Theoretical and Empirical Evaluation of Diversity-preserving Mechanisms in Evolutionary Algorithms

On the Rigorous Runtime Analysis of Diversity-preserving Mechanisms in Evolutionary Algorithms

Edgar Covantes Osuna

Department of Computer Science
University of Sheffield

This dissertation is submitted for the degree of
Doctor of Philosophy

February 2019

I would like to dedicate this thesis to my loving parents and to all my friends.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. Some pieces of this dissertation are based on articles that have been published elsewhere as specified in Section 1.1.

Edgar Covantes Osuna
February 2019

Acknowledgements

I would like to express my deep gratitude and appreciation to my supervisor Dr. Dirk Sudholt, I could not think of a better mentor. I would like to thank you for all your support, advice and for allowing me to grow as a research scientist. I would also like to specially thank Pietro S. Oliveto for his role as my second supervisor, and my co-authors Gao Wanru and Frank Neumann. I thank Carsten Witt for his advice and comments on one of the articles in which one of the chapters of this dissertation is based.

I would also like to thank my PhD buddies from the Algorithms group: Jorge, Alasdair, Donya, George, Juan Carlos, Dogan and Andrei. I thank as well all the members of the department that contributed to a enjoyable working atmosphere, specially to Abdullah. To my friends that I met here and that some have already left to their country of origin or are working overseas: Mario, Toñao, Alejandro, Ariel, Gerardo, Pablo, Rafael, Andres, Marcos, Alejandra, Lucy, Juana, Ale, Karla, Elizabeth. And to my friends in México who have made themselves felt despite the distance: Perla, Juan José, Pedro, Erika, Roberto, Hugo, Jesus, Cristopherson, Marisol, Dani, Rosendo, Bily, Miguel, Anselmo.

And of course my family, my father, César Covantes Rodríguez, mother, Elvia Yolanda Osuna Lizárraga, both of my brothers, Gerardo Covantes Osuna, and César Covantes Osuna and to his wife, Irma G. Tirado Lerma that together they brought to the world the most recent member of the family, César Alejandro Covantes Tirado, and the rest of my family deserve my deep gratitude for their endless support and patience during my PhD at Sheffield, specially to both of my grandmothers, Maria and Tomasa.

I gratefully acknowledge the financial support from the Department of Computer Science of The University of Sheffield, and the Consejo Nacional de Ciencia y Tecnología — CONACYT (the Mexican National Council for Science and Technology) under the grant no. 409151 and registration no. 264342 and the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 618091 (SAGE).

Last but not least, to my favourite pub in Sheffield, Red Deer, and to beer, thank you both for keeping me sane during all this journey.

Abstract

Evolutionary algorithms (EAs) simulate the natural evolution of species by iteratively applying evolutionary operators such as mutation, recombination, and selection to a set of solutions for a given problem. One of the major advantages of these algorithms is that they can be easily implemented when the optimisation problem is not well understood, and the design of problem-specific algorithms cannot be performed due to lack of time, knowledge, or expertise to design problem-specific algorithms. Also, EAs can be used as a first step to get insights when the problem is just a black box to the developer/programmer. In these cases, by evaluating candidate solutions it is possible to gain knowledge on the problem at hand.

EAs are well suited to dealing with multimodal problems due to their use of a population. A diverse population can explore several hills in the fitness landscape simultaneously and offer several good solutions to the user, a feature desirable for decision making, multi-objective optimisation and dynamic optimisation. However, a major difficulty when applying EAs is that the population may converge to a sub-optimal individual before the fitness landscape is explored properly.

Many diversity-preserving mechanisms have been developed to reduce the risk of such premature convergence and given such a variety of mechanisms to choose from, it is often not clear which mechanism is the best choice for a particular problem. We study the (expected/average) time for such algorithms to find satisfactory solutions for multimodal and multi-objective problems and to extract guidelines for the informed design of efficient and effective EAs. The resulting runtime bounds are used to predict and to judge the performance of algorithms for arbitrary problem sizes, further used to clarify important design issues from a theoretical perspective.

We combine theoretical research with empirical applications to test the theoretical recommendations for their practicality, and to engage in rapid knowledge transfer from theory to practice. With this approach, we provide a better understanding of the working principles of EAs with diversity-preserving mechanisms. We provide theoretical foundations and we explain when and why certain diversity mechanisms are effective, and when they are not. It thus contributes to the informed design of better EAs.

Table of contents

List of figures

List of tables

List of Algorithms

Nomenclature

Functions

$f = \mathrm{O}(g)$	Function f grows at most as fast as g
$f = \Omega(g)$	Function f grows at least as fast as g
$\mathrm{cp}(x)$	Closest peak, return the index of the closest peak to the search point x in the context of the Jansen-Zarges Multimodal Function Classes
$\mathrm{d}(x,y)$	Distance between x and y
$\mathrm{D}(t)$	Depth of a family tree of the (μ+1) EA at time t
$\mathrm{E}[X]$	Expectation of the random variable X
$f(x)$	Fitness function
$f_i(x)$	The i-th fitness function
$f_{\mathrm{sh}}(x,P)$	Shared fitness of individual x in the population P
$\mathrm{G}\big(x,x_{\mathrm{opt}}\big)$	GENERALISEDMAX function
$\mathrm{H}(x,y)$	Hamming distance between x and y for $x,\ y \in \{0,1\}^n$
$\mathrm{JZ}_1(x)$	Nearest peak function
$\mathrm{JZ}_2(x)$	Weighted nearest peak function
$\log n$	Logarithm of x to the base 2
$\ln n$	Natural logarithm of x, i. e., logarithm of x to base e
$f = \mathrm{poly}\,(n)$	Function f is polynomial
$\mathrm{Prob}(A)$	Probability of the event A

$\mathrm{sh}(x,y)$	Sharing function between x and y
$f = \mathrm{o}(g)$	Function f grows slower than g
$f = \omega(g)$	Function f grows faster than g
$f = \Theta(g)$	Functions f and g have the same order of growth
$\mathrm{T}(r^*)$	Family tree with root r^*
$\mathrm{u}(x)$	Unitation function

Greek Symbols

α	Scaling factor, a positive constant that regulated the shape of the sharing function in the context of fitness sharing
γ	The Euler-Mascheroni constant ≈ 0.577
κ	Niche capacity, maximum number of winners that a niche can accept in the context of clearing
λ	Offspring population size
μ	Population size
σ	Niching radius (sharing radius in the context of fitness sharing or clearing radius in the context of clearing), size of the niche
τ	Migration interval in the context of island models or maximum age in the context of ageing mechanism
ε	Some constant $\in \mathbb{R}^+$

Other Symbols

a_i	Slope of the i-th peak
b_i	Offset of the i-th peak
b^i	String with i concatenations of the letter b
e	Euler's number $e = \exp(1) = 2.7182\ldots$
F	Objective space

F^*	Pareto front
x_{opt}	Global optimum
H_n	Harmonic number $\mathrm{H}_n = \ln n + \Theta(1)$
x_i^t	The i-th individual or search point at the time t
k	Number of peaks
$\mathbb{Y}$	Set of local optima
m	Number of fitness functions
n	Problem size or length of a bitstring
$\mathbb{N}$	Set of natural numbers, $\mathbb{N} = \{1, 2, 3, \ldots\}$
$\mathbb{N}_0$	Set of natural numbers including zero
$\lvert x \rvert_1$	Number of 1-bits in individual x
p_c	Crossover probability
p_i	Position of the i-th peak
p_m	Mutation probability
P_t	Population at the time t
Q_t	Offspring population at the time t
$\mathbb{R}$	Set of real numbers
$\mathbb{R}^+$	Set of positive real numbers, $\mathbb{R}_{>0} = \{x \in \mathbb{R} \mid x > 0\}$
$\mathbb{R}_0^+$	Set of positive real numbers including zero, $\mathbb{R}_{\geq 0} = \{x \in \mathbb{R} \mid x \geq 0\}$
S	Search space
w	Window size, number of individual that participate in the tournament in the context of restricted tournament selection
X	Decision space in the context of Pareto optimality
X^*	Pareto set

$\|x\|_0$	Number of 0-bits in individual x

Acronyms / Abbreviations

aGA	Ageing Genetic Algorithm
AIS	Artificial Immune Systems
ASGA	Genetic Algorithms with Age Structure
CDC	Crowding Distance Contribution
cf.	*confer* or "compare"
CL	Clearing
DC	Deterministic Crowding
DE	Differential Evolution
DOPs	Dynamic Optimisation Problems
EA	Evolutionary Algorithm
EC	Evolutionary Computation
EDO	Evolutionary Dynamic Optimisation
e. g.,	*exempli gratia* or "for example"
EMO	Evolutionary Multi-objective Optimisation
EP	Evolutionary Programming
ES	Evolutionary Strategies
FEMO	Fair population-based Evolutionary Multi-objective Optimiser
FrEAK	Free Evolutionary Algorithm Kit
FS	Individual-based Fitness Sharing
GA	Genetic Algorithm
GDEMO	Global Diversity Evolutionary Multi-Objective Optimiser
GP	Genetic Programming

GSEMO	Global Simple Evolutionary Multi-objective Optimiser
HDC	Highest Diversity Contribution
HVC	Hypervolume Contribution
IBEA	Indicator-Based Evolutionary Algorithm
i. e.,	*id est* or "that is"
MOEA	Multi-Objective Evolutionary Algorithm
MOO	Multi-objective Optimisation
NFD	No Fitness Duplicates
NGD	No Genotype Duplicates
NMUAR	Non-Minimum Uniform at Random
NSGA-II	Non-dominated Sorting Genetic Algorithm II
PC	Probabilistic Crowding
PFS	Population-based Fitness Sharing
PL	Plain (μ+1) EA
RADEMO	Rank- And Distance-based Evolutionary Multi-Objective Optimiser
RLS	Randomized Local Search
RTS	Restricted Tournament Selection
SEMO	Simple Evolutionary Multi-objective Optimiser
SMS-EMOA	$\mathscr{S}$ Metric Selection Multi-Objective Optimisation Algorithm
SO	Single-objective Optimisation
SPEA2	Strength Pareto Evolutionary Algorithm 2
SSSP	Single Source Shortest Path Problem
STD	Standard Deviation
u. a. r.	Uniformly at random

w. h. p.	With high probability
w. l. o. g.	Without loss of generality
w. o. p.	With overwhelming probability

Part I

Introduction and Background

Chapter 1

Introduction

Darwinian evolution is intrinsically a robust search and optimisation mechanism. Living organisms demonstrate optimised complex behaviour at every level: cells, organs, individuals, and populations (Fogel, 1997). Biology is making continuous progress in the description of the components that make up living organisms and of the ways in which those components work together either by chaos, chance, temporality and nonlinear inter activities (Floreano and Mattiussi, 2008). The evolutionary process can be applied to optimisation problems. An optimisation problem refers to finding one or more feasible solutions which correspond to extreme values of one or more objectives. When an optimisation problem involves only one objective function, the task of finding the optimal solution is called *single-objective optimisation (SO)*. When an optimisation problem involves more than one objective function, the task of finding one or more optimum solutions is known as *multi-objective optimisation (MOO)*.

