simulations: Evolutionary Algorithms and Distributed Computing for Calibrating the Simpoplocal Geographical Model". In: ***Environment and Planning B: Planning and Design*** 42.2 (Apr. 1, 2015), pp. 300–315. ISSN: 0265-8135. DOI: 10.1068/b130064p. URL: https://doi.org/10.1068/b130064p (visited on 08/08/2023).

[27] Sergii Yaroshko and Svitlana Yaroshko. "Multithreaded Evolutionary Computing". In: ***2019 IEEE 2nd Ukraine Conference on Electrical and Computer Engineering (UKRCON)***. 2019 IEEE 2nd Ukraine Conference on Electrical and Computer Engineering (UKRCON). July 2019, pp. 1041–1045. DOI: 10.1109/UKRCON.2019.8879863.

[28] Alexandre Bergel. ***Agile Artificial Intelligence in Pharo: Implementing Neural Networks, Genetic Algorithms, and Neuroevolution***. For Professionals by Professionals. New York: Apress, 2020. 386 pp. ISBN: 978-1-4842-5384-7 978-1-4842-5383-0.

[29] Alexandre Bergel and Ignacio Slater Muñoz. "Beacon: Automated Test Generation for Stack-Trace Reproduction Using Genetic Algorithms". In: ***2021 IEEE/ACM 14th International Workshop on Search-Based Software Testing (SBST)***. 2021 IEEE/ACM 14th International Workshop on Search-Based Software Testing (SBST). Madrid, Spain: IEEE, May 2021, pp. 1–7. ISBN: 978-1-66544-571-9. DOI: 10.1109/SBST52555.2021.00007. URL: https://ieeexplore.ieee.org/document/9476241/ (visited on 06/07/2023).

[30] Zhengyuan Wang, Hui Zhang, and Yali Li. "Fast Polynomial Time Approximate Solution for 0-1 Knapsack Problem". In: ***Computational Intelligence and Neuroscience*** 2022 (Oct. 22, 2022), e1266529. ISSN: 1687-5265. DOI: 10.1155/2022/1266529. URL: https://www.hindawi.com/journals/cin/2022/1266529/ (visited on 08/09/2023).

[31] djakobovic. ***ECF - Evolutionary Computation Framework***. Mar. 13, 2023. URL: https://github.com/djakobovic/ECF (visited on 08/09/2023).

[32] Diego Giacomelli. ***GeneticSharp***. June 7, 2023. URL: https://github.com/giacomelli/GeneticSharp (visited on 06/08/2023).

[33] ***Kotlin Telegram Bot***. Kotlin Telegram Bot, Aug. 3, 2023. URL: https://github.com/kotlin-telegram-bot/kotlin-telegram-bot (visited on 08/03/2023).

[34] ***Kotlinx.Html***. Kotlin, Aug. 3, 2023. URL: https://github.com/Kotlin/kotlinx.html (visited on 08/03/2023).

[35] ***Pagmo***. European Space Agency, Aug. 8, 2023. URL: https://github.com/esa/pagmo2 (visited on 08/09/2023).

[36] Christian S. Perone. ***Perone/Pyevolve***. May 23, 2023. URL: https://github.com/perone/Pyevolve (visited on 06/08/2023).

[37] ***Pygmo***. European Space Agency, June 4, 2023. URL: https://github.com/esa/pygmo2 (visited on 06/08/2023).

[38] ***The Latest Programming Languages Statistics & Trends For 2023 • GITNUX***. Feb. 1, 2023. URL: https://blog.gitnux.com/programming-languages-statistics/ (visited on 07/21/2023).

[39] ***Android's Kotlin-first Approach***. Android Developers. URL: https://developer.android.com/kotlin/first (visited on 07/21/2023).

[40] ***Calling Java from Kotlin | Kotlin***. Kotlin Help. URL: https://kotlinlang.org/docs/java-interop.html (visited on 07/21/2023).

[41] ***Calling Kotlin from Java | Kotlin***. Kotlin Help. URL: https://kotlinlang.org/docs/java-to-kotlin-interop.html (visited on 07/21/2023).

[42] Fort Collins. "A Comparison of Genetic Sequencing Operators". In: (), p. 8.

[43] ***DEAP Documentation — DEAP 1.3.3 Documentation***. URL: https://deap.readthedocs.io/en/master/ (visited on 06/08/2023).

[44] ***Easy_ga/Src at Main · RubenRubioM/Easy_ga***. GitHub. URL: https://github.com/RubenRubioM/easy_ga (visited on 06/13/2023).

[45] ***ECJ***. URL: https://cs.gmu.edu/~eclab/projects/ecj/ (visited on 06/08/2023).

[46] ***EvolvingObjects: Welcome to Evolving Objects***. URL: https://eodev.sourceforge.net/eo/doc/html/index.html (visited on 06/08/2023).

[47] ***Genevo - Rust***. URL: https://docs.rs/genevo/latest/genevo/index.html (visited on 06/13/2023).

[48] ***GNU Lesser General Public License v3.0 - GNU Project - Free Software Foundation***. URL: https://www.gnu.org/licenses/lgpl-3.0.en.html (visited on 07/26/2023).

[49] ***Inspyred: Bio-inspired Algorithms in Python — Inspyred 1.0.1 Documentation***. URL: https://aarongarrett.github.io/inspyred/ (visited on 06/08/2023).

[50] ***Kotest | Kotest***. URL: https://kotest.io/ (visited on 08/03/2023).

[51] ***Kotlin Programming Language***. Kotlin. URL: https://kotlinlang.org/ (visited on 07/21/2023).

[52] ***Kotlin-Java Interop Guide***. Android Developers. URL: https://developer.android.com/kotlin/interop (visited on 07/21/2023).

[53] ***One Max Problem — DEAP 1.3.3 Documentation***. URL: https://deap.readthedocs.io/en/master/examples/ga_onemax.html (visited on 06/11/2023).

[54] ***Schlatterbeck/Pgapack: Parallel Genetic Algorithm Library Originally by David Levine from Argonne National Laboratory***. URL: https://github.com/schlatterbeck/pgapack/tree/master (visited on 08/09/2023).

[55] ***Schlatterbeck/Pgapy: Python Wrapper for Pgapack, the Parallel Genetic Algorithm Library***. URL: https://github.com/schlatterbeck/pgapy (visited on 08/09/2023).

[56] Ignacio Slater Muñoz. ***Keen | Kotlin Genetic Algorithm Framework***. URL: https://github.com/r8vnhill/keen.

[57] ***Stream (Java Platform SE 8)***. URL: https://docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html (visited on 07/26/2023).

[58] ***The Top Programming Languages***. The State of the Octoverse. URL: https://octoverse.github.com/2022/top-programming-languages (visited on 07/21/2023).

[59] ***Type-Safe Builders | Kotlin***. Kotlin Help. URL: https://kotlinlang.org/docs/type-safe-builders.html (visited on 07/31/2023).

[60] Franz Wilhelmstötter. ***Jenetics: Java Genetic Algorithm Library***. jenetics.io. URL: https://jenetics.io/ (visited on 06/08/2023).