Compared with SO problems, which have a unique solution, the solution to MOO problems consists of sets of trade-offs between objectives. The goal of MOO is to generate these trade-offs. Exploring all these trade-offs is particularly important because it provides the system designer/operator with the ability to understand and weight the different choices available to them. The same evolutionary process can also be applied when analytical models are very complex, and numerical methods are susceptible of being nonconvergent, depending on the model used, when the optimisation problem is not well understood, and the design of problem of problem-specific algorithms cannot be performed due to lack of time, knowledge, or expertise to design problem-specific algorithms. As a result, evolutionary algorithms have received increased interest, particularly with regard to the manner in which they are applied for practical problem solving (Lee and El-Sharkawi, 2008).

Evolutionary Computation (EC) is the branch of computer science focusing on algorithms loosely inspired by the theory of evolution. Natural evolution is a hypothetical population-

based optimisation process. Simulating this process on a computer results in stochastic optimisation techniques that can often outperform classic methods of optimisation when applied to difficult real-world problems. EC is a branch of computational intelligence, it is also included into the broad framework of bio-inspired heuristics (Lee and El-Sharkawi, 2008; Squillero and Tonda, 2016).

In 1948, Turing (1948) was probably the first to propose a "genetical or evolutionary search", and by 1962, Bremermann (1962) had actually executed computer experiments on "optimisation through evolution and recombination". More commonly, the birth of EC is set in the 1960s with the appearance of four independent research lines: *Genetic Algorithms (GA)* (Goldberg, 1989; Holland, 1973), *Evolution Strategies (ES)* (Beyer and Schwefel, 2002; Schwefel and Rudolph, 1995), *Evolutionary Programming (EP)* (Fogel, 1995, 1999) and *Genetic Programming (GP)* (Koza, 1994). These four main paradigms, together with several variants proposed over the years, have been grouped under the term *Evolutionary Algorithms (EAs)* (Bäck et al., 1997; Eiben and Smith, 2003; Yu and Gen, 2012). For a more complete overview we refer the reader to a general book on EC by Eiben and Smith (2003) and for a more detailed overview of in the history of EC on the context of optimisation we refer to Chapter 1 in Yang (2010).

In ES, the algorithms share many features with the GA, the major similarity between these two types of algorithms is that they both maintain populations of potential solutions, and use selection mechanisms for choosing the solutions from the populations. The main difference is that ES relied on mutation as search operator, GAs rely mainly on operators called recombination or crossover, and ES is an abstraction of evolution at individual behaviour level, stressing the behavioural link between the parent population and its offspring, whereas GAs maintain the genetic link. Both algorithms evaluates the objective function (fitness) of each solution to guide its search. There is no requirement for other auxiliary knowledge or particular problem-knowledge.

Similar to ES, EP is a useful method of optimisation when other techniques such as gradient descent or direct analytical discovery are not possible. Combinatorial and real-valued function optimisation in which the optimisation surface or fitness landscape is "rugged", possessing many locally optimal solutions, are well suited for EP. In GP, each individual is a computer program. This approach is used as induction to devise a computer program. This is achieved by using evolutionary operators on candidate programs with a tree structure to improve the adaptive fit between the population of candidate programs and an objective function. An assessment of a candidate solution involves its execution.

Aside from the previous four approaches, there is a relatively new population-based optimisation technique called *Differential Evolution (DE)* (Storn and Price, 1997). Unlike

the conventional evolutionary algorithms that depend on predefined probability distribution function for mutation process, DE uses the differences of randomly sampled pairs of objective vectors for its crossover and mutation process. The object vectors' differences will pass the objective functions' topographical information toward the optimisation process with the intention of providing more efficient global optimisation capability. Another main characteristic of DE is its ability to search with floating point representation instead of binary representation as used in many basic EAs such as in GA and ES.

All EAs mentioned before can be applied to virtually any problem that can be formulated as a function optimisation task. The main steps for any of these algorithms is the candidate solution representation, variation operators to generate new solutions from the old ones, a performance metric in order to evaluate the solutions, and finally, some way of selection that allows the best individuals pass to next generations.

The representation of the candidate solution is commonly named as *genotype*, or in the context of GAs, the genetic material of a candidate solution or individual. The multi-set formed by all these candidate solutions is commonly named as population. Normally this population is created by generating potential solutions uniformly at random, using previous knowledge or as a product of another process like any other algorithm.

The solutions in the population are directly manipulated by the variation operators and can be chosen independent from the problem, the solution representation has to allow the variation operators the maintenance of the link between parent and offspring. The variation operators should allow small changes in the structure of a parent and lead to small changes in the resulting offspring in order to facilitate an understanding of the problem space, and likewise large changes should engender gross alterations.

The performance metric is commonly named as *fitness*, it is a measure of how well the candidate solution is able to solve the target problem. While alternative definitions have been proposed in literature, we will use the term *phenotype* as the intermediate form in which the genotype needs to be transformed into before it is evaluated (Squillero and Tonda, 2016). Finally, some kind of selection is used to ensure that the best solutions pass to the following iteration (also called generation). Over iterations of random variation and selection, the population can be made to converge asymptotically to optimal solutions (Fogel, 1997).

By iteratively applying selection for reproduction, variation operators (mutation and recombination), and selection for survival, a multi-set (population) of solutions (individuals) is evolved until a satisfactory solution is found. The strength of these algorithms relies on the stochasticity of the operators, which when well designed, will lead to an artificial evolution towards the optimal solution (Pérez Heredia, 2017).

The idea is to solve an optimisation problem by evolving sets of search points such that satisfying results are obtained. An EA maintains a population of individuals, $P_t = \{x_1^t, \ldots, x_n^t\}$ for iteration $t := 0$ (*initialisation*). Each individual represents a potential solution to the problem implemented on the search space S. This search space normally is structured as a Cartesian product of some other sets ($S = S_1 \times S_2 \times \cdots \times S_n$). The most common cases for theory and practical applications are $S = \{0,1\}^n$, $S = \mathbb{R}^n$ and the permutation set ($S = S_n = \{\pi \mid \pi \text{ is permutation on } \{1,2,\ldots,n\}\}$) normally represented as bitstrings of length n, vectors of n real numbers, and n natural numbers, respectively. In this thesis we will only focus on the search space or *genotype space* $\{0,1\}^n$.

The structure of an EA is shown in Algorithm 1. A population is created with μ individuals. Each solution x_i^t is evaluated (referred to as parents) to give some measure of its "fitness" given by the objective value of the function to be optimised; one speaks of the *fitness* of a solution and refers to the function as *fitness function*. Then, a new population Q_t with size λ is formed by selecting some parents (*selection for reproduction step*) and by applying transforming methods like mutation, which takes one parent as an input and randomly creates one individual, and/or higher order transformations like crossover, which creates (at least) one new individual by combining parts from several (two or more) individuals (*variation step*). The resulting population Q_t is called *offspring population*.

After creating the offspring population, most often there is some kind of replacement policy in which μ individuals are selected from the parent and/or the offspring population to be part of the new population P_{t+1}, and some are discarded (*selection for survival/replacement step*). After the new generation P_{t+1} is created, it is checked if some *termination criterion* is met. If not, the algorithm starts its cycle starting from the selection for reproduction step. It is hoped that the best individual represents an optimum (or reasonable) solution at the end of the evolutionary cycle.

Algorithm 1 Structure of an Evolutionary Algorithm

1: Let $t := 0$ and initialise the population P_0 with μ individuals.
2: Evaluate P_t.
3: **while** stopping criterion **not** met **do**
4: Select a multiset of parents from population P_t.
5: Create an offspring population Q_t of size λ by recombining the selected parents.
6: Apply mutation to the offspring population Q_t.
7: Select μ individuals from populations $P_{t+1} = P_t \cup Q_t$.
8: Evaluate P_{t+1} and let $t := t + 1$.
9: **end while**

Because of the modularity and flexible architecture of the EAs it is not unusual to use and/or combine different methods for each step. In the following we describe some of the

different methods that can be used in each step, and the more important ones for the present investigation, of course these procedures are not the only ones since there is no limit in the design of new methods and how to used them in the EA (cf. Jansen (2013) for more information about different modules and methods for the EAs).

Initialisation

Normally, for the search space $S = \{0,1\}^n$ this process is done purely at random (*uniform initialisation*). In some cases, initialisation may use previous knowledge. If there is enough information or any idea about how a "good" solution may be, it can be initialised accordingly (e. g., Friedrich et al., 2017) or it can be product of another process like any other heuristic.

Selection

As mentioned before, an EA has two selection procedures: selection for reproduction in which a set of individuals is selected as parents to undergo reproduction, and selection for survival, a set of individuals is selected to be part of the next generation. For simplicity we have merged both selection steps into a single section. Since selection for replacement can be described as selection for survival, it follows the same idea as selection for reproduction. Selection is often based on the fitness of the individuals but some approaches may use other properties like the age of the individual or taking even the population into consideration. Normally there are two ways to apply selection, by defining the probability to be selected or selecting all individuals that are needed in one batch.

The simplest method is called *uniform selection* where individuals are selected uniformly at random (u. a. r.). However, the original purpose of selection is to mimic the principle of *survival of the fittest*, hence it is reasonable that individuals with higher fitness have a higher probability of being selected, both for mating and for surviving to the next generation. Following this principle, we can find popular selection mechanisms such as *fitness-proportional selection* where an individual x from population P is selected with probability $f(x)/\sum_{y \in P} f(y)$ (this method assumes that all fitness values are positive). If differences between values in the population are very large, fitness-proportional selection behaves almost deterministically. On the other hand, small differences between values can lead to a uniform selection scheme.

Another method based on favouring higher fitness values is the *rank selection*, it uses fitness-proportional selection but it replaces the fitness value of an individual $x \in P$ by its rank, i. e., its position in the list of all individuals of the current population sorted in descending order with respect to fitness, ties are broken uniformly at random. *Tournament selection* is another very common selection mechanism, it selects individuals as winners of tournaments.

The individuals for the tournaments are chosen uniformly at random, and the winner is put into the population. This procedure is repeated until the specified number of individuals is achieved.

There are selection mechanisms where the best individuals are chosen deterministically from the current population or parent population with size μ and/or from the offspring population with size λ, e. g., *truncation selection*: the best individuals are selected with respect to its fitness, and ties are broken uniformly at random. There are two variants of truncation selection used for the selection for replacement step. *Comma-selection* chooses the best μ individuals from the offspring population. In this case if μ individuals are selected from λ individuals, λ needs to be chosen greater than μ. This kind of selection is denoted as (μ, λ). And *plus-selection* where the best μ individuals are selected from both, the parent and the offspring population, and it is denoted as $(\mu+\lambda)$. Normally plus-selection has a slight preference for the offspring, i. e., if the parent and offspring have equal fitness, the offspring is preferred.

Mutation

Is a variation operator that only uses one individual as input $M : \{0,1\}^n \to \{0,1\}^n$ (unary operator). This operator depends on the structure of the individuals and thus on the search space. All mutation operators have in common that they tend to create rather small changes. Typically, mutation uses only the genotypic information of the parent to produce a slightly different genotype. Two of the most extended mutation operators for the $\{0,1\}^n$ search space are *standard bit mutation* or *global mutation* and *k-bit mutation*. Standard bit mutation creates a new offspring by flipping each bit from a parent x independently at random with probability p_m (mutation probability). The most common mutation probability is $p_m = 1/n$, flipping just one bit on average. In k-bit mutation, an offspring y is created by flipping exactly k bits from a parent x. Each position of these k bits is chosen uniformly at random, and is user defined. The most common value for k is 1 and is also known as *local mutation*.

Crossover

Similar to mutation, this operator cannot be designed independently of the search space. The difference from mutation is that more than one parent is used. Most operators make use of two parents (binary operator $C : \{0,1\}^n, \{0,1\}^n \to \{0,1\}^n$), which is clearly closer to the natural paradigm. For any two parents $x_1, x_2 \in \{0,1\}^n$, is not unusual for the crossover operator to create two offspring, one offspring created by assembling pieces of the two parents and the other with the unused pieces of the two parents.

In *k-point crossover*, the method selects k different positions from $\{1,2,\ldots,n-1\}$ uniformly at random. Let these positions be $p_1 < p_2 < \cdots < p_k$. Then, the offspring y is defined by copying the first p_1 bits from x_1, the second $p_2 - p_1$ from x_2, the next $p_3 - p_2$ bits from x_1, and so on, alternating between x_1 and x_2. This method can be visualised as having the two parent bitstrings cut into pieces after each p_i-th position. Then the offspring is the concatenation of the pieces taken alternately from the two parents. The most common forms of k-point crossover are 2-point crossover and even 1-point crossover. In *uniform crossover*, for each bit the value is copied from one of the parents and the decision the bits to be copied is made independently and with equal probability for each bit.

The idea of crossover is to generate an offspring by inheriting beneficial properties from both parents. However, the same rationale works for detrimental properties. Although the research question is not settled, crossover has been proved to be useful for some scenarios (as we will see in Section 3.3).

Termination Criterion

As the final step of the evolutionary cycle, it decides if the algorithm is stopped or if another generation is to be started. This criteria can be more or less flexible depending of the method used. With *adaptative termination criteria*, it is possible to considerate any kind of properties like population, fitness values, or statistics; common examples are stagnation of the population or number of generations since the last improvement. *Fixed termination criteria* uses a predefined number of generations or fitness evaluations to stop the algorithm. And finally, from the theoretical perspective, a *no termination criterion* is used. This kind of termination criterion is mainly used in theoretical analysis, the EA runs until the first time an optimum has been found, this time takes into consideration the number of function evaluations required to find such global optimum. As mentioned before, this time is called optimisation time, and it is used to describe how long it is necessary to wait on average in order to find the optimal solution.

As described before, some modules of the EA may have several parameters that need to be set but an EA also has global parameters aside from the methods mentioned before. The value of these global parameters has a direct effect on the complexity of the algorithm. Most of the empirical and theoretical analyses are focused on providing guidelines of how to set these global parameters and to explain the complete behaviour of the EA. In this section we introduce the most basic parameters.

The *dimension of the search space n* does not belong to the EA process, it is a property of the search space and thus the fitness function. This parameter is one of the most important since the performance of the EA depends on the size of the input, and its results are expressed

in an asymptotic form (see Definition 2.2). One assumes that the size of the input grows to infinity. Finally, some parameters depends on n like the mutation rate, which often is defined as $1/n$.

Other parameters related to the populations sizes like *population size* μ and the *offspring population size* λ, the number of individuals in the population and the number of offspring generated in each generation have to be defined during the design phase. Concerning mutation, the probability to produce an offspring via mutation is defined as *probability for mutation* p_m. We set this probability to 1, this means that the algorithm always is going to apply mutation to an individual. Since all of the EAs analysed in this thesis only use mutation as a variation operator, and mutation is applied with probability 1, we will refer to the parameter p_m as *mutation rate*, i. e., the probability of each bit in the bitstring is flipped. And finally, the *crossover probability* p_c, the probability to produce an offspring via crossover. Any choice $p_c \in [0,1]$ is possible; the most often recommended choices are quite large constant values like $p_c \in [0.5, 0.8]$. In some EAs either crossover or mutation is applied. In some cases crossover is applied with probability p_c and consequently mutation with probability $1 - p_c$ and in other instances, the application of crossover and mutation is independent and an offspring created by crossover undergo a subsequent mutation.

According to Floreano and Mattiussi (2008), the theory of natural evolution rests on four pillars: *population*, *diversity*, *heredity*, and *selection*. It is not possible to speak of evolution of a single organism. A necessary component of evolution is the existence of a population, which consists of two or more individuals. Diversity means that the individuals of the population vary from one another to some extent.

According to Squillero and Tonda (2016), this diversity can be quantified in three different ways: as a distance metric between individuals; as a measurable attribute of individuals (individual diversity); as a characteristic of the population as a whole (population diversity). Using some distance metric it is possible to measure the diversity contribution of an individual (how far the individual is from the other individuals in the population) or the population (the average of the individual diversity) (Squillero and Tonda, 2016). In many cases the use of a distance metric is not necessary to define individual diversity or population diversity, in some cases a problem related metric can be used like: the number of optimal solutions have been obtained.

Heredity indicates that individual characters can be transmitted to offspring through reproduction (Floreano and Mattiussi, 2008), ideally it is desired that the parents inherit their best features to the next generation of offspring. To get better offspring, it is necessary to select "good" parents. Selection indicates that only a subset of individuals will be able to reproduce, and in the same time, that only a subset of individuals will be able to survive

and to transmit its characteristics to the future generations. This selection for reproduction and survival are not completely random, it is regulated by environmental conditions like too many individuals competing for the available resources, better or faster individuals will have higher chances of reproduction and survival (Floreano and Mattiussi, 2008).

Considering diversity, many solutions have been proposed to maintain or promote diversity in an EA (Črepinšek et al., 2013; Glibovets and Gulayeva, 2013; Shir, 2012; Squillero and Tonda, 2016). There are many methods that have been proposed to maintain and/or promote diversity in an evolutionary algorithm. These mechanisms range from simple ones like: eliminating duplicates, replacing a fraction of the current population with new generated solutions (or even replacing the whole population) or varying the selection pressure from an elitist approach to more relaxed selection approaches to let less fit individuals participate in the mating process. Other more advanced approaches divide the population into subpopulations like island models, cellular evolutionary algorithms or niching techniques. Given the plethora of mechanisms to be applied, it is often not clear what the best strategy is. Which diversity mechanisms work well for a given problem, which don't, and, most importantly, why? In particular, the effect of such mechanisms have on search dynamics and performance are often not well understood.

The main goal of this thesis is to narrow the gap between theory and practice by analysing diversification components or diversity-preserving mechanisms for population-based EAs as tools to increase diversification. We want to observe if these EAs can be potentiated or enhanced with diversity techniques. We want to use the tools and techniques for the analysis of evolutionary algorithms with any diversity-preserving mechanism. For the case of SO, we want to analyse the expected time till a diverse set of optima have been found, and for the case of MOO, we want to analyse the expected time till the set of possible trade-offs between objectives has been generated or found.

The main contribution of this thesis is the plethora of rigorous theoretical and empirical results of EA in which we show how and why diversity play a key role. There has been a line of work comparing various diversity mechanisms on several test functions (we will make a rigorous review of this line of work in Chapter 3) in the context of various evolutionary algorithms. The results obtained in this thesis help to get insight into the search behaviour of evolutionary algorithms in the presence or absence of diversity, and how parameters and explicit diversity mechanisms affect the performance. These results in particular highlight which diversity mechanisms are effective for particular problems, and which are ineffective. More importantly, they explain how to design the most effective evolutionary algorithms for the problems considered.

Aside from this introduction, this thesis contains 2 introductory chapters, Chapter 2 introduces the reader to the field of runtime analysis of evolutionary algorithms. In Section 2.1 we describe some prominent fitness functions, we review some other functions in later sections in the context of different algorithms and desired goals. In that section we describe the most common functions analysed in the area of runtime analysis, we describe several single-objective functions with one optimum and functions with more than one optimum (multimodal functions). We also describe multi-objective functions and some of their properties.

The most important evolutionary algorithms for this thesis are defined in Section 2.2. There we have divided the randomised search heuristics in single-objective algorithms and multi-objective algorithms. Finally, in Section 2.3 we describe some of the methods and tools from the analysis of randomised algorithms that are adapted to the analysis of evolutionary algorithms. As these methods will be used throughout this thesis, we normally present an application example to familiarise the reader with their application. We also make use of these examples to give a first impression on the asymptotic runtime of evolutionary algorithms on the example functions defined in Section 2.1.

The second introductory chapter (Chapter 3) provides an extensive survey of the work done in runtime analysis of diversity on evolutionary algorithms. This literature review is based on the work done by Sudholt (2018), we extend his survey by adding an extensive review on the ageing mechanism (commonly applied in the area of artificial immune systems) on evolutionary algorithms (Section 3.2.6). This review of rigorous theoretical runtime analyses of evolutionary algorithms cover several algorithmic approaches where diversity plays a key role, such as results related to implicit operators of evolutionary algorithms like mutation and crossover or explicit operators embedded to evolutionary algorithms like avoiding genotypes or phenotype duplicates, dividing the population into subpopulations like island models, and niching techniques that try to establish niches of similar search points, and preventing niches from going extinct. Some of the niching techniques reviewed and analysed in this thesis include deterministic crowding and fitness sharing. Another contribution done in this thesis is a review of diversity in multi-objective optimisation (Section 3.5). We found that diversity mechanisms that are effective for one problem may be ineffective for other problems, and vice versa.

After both introductory chapters we continue with the main contribution of this thesis and original work from this research. Chapter 4 extends the results presented in Section 3.2 by presenting the runtime analysis of three niching mechanisms on a bimodal function with two hills with symmetric slopes (branches) with 0^n and 1^n as global optima called TwoMax. In Section 4.1 we analyse the niching mechanism called probabilistic crowding, here the

offspring competes against its parent using fitness-proportional selection. We rigorously prove that this method fails miserably and we include additional experimental results used to corroborate our theoretical proofs. Probabilistic crowding is not even able to evolve search points that are significantly better than those found by random search. Section 4.2 contains the results for restricted tournament selection (RTS), here the offspring competes with the closest element from w (window size) members selected uniformly at random from the population, and the best individual from this competition is allowed to pass to the next generation. The analysis of RTS is divided in 3 sections: Section 4.2.1 provides a positive result on the example bi-modal function TwoMax, if w is chosen very large, the evolutionary algorithm is able to find both optima in the same fashion as deterministic crowding. On the contrary, in Section 4.2.2, if w is chosen to small, it may be possible that the offspring competes with an individual that is too different, resulting in a competition between individuals in different subpopulations, leading to an unwanted replacement which translates into a loss of diversity. The third section consists of complementary experiments in which we explore different settings for the population size and the parameter w (Section 4.2.3).

Finally in Section 4.3, we present an extensive analysis on the clearing diversity mechanism. The basic idea of this mechanisms is that the search space is divided into niches (or subpopulations) by a parameter σ, and in each niche there is an individual called the winner that has its fitness value preserved while the others individuals contained in that niche have their fitness value cleared or removed. This mechanism allows for both exploitation and exploration: it allows winners to find fitness improvements, while at the same time enabling cleared individuals to tunnel through fitness valleys. In fact, cleared individuals are agnostic to the fitness landscape as they always have the worst possible fitness. Hence cleared individuals can explore the landscape by performing random walks. As we will show, this allows the algorithm to escape from local optima with even very large basins of attraction. As in previous analyses, the results are divided into different sections that explore different aspects of the algorithm. Section 4.3.1 deals with a small setting for the σ parameter and how this allows to optimise all functions of unitation when the proper values for other parameters are chosen, and a proper distance metric is used. Section 4.3.2 deals with a large setting for σ, and how it can optimise the TwoMax function. We also provide additional experimental results used to highlight the strengths of this mechanism.

In Chapter 5 we provide an extensive empirical analysis on the performance of several diversity-preserving mechanisms on example landscapes for multimodal optimisation. This analysis include all diversity mechanisms mentioned in Section 3.2 and Chapter 4. We provide insights into the working principles of these mechanisms by testing their ability to find many peaks with equal height (Section 5.2.1) and their ability of each mechanism to

maintain the population diversity when considering peaks with different heights to yield global and local optima (Section 5.2.2). Finally, the third one focusses on landscapes with two peaks, we want to test the ability of the diversity mechanisms to deal with different basins of attraction and to observe which mechanisms are able to escape from local optima by tunnelling through the fitness valley that separates two peaks (Section 5.2.3). We make use of previous theoretical results to inform the choice of algorithm parameters and to discuss in how far our empirical results agree with theoretical results obtained for TWOMAX.

Chapter 6 deals with the analysis of multi-objective evolutionary algorithms (MOEAs). In this chapter we analyse the performance of well-known MOEAs in the context of diversity-based parent selection. We introduce parent selection mechanisms that make use of information provided by two diversity contribution metrics to select promising individuals for reproduction. The general framework is based on the idea that individuals in less populated areas (or poorly explored areas) of the objective space should have better chances to create new individuals than those in areas where there are several individuals.

We have designed different parent selection schemes with different degrees of strength, from mild preferences for more appealing parents to very aggressive schemes. All these parent selection schemes use hypervolume contribution or crowding distance contribution to select individuals, the higher the contribution the better. Section 6.1 introduces the algorithmic framework like the MOEAs and functions used for the theoretical analysis. Section 6.2 provides formal definitions of the parent selection schemes, definitions of the diversity metrics, and a description of how to modify the plain algorithms by introducing the diversity-based parent mechanisms into the MOEAs. Section 6.3 establishes the analytical framework used on the theoretical analysis; here we stablish some general properties that show how the MOEAs with diversity-based parent selection mechanism can outperform its plain variant.

The theoretical analysis for both of our example functions is presented in Section 6.4 and 6.5, respectively. In each section we show that it is possible to achieve speedups on the performance of the algorithms. The theoretical results show that a linear factor can be saved for the investigated settings. In Section 6.6 we provide experimental results related to the effectiveness of the parent selection mechanisms. The experiments show that it is possible to achieve a speedup of one magnitude for problems of size $n = 100$.

Finally, in Section 6.7 we conclude the chapter by complementing the theory done in Section 6.4 and 6.5 with more theoretical studies on the effectiveness of greediness in parent selection. There we show that very extreme schemes can lead to undesired results. This thesis ends with conclusions and an outlook in Chapter 7 and an appendix containing some useful mathematical tools collected from the literature needed through all the thesis. In this appendix

it is contained basic probability theory needed to understand this thesis, mathematical tools like Chernoff bounds and some useful inequalities that were used during the writing of this thesis and from the following underlying publications.

1.1 Underlying Publications

Aside from the literature review provided in Chapters 2 and 3, we provide a list of the main contributions of this thesis based on the following publications. Authors' names are sorted alphabetically. For all joint papers with k authors, this author's contribution in terms of ideas, proofs, and writing can roughly be quantified as $1/k$.

Chapter 4 is based on the following papers:

1. Covantes Osuna, E. and Sudholt, D. (2018b). On the Runtime Analysis of the Clearing Diversity-Preserving Mechanism. *Evolutionary Computation*. To appear. Preprint available from http://arxiv.org/abs/1803.09715.

   A preliminary version was published in:

   Covantes Osuna, E. and Sudholt, D. (2017). Analysis of the Clearing Diversity-Preserving Mechanism. In *Proceedings of the 14th ACM/SIGEVO Conference on Foundations of Genetic Algorithms*, FOGA '17, pages 55–63. ACM.

2. Covantes Osuna, E. and Sudholt, D. (2018c). Runtime Analysis of Probabilistic Crowding and Restricted Tournament Selection for Bimodal Optimisation. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '18, pages 929–936. ACM. **Nominated for a best paper award in the track 'Genetic Algorithms'**.

Chapter 5 is based on the following paper:

3. Covantes Osuna, E. and Sudholt, D. (2018a). Empirical Analysis of Diversity-Preserving Mechanisms on Example Landscapes for Multimodal Optimisation. In *Parallel Problem Solving from Nature – PPSN XV*, pages 207–219. Springer International Publishing.

Chapter 6 is based on the following papers:

4. Covantes Osuna, E., Gao, W., Neumann, F., and Sudholt, D. (2018). Design and analysis of diversity-based parent selection schemes for speeding up evolutionary multi-objective optimisation. *Theoretical Computer Science*. To appear. Preprint

available from http://arxiv.org/abs/1805.01221.

A preliminary version was published in:

Covantes Osuna, E., Gao, W., Neumann, F., and Sudholt, D. (2017). Speeding Up Evolutionary Multi-objective Optimisation Through Diversity-based Parent Selection. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '17, pages 553–560. ACM.

Chapter 2

Runtime Analysis of Evolutionary Algorithms

In this chapter we will introduce the most important EAs, fitness functions and tools for the analysis of EAs will be also defined in this chapter. We will like to mention that some parts of this chapter have been freely adapted from the main three textbooks regarding the theoretical runtime analysis of EAs: Neumann and Witt (2010), Auger and Doerr (2011) and Jansen (2013), for any other source we will make explicit reference to each publication.

In the case of runtime analysis, the specific definition of a stopping criterion is often omitted since the algorithm is considered as an infinite stochastic process. The main interest is the random time until a global optimum is found, this performance measured is denoted as *runtime* or *optimisation time* (in our case, optimisation time based on maximal fitness), the time T (number of fitness evaluations) needed for an algorithm to finish its execution.

Definition 2.1 (Optimisation Time)**.** *Let $\{X_t\}_{t\geq 0}$ be a stochastic process on the search space $S = \{0,1\}^n$ and $f : S \to \mathbb{R}$ a fitness function. The optimisation time T is the first point in time when the process' value yields the maximum fitness value in the population, i. e.,*

$$T := \inf\left\{ t \geq 0 \mid X_t = \arg\max_{X_t \in S} f(X_t) \right\}.$$

As mentioned before, EAs are optimisation and randomised search heuristics, hence the question translates to finding the expected time ($\mathrm{E}[T]$) needed to find the optimal solution. This time is often expressed with asymptotic notation (also known as Landau notation or "big-Oh" notation) in terms of the problem size n. This reflects the order of growth of the bound and neglects constant factors or smaller order terms, leading to simplified terms for

the bounds (the books by Motwani and Raghavan (1995) and Cormen et al. (2009) give more details on the following definitions).

Definition 2.2 (Asymptotic Notation)**.** *Let $f,\ g : \mathbb{N}_0 \to \mathbb{R}^+$ be two functions, we say that:*

- *f grows at most as fast as g and write $f = \mathrm{O}(g)$ if and only if there exist constants $c \in \mathbb{R}^+$ and $n_0 \in \mathbb{N}$, such that for all $n \geq n_0$ it holds that $f(n) \leq cg(n)$.*
- *f grows at least as fast as g and write $f = \Omega(g)$ if and only if $g = \mathrm{O}(f)$.*
- *f and g have the same order of growth and write $f = \Theta(g)$ if and only if $f = \mathrm{O}(g)$ and $f = \Omega(g)$.*
- *f grows faster than g and write $f = \omega(g)$ and only if $\lim_{n\to\infty} g(n)/f(n) = 0$.*
- *f grows slower than g and write $f = \mathrm{o}(g)$ if and only if $g = \omega(f)$.*

By using asymptotic notation we can focus on the order of growth by abstracting away some details of this function. Furthermore, it is possible to establish efficiency criteria for EAs. Generally speaking, a polynomial optimisation time denotes an efficient algorithm while an algorithm with an exponential optimisation time is inefficient for the considered problem. Here some notions for inverse polynomial and inverse exponential functions that will be used mostly for probabilities.

Definition 2.3 (Polynomial/Superpolynomial/Exponential)**.** *For a function $f : \mathbb{N} \to \mathbb{R}^+$, we say that*

- *f is polynomial $f = \mathrm{poly}(n)$ if $f(n) = \mathrm{O}(n^c)$ for some constant $c \in \mathbb{R}_0^+$.*
- *f is superpolynomial if $f(n) = \omega(n^c)$ for every constant $c \in \mathbb{R}_0^+$.*
- *f is exponential if $f(g) = \Omega\left(2^{n^\varepsilon}\right)$ for some constant $\varepsilon \in \mathbb{R}^+$.*
- *f is polynomially small and write $f = 1/\mathrm{poly}(n)$ if $1/f$ is polynomial.*
- *f is superpolynomially small if $1/f$ is superpolynomial.*
- *f is exponentially small if $1/f$ is exponential.*

Since we use the Landau notation to express the expected optimisation time needed to find the optimal solution in terms of the problem size n and using the efficiency criteria defined in Definition 2.3, we use the same handy notation for events that are very likely defined as *notions for probabilities* by Sudholt (2008).

Definition 2.4 (Notions for probabilities, Sudholt, 2008)**.** *We say that an event A occurs with high probability (w. h. p.) if $1-\text{Prob}(A) = \text{O}\left(n^{-k}\right)$ for some $k \in \mathbb{R}^+$. We say that an event A occurs with overwhelming probability (w. o. p.) if $1-\text{Prob}(A)$ is exponential small.*

2.1 Fitness Functions

In the following we will make use of definitions and conventions that are standard in this field (see e. g., Pérez Heredia, 2017; Sudholt, 2008). Throughout this thesis, we only consider the maximisation of pseudo-Boolean functions $f : \{0,1\}^n \to \mathbb{R}$. Throughout this thesis n always denotes the number of bits, μ the population size, and λ the offspring population size. Although we only consider maximisation, it is straightforward to notice that minimisation problems can be obtained just by multiplying the fitness function by -1. Elements of the search space are synonymously called *search points*, *solutions*, or *individuals* in the context of EAs. For a search point x we often denote $x = x_1 \dots x_n$ where x_i represents the value of the i-th bit in the bitstring. Thereby, we use the string notation $x_1 \dots x_n$ as concatenation of $x_1, x_2, \dots, x_n$. In particular, we adopt the widely used notation b^i for the i concatenations of the letter b. Thus, the all-one bitstring of length n can be written as 1^n.

Each search point gets assigned a real value by the pseudo-Boolean fitness function and each search point can be located in the the Boolean hypercube by the number of 1-bits (vertical position) and the number of 1-bits until the first appearance of a 0-bit (horizontal position). Figure 2.1 sketches the 2-dimensional projection of the structure of the hypercube and the location of some points in the hypercube.

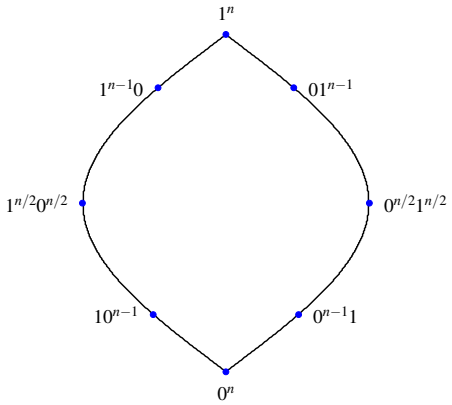


Figure 2.1: Projection of the Boolean hypercube in two dimensions with some points located based on the position and number of zeroes and ones.

One way to define how close or far, equal or different two search points are when studying pseudo-Boolean functions, is to define some distance (or dissimilarity) metric between search

points. The most common distance metric for the Boolean hypercube is the *Hamming distance* or also called *genotypic distance* in the context of dissimilarity metric between two individuals in the space of genotypes.

Definition 2.5 (Hamming Distance or Genotypic Distance)**.** *Given two bitstrings* $x,\ y \in \{0,1\}^n$*, the Hamming distance between them is the number of bits that have different values:*

$$\mathrm{d}(x,y) := \mathrm{H}(x,y) := \sum_{i=1}^{n} |x_i - y_i|,$$

if $\mathrm{H}(x,y) := 1$ *we say that x and y are Hamming neighbours.*

We define optima with respect to Hamming distance. The term *local optimum* will be used for bitstrings without Hamming neighbours of higher fitness that are not global optima. And *global optimum* will be used for the search point with the highest fitness.

Definition 2.6 (Local and Global Optima)**.** *Given a function* $f : \{0,1\}^n \to \mathbb{R}$*, the set* $\mathbb{Y}$ *of local optima is given by* $\mathbb{Y} := \{y \mid y \in \{0,1\}^n, \forall x \in \{0,1\}^n : \mathrm{H}(x,y) = 1 \Rightarrow f(y) \geq f(x)\}$*. And the global optimum* x_{opt} *is defined as* $x_{\mathrm{opt}} := \arg\max\{f(y) \mid y \in \{0,1\}^n\}$*.*

Most of the theoretical analyses are made on *example functions* with specific characteristics based on natural landscapes that reflects the search space. A single global optimum can be seen as a "hill-climbing" task where the goal is to reach the top of a "hill". In cases of multiple global optimum, the goal is to find regions of the search space where the fitness points towards a "peak" (this region is informally called "basin of attraction"), and to reach the top of the peak. Sometimes we are interested in escaping from the basin of attraction of a suboptimal peak in order to reach the basing of attraction of a better peak. This kind of behaviour resembles to the need to cross a "valley" with paths of Hamming neighbours with high fitness surrounded of points of lower fitness (ridges) or paths of Hamming neighbours with equal fitness values (plateau). We do not present formal definitions for these terms as they are only used in informal discussions on a high level of abstraction. We normally will use these terms to refer a particular task and to highlight some of the properties of the algorithms analysed. We will refer to this high level abstraction as *fitness landscape*.

2.1.1 Single-objective Functions

When an optimisation problem involves only one objective function, the task of finding the optimal solution is called *single-objective optimisation*. In single-objective optimisation, every solution is mapped to a real value and solutions can always be compared in pairs via the less or equal relation $\geq$ on $\mathbb{R}$.

2.1.1.1 Unimodal Functions

In cases where a function f only contains exactly one global optimum, f it is called an *unimodal function*.

Definition 2.7 (Unimodal Function)**.** *A function f is called unimodal if and only if for every search point x that is not a global optimum there is a Hamming neighbour y of x with $f(y) > f(x)$.*

The GENERALISEDMAX function as defined by Pérez Heredia (2017) simply counts the number of mismatched bits between a solution x and a given target solution x_{opt}. It represents an easy hill climbing task since each bit-position's fitness contains information about the direction towards the optimum.

Definition 2.8 (GENERALISEDMAX)**.** *Let $x,\ x_{\text{opt}} \in \{0,1\}^n$, then*

$$\mathrm{G}\big(x, x_{\text{opt}}\big) := n - \mathrm{H}\big(x, x_{\text{opt}}\big).$$

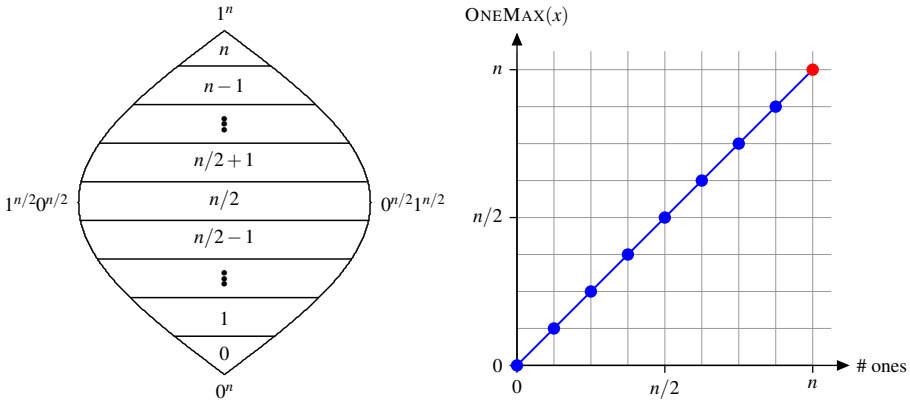
The choice of the target solution x_{opt} does not affect the optimisation process, since the algorithms studied here are unbiased (i. e., they do not favour flipping 0-bits into 1-bits or vice versa). Then, the typical choice for the target solution is the all ones bitstring, this way we introduce the most popular unimodal test function used to study EAs, the function counts the number of ones in the bitstring x: $\textsc{OneMax}(x) := \textsc{GeneralisedMax}(x, 1^n)$.

Definition 2.9 (ONEMAX)**.** *The function counts the number of ones in the bitstring $x \in \{0,1\}^n$, then*

$$\textsc{OneMax}(x) := \sum_{i=1}^{n} x_i.$$

The goal of the optimization process is to find the maximum number of ones in a bitstring. The global optimum is the 1^n bitstring (see Figure 2.2). By symmetry, $\textsc{ZeroMax}(x) := n - \textsc{OneMax}(x)$ holds for the 0^n bitstring.

An example of a function with a ridge landscape is the LEADINGONES function. In this function the fitness value is defined by the length of uninterrupted sequence of 1-bits starting at x_0. In this function the bits positions are key: bits following the first 0-bit have no effect on fitness and flipping one 1-bit in the sequence of consecutive bits can yield to a huge fitness loss. As in the previous case, the choice of the all ones bitstring as target solution does not affect the optimisation process.



(a) Genotype

(b) Phenotype

Figure 2.2: Sketches of the function OneMax with $n = 8$ on the genotypic space (2.2a) and on the phenotypic space (2.2b). The position of a point on the genotypic space is given by the position and number zeroes and ones in the bitstring, and the position of a point on the phenotypic space is just given by the number of ones in the bitstring.

Definition 2.10 (LeadingOnes)**.** *The function counts the number of 1-bits at the beginning of the bitstring until the appearance of the first 0-bit. Let $x \in \{0,1\}^n$, then*

$$\text{LeadingOnes}(x) := \sum_{i=1}^{n} \prod_{j=1}^{i} x_j.$$

The goal of the optimisation process is to find the bitstring 1^n.

2.1.1.2 Multimodal Functions

Functions with several local and global optima of equal or different fitness are commonly called *multimodal*. Real optimisation problems often lead to multimodal domains and so require the identification of multiple optima, either local or global (Sareni and Krahenbuhl, 1998; Singh and Deb, 2006). Multimodality in a search and optimisation problem usually causes difficulty to any optimisation algorithm in terms of finding the global optimum solutions. This is because in these problems there exist many attractors in which an algorithm can become directed to. Finding the global optimum solutions in such a problem becomes a challenge to any optimisation algorithm.

A well-known multimodal function in the area of runtime analysis is the bimodal function TwoMax (also investigated in the context of GAs by Hoyweghen et al., 2002; Pelikan and Goldberg, 2000). TwoMax can be seen as a bimodal equivalent of OneMax. The fitness landscape consists of two hills with symmetric slopes (or branches) ZeroMax and

ONEMAX with 0^n and 1^n as global optima, respectively. In contrast to Friedrich et al. (2009) where an additional fitness value for 1^n was added to distinguish between a local optimum 0^n and a unique global optimum, we have opted to use the same approach as Oliveto et al. (2014), and leave unchanged TWOMAX (see Definition 2.11).

Definition 2.11 (TWOMAX)**.** *A bimodal function which consists of two different symmetric slopes* ZEROMAX *and* ONEMAX *with* 0^n *and* 1^n *as global optima, respectively.*

$$\text{TWOMAX}(x) := \max\left\{\sum_{i=1}^{n} x_i, n - \sum_{i=1}^{n} x_i\right\}.$$

In the region of search points with more than $n/2$ *1-bits, the fitness increases with the number of 1-bits and in the region of search points with less than* $n/2$ *1-bits, the fitness increases with the number of 0-bits. These sets are refereed as branches. The aim is to find a population containing both optima (see Figure 2.3).*

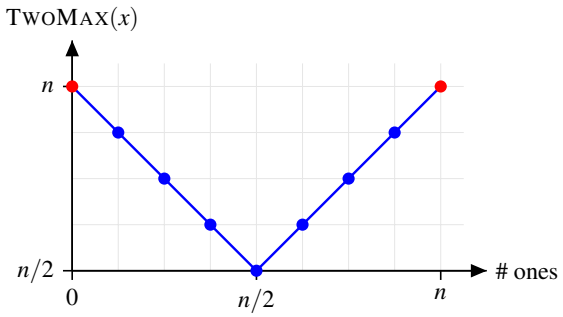


Figure 2.3: Sketch of the function TWOMAX with $n = 8$.

TWOMAX is an example of a function of unitation, which are defined as follows

Definition 2.12 (Function of Unitation)**.** *Any function* $\mathrm{u} : \{0,1\}^n \to \mathbb{R}$*, where* $\mathrm{u}(x)$ *depends only on the number of ones in x and is always non-negative. That is,* u *is entirely defined by a function* $f : \{0,\ldots,n\} \to \mathbb{R}_0^+$.

In the case of functions of unitation, instead of using Hamming distance between two search points, often a dissimilarity metric in the space of phenotypes is used (Covantes Osuna and Sudholt, 2017, 2018b,c; Friedrich et al., 2009; Oliveto et al., 2014). Usually defined as Euclidean distance between two phenotypes, a phenotypic distance function, allows the distance function $\mathrm{d}(\cdot,\cdot)$ to depend on the number of ones (see Definition 2.13).

Definition 2.13 (Phenotypic Distance)**.** *Given two bitstrings $x,\ y \in \{0,1\}^n$, the phenotypic distance is defined only on the number of ones:*

$$\mathrm{d}(x,y) := ||x|_1 - |y|_1|,$$

where $|x|_1$ and $|y|_1$ denote the number of 1-bits in individual x and y, respectively.

Jansen-Zarges Multimodal Function Classes

Introduced by Jansen and Zarges (2016), the authors addressed the need for more suitable benchmark functions for the theoretical analysis of EAs on multimodal functions. Their benchmark functions allow the control of different features like the number of peaks (defined by their position), their slope and their offset (also called height provided in an indirect way), while still enabling a theoretical analysis. This benchmark setting is defined in the search space $\{0,1\}^n$ and it uses the Hamming distance between two bitstrings.

Jansen and Zarges (2016) define their notion of a landscape as the set of peaks $k \in \mathbb{N}$ and the definition of the k peaks (numbered $1,2,\ldots,k$) where the i-th peak is defined by its position $p_i \in \{0,1\}^n$, its slope $a_i \in \mathbb{R}^+$, and its offset $b_i \in \mathbb{R}_0^+$ (both slope and offset are used to influence and modify the fitness value). The general idea is that the fitness value of a search point depends on peaks in its vicinity and by using the slope and offset of the peaks higher peaks can "overrule" closer but smaller peaks. The main objective for any optimisation algorithm operating in this landscape is to identify those peaks: a highest peak in unimodal optimisation or a collection of peaks in multimodal optimisation. A peak has been identified or reached if the Hamming distance of a search point x and a peak p_i is $\mathrm{H}(x,p_i) = 0$. Since we are considering maximisation, it is more convenient to consider $\mathrm{G}(x,p_i)$ instead (see Definition 2.8).

There are three different fitness functions used to deal with multiple peaks in Jansen and Zarges (2016); we consider the two most interesting function classes JZ_1 and JZ_2 defined in the following.

Definition 2.14 (Definition 3 in Jansen and Zarges, 2016)**.** *Let $k \in \mathbb{N}$ and k peaks (p_1,a_1,b_1), $(p_2,a_2,b_2),\ldots,(p_k,a_k,b_k)$ be given, then*

- $\mathrm{JZ}_1(x) := a_{\mathrm{cp}(x)} \cdot \mathrm{G}\big(x, p_{\mathrm{cp}(x)}\big) + b_{\mathrm{cp}(x)}$*, called nearest peak function,*
- $\mathrm{JZ}_2(x) := \max\limits_{i\in\{1,2,\ldots,k\}} a_i \cdot \mathrm{G}(x,p_i) + b_i$*, called weighted nearest peak function,*

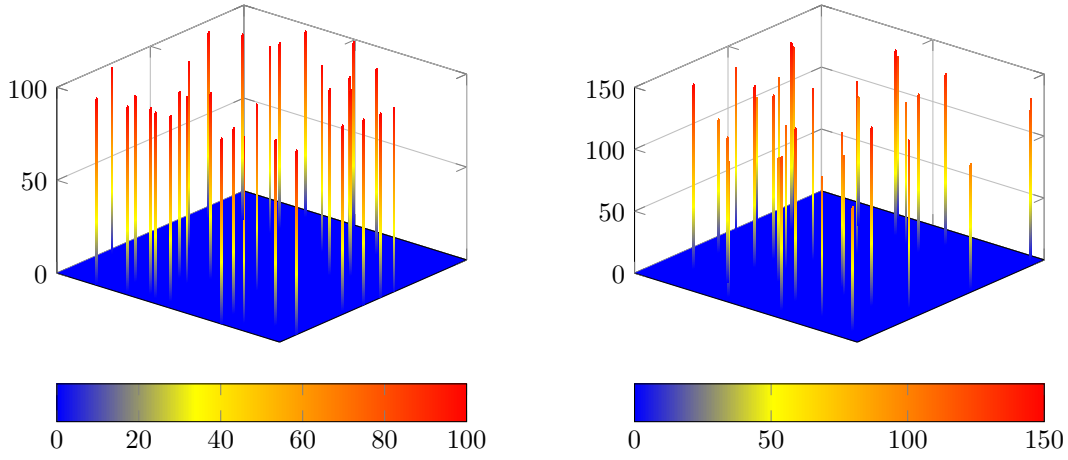
where $\mathrm{cp}(x) := \operatorname{arg\,min}\limits_{i\in\{1,2,\ldots,k\}} \mathrm{H}(x,p_i)$ *is defined by the closest peak to a search point, and* $\mathrm{G}(x,p_i)$ *indicates the proximity of x to p_i.*

The nearest peak function, JZ_1, has the fitness of a search point x determined by the closest peak $i = \mathrm{cp}(x)$ that determines the slope a_i and the offset b_i. In cases where there are multiple i that minimise $\mathrm{H}(x, p_i)$, i should additionally maximise $a_i \cdot \mathrm{G}(x, p_i) + b_i$. If there is still not a unique individual, a peak i is selected uniformly at random from those that minimises $\mathrm{H}(x, p_i)$ and those that maximises $a_i \cdot \mathrm{G}(x, p_i) + b_i$.

In Figure 2.4a we show an example landscape defined by the JZ_1 function with $k = 32$ peaks, $a_i = 1$ and $b_i = 0$ for all i peaks (all peaks have the same height). An individual in this landscape will also looks like a peak and as mentioned before, its fitness will be determined by the closest peak.

The weighted nearest peak function, JZ_2, takes the height of peaks into account. It uses the peak i that yields the largest value to determine the function value. The bigger the height of the peak, the bigger its influence on the search space in comparison to smaller peaks.

The landscape defined by JZ_2 is more difficult to explore than the landscape defined by JZ_1 due to the influence of higher peaks. In Figure 2.4b we show an example landscape defined by the JZ_2 function with $k = 32$ peaks, $a_i = 1$ and $b_i = [0, 50]$. As can be seen the height of each peak can vary, an individual in this landscape can easily be trapped by the influence of higher peaks. If the goal is to find as many peaks as possible, this setting can be used to analyse the ability of an algorithm to escape from the basin of attraction of higher peaks in order to find other peaks on the search space.



(a) JZ_1, with $k = 32$, $a_i = 1$ and $b_i = 0$.

(b) JZ_2, with $k = 32$, $a_i = 1$ and $b_i = [0, 50]$.

Figure 2.4: Example landscapes from the Jansen-Zarges Multimodal function classes. Where k represents the number of peaks in the landscape and each i-th peak is defined by its position $p_i \in \{0,1\}^n$, its slope $a_i \in \mathbb{R}^+$ and its offset $b_i \in \mathbb{R}_0^+$.

2.1.2 Multi-objective Functions

In this case, when an optimisation problem involves more than one objective function, the task of finding one or more trade-offs between solutions is known as *multi-objective optimisation*, one searches for a set of these trade-offs instead of a single one.

In our investigations we consider problems $f = (f_1, \ldots, f_m) : \{0,1\}^n \to \mathbb{R}^m$. Throughout this thesis, we assume w. l. o. g. that each function f_i, $1 \leq i \leq m$, should be maximised. As there is no single point that maximises all functions simultaneously, the goal is to find a set of so-called Pareto-optimal solutions.

Definition 2.15 (Pareto Optimality)**.** *Let $f : X \to F$, where $X \subseteq \{0,1\}^n$ is called decision space and $F \subseteq \mathbb{R}^m$ objective space. The elements of X are called decision vectors and the elements of F objective vectors. A decision vector $x \in X$ is Pareto optimal if there is no other $y \in X$ that dominates x. y dominates x, denoted as $y \succ x$, if $f_i(y) \geq f_i(x)$ for all $i = 1, \ldots, m$ and $f_i(y) > f_i(x)$ for at least one index i. A decision vector y weakly dominates x, denoted by $y \succeq x$, if $f_i(y) \geq f_i(x)$, for all i. The set of all Pareto-optimal decision vectors X^* is called Pareto set. $F^* = f(X^*)$ is the set of all Pareto-optimal objective vectors and denoted as Pareto front.*

We consider ONEMINMAX and LOTZ (Leading Ones, Trailing Zeroes) problems introduced in Giel and Lehre (2010) and Laumanns et al. (2004), respectively. ONEMINMAX generalises ONEMAX function, and LOTZ generalises LEADINGONES function to the multi-objective case. ONEMINMAX has the property that every single solution represents a point in the Pareto front and that no search point is strictly dominated by another one. The goal is to cover the whole Pareto front, i. e., to compute a set of individuals that contains for each i, $0 \leq i \leq n$, an individual with exactly i ones.

Definition 2.16 (ONEMINMAX)**.** *A pseudo-Boolean function $\{0,1\}^n \to \mathbb{N}^2$ with the objective functions*

$$\text{ONEMINMAX}(x_1, \ldots, x_n) := \left(\sum_{i=1}^{n} x_i, n - \sum_{i=1}^{n} x_i \right),$$

where the aim is to maximise the number of ones and zeroes at the same time (see Figure 2.5).

In the case of LOTZ, all non-Pareto optimal decision vectors only have Hamming neighbours that are better or worse. This fact facilitates the analysis of the population-based algorithms, which certainly cannot be expected from other multi-objective optimisation problems. Note that the Pareto front for LOTZ is given by the set of $n+1$ search points $\{1^i 0^{n-i} \mid 0 \leq i \leq n\}$.

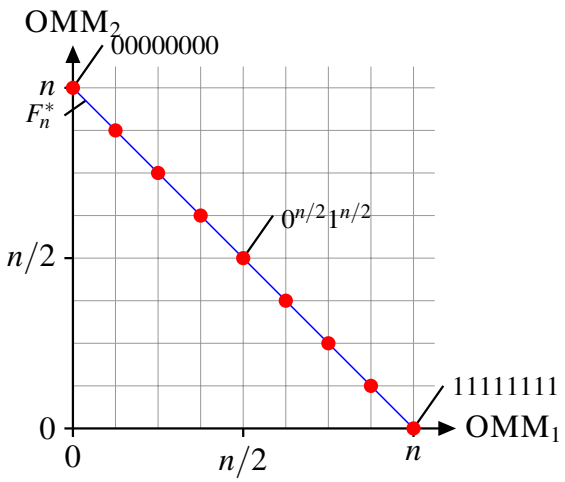


Figure 2.5: Sketch of the function ONEMINMAX (OMM) with $n = 8$. Where the red points represent the set of all Pareto-optimal decision vectors X^* called Pareto set and the blue line represents the Pareto front F_n^* that contains all the set of all Pareto-optimal objective vectors.

Definition 2.17 (LOTZ)**.** *A pseudo-Boolean function* $\{0,1\}^n \to \mathbb{N}^2$ *defined as*

$$\text{LOTZ}(x_1,\ldots,x_n) := \left(\sum_{i=1}^{n}\prod_{j=1}^{i} x_j, \sum_{i=1}^{n}\prod_{j=i}^{n}(1-x_j)\right),$$

where the goal is to simultaneously maximise the number of leading ones and trailing zeroes (see Figure 2.6).

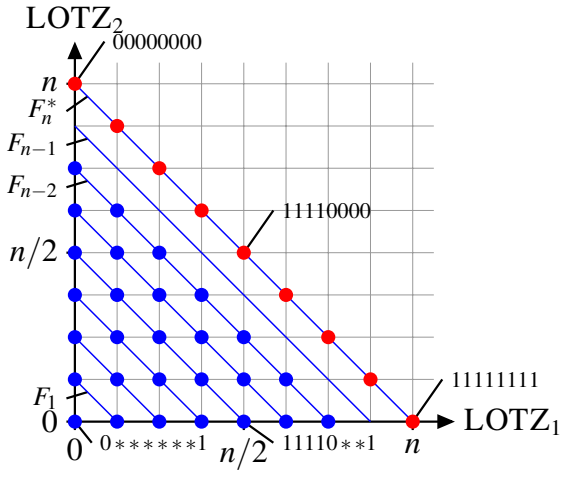


Figure 2.6: Sketch of the function LOTZ with $n = 8$. Where the blue lines represents the different fronts (F_i) that the population (blue points) has to go through before it reaches the last blue line F_n^* called Pareto front that contains the set of all Pareto-optima decision vectors X^* called Pareto set and all the set of all Pareto-optimal objective vectors.

2.2 Randomised Search Heuristics

The analysis of EAs started with very simple variants. In this section we introduce some of the most important stochastic search algorithms analysed in the runtime analysis area. All algorithms are described for the maximisation of a given fitness function f. This section is divided in two sections, for single-objective optimisation and multi-objective optimisation. Let us note that not all the algorithms mentioned in the following are subject of our investigation but for its importance on the area of runtime analysis we choose to mention them.

2.2.1 Single-Objective Algorithms

Randomised Local Search (RLS) in the binary case produces from a current solution $x_t \in \{0,1\}^n$ a new one y by flipping a randomly chosen bit (Algorithm 2). Here, the bit-flip operator refers to the mutation operator and normally this 1-bit mutation it is also known as local mutation.

Algorithm 2 Randomized Local Search (RLS)

1: Let $t := 0$ and choose $x_t \in \{0,1\}^n$ uniformly at random.
2: **while** stopping criterion **not** met **do**
3: Create y by flipping 1 bit chosen uniformly at random in x_t.
4: **if** $f(y) \geq f(x_t)$ **then** $x_{t+1} = y$ **else** $x_{t+1} = x_t$ **end if**
5: Let $t := t + 1$.
6: **end while**

The most general EA for single-objective optimisation that works with a population size μ and an offspring population size λ is the (μ+λ) EA (Algorithm 3). This algorithm uses uniform initialisation and keeps a parent population of size μ. A fitness function describes the quality of each candidate solution and in each generation, individuals from the parent population are mutated using standard bit mutation to create the offspring population of size λ. Then a new population is created by choosing the best μ individuals out of the $P \cup Q$ parent and offspring populations (plus-selection method). In case of ties, the offspring are preferred over parents. This process continues until the termination condition is satisfied.

Another important EA is the (1+1) EA defined in Algorithm 4; this algorithm is the most simple and theoretically analysed EA. It can be obtained from Algorithm 3 by setting $\mu = \lambda = 1$. It starts with a randomly chosen bitstring x_t of length n, and the algorithm produces in each iteration a child by flipping each bit of x_t with probability $1/n$. As can be seen, the (1+1) EA can be considered as a variant of the RLS with a different mutation operator.

Algorithm 3 (μ+λ) EA

Let $t := 0$ and initialise P_0 with μ individuals chosen uniformly at random.
while stopping criterion **not** met **do**
    Let $Q_t = \emptyset$.
    for $i = 1$ **to** λ **do**
        Choose x_i from P_t uniformly at random.
        Create y by flipping each bit in x_i independently with probability $1/n$.
        Let $Q_t = Q_t \cup y$.
    end for
    Create the new population P_{t+1} by choosing the best μ individuals out of the $P_t \cup Q_t$ parent and offspring population.
    Let $t := t + 1$.
end while

Algorithm 4 (1+1) EA

Let $t := 0$ and choose $x_t \in \{0,1\}^n$ uniformly at random.
while stopping criterion **not** met **do**
    Create y by flipping each bit in x_t independently with probability $1/n$.
    if $f(y) \geq f(x_t)$ **then** $x_{t+1} = y$ **else** $x_{t+1} = x_t$ **end if**
    Let $t := t + 1$.
end while

Another variant of the (μ+λ) EA with finite population size > 1 is the (μ+1) EA. This algorithm uses uniform parent selection with population size μ, with standard bit mutation (and no crossover) and elitist selection for survival. The offspring y replaces a worst individual z from the population if $f(y) \geq f(z)$ (see Algorithm 5).

Algorithm 5 (μ+1) EA

Let $t := 0$ and initialise P_0 with μ individuals chosen uniformly at random.
while stopping criterion **not** met **do**
    Choose $x \in P_t$ uniformly at random.
    Create y by flipping each bit in x independently with probability $1/n$.
    Choose $z \in P_t$ with worst fitness uniformly at random.
    if $f(y) \geq f(z)$ **then** $P_{t+1} = P_t \setminus \{z\} \cup \{y\}$ **else** $P_{t+1} = P_t$ **end if**
    Let $t := t + 1$.
end while

2.2.2 Multi-Objective Algorithms

We want to examine MOEAs that are generalisations of the RLS and the (1+1) EA. Therefore, we investigate and analyse a simple algorithm called Simple Evolutionary Multi-Objective Optimiser (SEMO) due to Laumanns et al. (2004).

SEMO starts with an initial solution $s \in \{0,1\}^n$ chosen uniformly at random. All non-dominated solutions are stored in the population P. Then, it selects a solution s uniformly at random from P, and a new search point s' is produced by the mutation step which flips one bit of s chosen uniformly at random. The new population contains for each non-dominated fitness vector $f(s)$, $s \in P \cup \{s'\}$, one corresponding search point (dominated individuals are removed from the population), and in the case where $f(s')$ is not dominated, s' is added to P (see Algorithm 6).

Algorithm 6 SEMO

```
1: Choose an initial solution s ∈ {0,1}^n uniformly at random.
2: Determine f(s) and initialize P = {s}.
3: while stopping criterion not met do
4:     Choose s uniformly at random from P.
5:     Choose i ∈ {1,...,n} uniformly at random.
6:     Define s' by flipping the i-th bit of s.
7:     if s' is not dominated by any individual in P then
8:         Add s' to P, and remove all individuals weakly dominated by s' from P.
9:     end if
10: end while
```

Applying SEMO to a single-objective optimisation problem, we obtain the RLS where in each step a single bit is flipped. Giel (2003) has introduced an algorithm called Global SEMO (GSEMO), which is shown in Algorithm 7. In GSEMO a new solution s' is created by flipping each bit from a solution s independently with probability $1/n$, then it proceeds in the same way as SEMO. GSEMO differs from SEMO by using the more general mutation operator of the (1+1) EA, in this sense GSEMO applied to single-objective optimisation problems equals the (1+1) EA.

2.3 Methods for the Analysis of Evolutionary Algorithms

In this section, we will discuss some important methods used for the runtime analysis of EAs. Such methods have already been applied in the field of randomised algorithms (Motwani and Raghavan, 1995). Many methods are accompanied by examples showing their concrete application for some the algorithms mentioned in Section 2.2.

Algorithm 7 GSEMO

1: Choose an initial solution $s \in \{0,1\}^n$ uniformly at random.
2: Determine $f(s)$ and initialize $P = \{s\}$.
3: **while** stopping criterion **not** met **do**
4: Choose s uniformly at random from P.
5: Define s' by flipping each bit in s independently with probability $1/n$.
6: **if** s' is **not** dominated by any individual in P **then**
7: Add s' to P, and remove all individuals weakly dominated by s' from P.
8: **end if**
9: **end while**

In this section we introduce different methods for the time complexity analysis of EAs. We introduce the methods used throughout this thesis, and we illustrate them by providing examples where algorithms mentioned in Section 2.2 are analysed on some example functions. Some parts of the following subsections have been freely adapted from the main three textbooks regarding the theoretical runtime analysis of EAs: Neumann and Witt (2010), Auger and Doerr (2011) and Jansen (2013), and the previous writen works: Sudholt (2008) and Pérez Heredia (2017). For any other source we will make explicit reference to each publication.

2.3.1 Standard Mutations

As mentioned in Section 2.2, most of the EAs studied in theoretical runtime analyses use standard mutations as a variation operator. The operator creates a new individuals by flipping each bit from the selected parent with probability $p_m := 1/n$. Let us remember that the mutation probability is always set to 1, in this setting, p_m refers to the mutation rate. Because of its relevance, it is important to gain some insight of the characteristics of this operator. This section is based on Section 2.3.1 from Sudholt (2008), where all these properties have been mentioned.

Using the nomenclature from the algorithms defined in Section 2.2.1, the selection for reproduction chooses an individual x. An individual y is create by flipping each bit in x independently with probability $1/n$. The number of bits changed due to mutation can be measured using the Hamming distance. So the change from x to y can be measured by $\mathrm{H}(x,y) := k \geq 1$, then the probability that the mutation creates y from x is

$$\left(\frac{1}{n}\right)^k \cdot \left(1 - \frac{1}{n}\right)^{n-k}.$$

This probability is bounded from above by $(1/n)^k$. For the lower bound $\left(1/en^k\right)$, it can be obtained by $(1-1/n)^{n-k} \geq (1-1/n)^{n-1} \geq 1/e$ with $e := \exp(1) := 2.7182\ldots$ by Lemma A.16 in the appendix.

Any search point with Hamming distance k from x have the same probability of being generated of

$$\binom{n}{k} \cdot \left(\frac{1}{n}\right)^k \cdot \left(1-\frac{1}{n}\right)^{n-k}.$$

By Lemma A.19, $\binom{n}{k} \leq n^k/k!$ yields $1/k!$ as upper bound. The following lemma concentrate the properties define above.

Lemma 2.18 (Lemma 2.3.1 in Sudholt, 2008)**.** *Consider a standard mutation of a search point x and let $1 \leq k \leq n$. The probability that a specific search point with Hamming distance k to x is created is bounded from below and above by*

$$\frac{1}{en^k} \leq \left(\frac{1}{n}\right)^k \cdot \left(1-\frac{1}{n}\right)^{n-k} \leq \frac{1}{n^k}.$$

The probability that an arbitrary search point with Hamming distance k is created is

$$\binom{n}{k} \cdot \left(\frac{1}{n}\right)^k \cdot \left(1-\frac{1}{n}\right)^{n-k} \leq \frac{1}{k!}$$

The expected time until one of the events described in Lemma 2.18 with probability p happens is geometrically distributed with parameter p. From Definition A.14, the expected waiting time equals $1/p$. The expected waiting time for a specific k-bit mutation is at least n^k and at most en^k, and the expected waiting time for a mutation flipping k-bits is at least $k!$

2.3.2 Accounting Method

This method is part of the field of *amortised analysis* for the analysis of algorithms. With amortised analysis, it is possible to show that the average cost of an operation is small, if we average over a sequence of operations, even though a single operation within the sequence might be expensive. An amortised analysis guarantees the average performance of each operation in the worst case. We will base this section in Section 17.2 in Cormen et al. (2009).

In the *accounting method*, different operations can have different "charges", with some operations charged more or less than they actually cost. The amount charged to an operation is called *amortised cost*. The most common analogy used in this scenario is: when the amortised cost is higher than the actual cost of an operation, the difference is saved as a

credit, and when the amortised cost is lower than the actual cost of an operation, the credit saved from previous operations can be used to pay the deficit on the cost.

For this method it is necessary to choose the amortised costs of operations carefully. Choosing the correct parameters for the accounting method requires as much knowledge of the problem and the complexity bounds one is attempting to prove. The main requirement when analysing with amortised costs is to ensure that the total amortised cost of a sequence of operations provides an upper bound on the total actual cost of the sequence. For example, let say that the actual cost of the i-th operation is c_i and the amortised cost of the i-th operation is a_i; it is necessary to ensure

$$\sum_{i=1}^{n} a_i \geq \sum_{i=1}^{n} c_i \tag{2.1}$$

for all n operations. As mentioned before, the credit used to pay for the deficit cost is the difference between the total amortised cost and the total actual cost $(\sum_{i=1}^{n} a_i - \sum_{i=1}^{n} c_i)$, and by inequality (2.1) we ensure that the total credit is not negative all the times. If this condition is not fulfilled, then the total amortised costs incurred at the i-th operation would be below than the total actual cost at the i-th operation; for the sequence of operations up to the i-th operation, the total amortised cost would not be an upper bound on the total actual cost.

We use this method in Chapter 6 to bound the time required to create desired offspring by selection of a parent and by mutation. The main idea is to count the expected number of mutations required to create an individual from a parent selected for mating. The sum (upper bounds) on all these times across all $0 \leq i \leq n$, is used to estimate the expected total number of generations needed for SEMO and GSEMO to find all solutions in the Pareto set on ONEMINMAX and LOTZ in the context of diversity-based parent selection.

2.3.3 Typical Run Investigations

Typical runs were introduced in the analysis of EAs (Wegener, 2002) following the consideration that sometimes the "global behaviour of a process" is predictable with high probability contrary to the local behaviour which instead, is quite unpredictable. It is often necessary to track the typical behaviour of an EA by dividing the process into i phases which are long enough to assure that some event happens with probability p_i and does not happen with probability $1 - p_i$.

According to Sudholt (2008), the total runtime can be estimated by the sum of times spent in all phases. By dividing the overall runtime in i phases it is possible to divide the search into specific goals, these goals can be specific behaviours of the algorithm or problem-based. The main benefit is that a carefully specified goal for phase i allows a more detailed analysis of the following phases. For example, when looking for an upper bound, the goal of phase i

may imply that then the goal of phase $i+1$ is reached efficiently. This phase division is also useful to make the runtime analysis easier to understand; the whole analysis is divided into different sub-analyses with specific details and ideas in order to solve the sub-task, and by combining all the sub-analysis together, the complete analysis can be obtained. By careful specification of phases and their corresponding goals, it is possible to obtain strong results on the runtime distribution of an EA (Wegener, 2002).

The following theorem is an example application for SEMO (Algorithm 6) on LOTZ (Definition 2.17). Here, the analyses is divided into two phases; the first phase covers the time until the first Pareto-optimal point is found and the second phase stops if the entire Pareto set is found.

Theorem 2.19 (Adapted from Lemma 1 and 2 in Laumanns et al., 2004)**.** *The expected time for SEMO to cover the whole Pareto front on* LOTZ *is* $\Theta\left(n^3\right)$.

Proof. For the first phase let us consider the time until the first Pareto-optimal is found. Using the notation from Definition 2.15 and Algorithm 6, SEMO starts with a single individual s in the population, and only one individual s' can be created due to 1-bit mutation that either produces an individual that dominates s or it is dominated by s. Hence if an offspring is accepted, it will replace the parent from which it is produced if and only if $s \in F_i$ and $s' \in F_j$ with $i < j$ (i. e., s' belongs to a better front). Since this can only happens by flipping one specific bit (increase the number of leading ones or trailing zeroes) with probability $1/n$, the waiting time to leave F_i is $\mathrm{O}(n)$. The time until F^* is reached is $\mathrm{O}\left(n^2\right)$ since there are at most $\mathrm{O}(n)$ such steps necessary.

Once we have reached the Pareto front, the goal of the next phase consists in finding all the individuals in the Pareto set. As the population is a subset of the Pareto front, only these individuals can create a new Pareto-optimal point. Let i be the current number of individuals found on the Pareto front. A new individual on the front is created if the leading ones or the trailing zeroes is improved; this can be achieved by sampling one individual from the i with probability of at least $1/i$ and at most $2/i$ (two individuals that can produce a new individual in opposite directions), and the probability of a subsequent successful mutation is at least $1/n$ and at most $2/n$.

Hence, by summing up the expected runtimes for finding the $(i+1)$-th Pareto-optimal solution which is at most in and at least $i/2 \cdot n/2 = in/4$, we bound the waiting time until the entire Pareto set is found by at most $\sum_{i=1}^{n-1} ni = n^3/2 - n^2/2$ and at least $\sum_{i=1}^{n-1} ni/4 = n^3/8 - n^2/8$. □

Here, as well as in several other studies, we will make use of this method to derive upper bounds for SEMO and GSEMO on ONEMINMAX and LOTZ in the context of diversity-based parent selection in Chapter 6.

2.3.4 Coupon Collector Problem

Occupancy problems are very important stochastic processes and are the core of the analyses of many randomized algorithms ranging from data structures to routing in parallel computing. In such problems we define m objects, normally referred as "balls" being randomly assigned to one of n distinct classes, normally referred as "bins". In other words, each ball is placed in a bin chosen independently, uniformly at random, and normally the problem is translated to questions like: what is the maximum number of balls in any bin? what is the expected number of bins with k balls in them (Motwani and Raghavan, 1995)?

An important variant of the occupancy problem is the one known as *coupon collector's problem*. Suppose that a certain product contains one of n different coupons. Once you obtain one of every type of coupon, you can exchange them for a prize. Assuming that the coupon in each product is chosen independently and uniformly at random from the n possibilities and you do not collaborate with others to collect the coupons, how many products must you buy before you obtain at least one of every type of coupon (Mitzenmacher and Upfal, 2005)? The formal definition of the coupon collector problem is given below (Doerr, 2012).

Theorem 2.20 (Coupon Collector)**.** *The expected time to collect all n coupons is $n\mathrm{H}_n$, where $\mathrm{H}_n := \sum_{i=1}^{n} 1/i$ is the n-th Harmonic number (see Lemma A.17). Since $\mathrm{H}_n = \ln n + \Theta(1)$, the coupon collector needs an expected time of $n \ln n + \Theta(n)$.*

Proof. Let X be a random variable that represents the number of products bought until at least one of every type of coupon is obtained. Now determine $\mathrm{E}[X]$. X_i is a geometric random variable representing the number of products bought while i different coupons have been obtained, then $X = \sum_{i=1}^{n} X_i$. The probability of obtaining a new coupon is $\mathrm{Prob}_i = (n-i)/n$. By Definition A.14 we got

$$\mathrm{E}[X_i] = \frac{1}{\mathrm{Prob}_i} = \frac{n}{n-i}.$$

And by linearity of expectation (see Theorem A.9), we have that

$$\mathrm{E}[X] = \mathrm{E}\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} \mathrm{E}[X_i] = \sum_{i=1}^{n} \frac{n}{n-i} = n \sum_{i=1}^{n} \frac{1}{i}.$$

Thus, for the coupon collector's problem, the expected time required to obtain all n coupons is $n \ln n + \Theta(n)$. □

This method is useful when analysing elitist algorithms on functions of unitation. We show an example of its application by analysing RLS (Algorithm 2) on the ONEMAX function (see Definition 2.9).

Theorem 2.21. *The expected optimisation time for the RLS starting from* 0^n *on* ONEMAX *is* $n\mathrm{H}_n = n\ln n + \Theta(n)$.

Proof. Let i denote the number of 1-bits in the current solution. By local mutations, it can only create Hamming neighbours. The algorithm can only create an improvement if one of the $n-i$ bits (the remaining 0 bits) are flipped creating an individual with higher ONEMAX fitness. Since each bit has a probability of being flipped of $1/n$, the probability of flipping one of the remaining $n-i$ bits is $(n-i)/n$, exactly as the coupon collector problem. Using Theorem 2.20 completes the proof. □

2.3.5 Fitness-based Partitions

This simple method has been used for a wide class of problems. We assume that we are considering a stochastic search algorithm that works in each iteration with at least one solution that produces at least one offspring. All variants of $(\mu+\lambda)$ EA fit into this scenario. Assume that we are working in a search space S and consider w. l. o. g. a function $f : S \to \mathbb{R}$ that should be maximized. S is partitioned into disjoint sets $A_1, \ldots, A_m$ such that $A_1 <_f A_2 <_f \cdots <_f A_m$ holds, where $A_i <_f A_j$ means that $f(a) < f(b)$ holds for all $a \in A_i$ and all $b \in A_j$. In addition, A_m contains only optimal search points. An illustration is given in Figure 2.7, here we have partitioned the search space $S \in \{0,1\}^n$ into the sets $A_1, \ldots, A_8$ where the best individuals point towards to A_8.

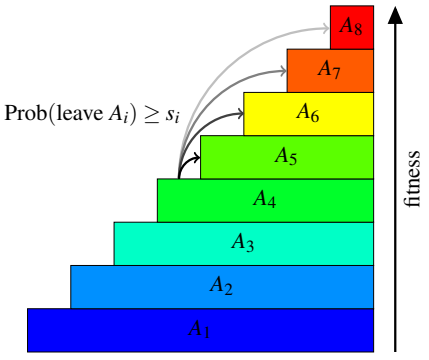


Figure 2.7: Illustration of fitness-based partitions with $n = 8$. The darker the arrow, more easily is to jump from one partition to another with higher fitness.

We denote for a search point $x \in A_i$ by $\mathrm{Prob}(x)$ the probability that in the next step a solution $x' \in A_{i+1} \cup \cdots \cup A_m$ is produced. Let $\mathrm{Prob}_i = \min_{a \in A_i} \mathrm{Prob}(x)$ be the smallest

probability of producing a solution with a higher partition number (Neumann and Witt, 2010). Here we show the formal definition of the fitness-based partition method (Wegener, 2002).

Theorem 2.22 (Fitness-based Partition Method in Wegener, 2002)**.** *For two sets $A, B \subseteq \{0,1\}^n$ and fitness function f let $A <_f B$ if $f(a) < f(b)$ for all $a \in A$ and all $b \in B$. Consider the* $(\mu+\lambda)$ EA *and a partition of the search space into non-empty set $A_1, \ldots, A_m$ such that*

$$A_1 <_f A_2 <_f \cdots <_f A_m$$

and A_m only contains global optima. Let s_i be a lower bound on the probability of creating a new offspring in $A_{i+1} \cup \cdots \cup A_m$, provided the population contains a search point in A_i. Then the expected number of generations for the $(\mu+\lambda)$ EA *to find an optimum is bounded by*

$$\sum_{i=1}^{m} \frac{1}{s_i}.$$

As an example of its application, we apply the method to the analysis of the (1+1) EA on ONEMAX already investigated by Droste et al. (2002).

Theorem 2.23. *The expected optimisation time for the* (1+1) EA *on* ONEMAX *is* $O(n \log n)$.

Proof. When analysing unitation functions, it is natural to partition the space in sets depending on the number of ones of a bitstring. So let's consider a set A_i with i ones. Then we can calculate the lower bound on the probability Prob_i to reach a partition A_j where $j > i$. It is just necessary to flip one of the $n-i$ zeroes to one, and to leave the remaining bits unchanged. Since each bit has a probability of $1/n$ of being flipped and not being flipped of $(1-1/n)$, the probability of flipping one of the zero bits to one is $(n-i)/n$ and the probability of leaving the remaining bits unchanged is $(1-1/n)^{n-1}$. Hence the probability is bounded from below as follows

$$\text{Prob}_i \geq \frac{n-i}{n} \cdot \left(1 - \frac{1}{n}\right)^{n-1} \geq \frac{n-i}{en}.$$

The inequality follows because according to Lemma A.16, $(1-1/n)^{n-1} \geq 1/e$. So by Theorem 2.22, the expected optimisation time is at most

$$\sum_{i=0}^{n-1} \frac{1}{p_i} \leq \sum_{i=0}^{n-1} \frac{en}{n-i} = en \sum_{i=1}^{n} \frac{1}{i} \leq en \cdot (\ln n + 1) = O(n \log n). \qquad \square$$

Now, we would like to mention that since we are using O-notation, constant factors are usually ignored or omitted. Changing from one base of a logarithm to another ($\ln a = \frac{\log a}{\log e}$) changes the value of the logarithm by a only a constant factor. Computer scientists find 2

to be the most natural base for logarithms because so many algorithms and data structures involve splitting a problem into two parts (Cormen et al., 2009). From now one we will use this convention and we will make use of constant terms when needed.

Another example of the application of the fitness-based partitions method but on a population-based EA it is the analysis of the (μ+1) EA on ONEMAX already investigated in Witt (2006).

Theorem 2.24 (Adapted from Theorem 2 from Witt, 2006)**.** *The expected optimisation time for the* (μ+1) EA *with* $\mu = \text{poly}(n)$ *on* ONEMAX *is* $O(\mu n + n \log n)$.

Proof. The proof follows the progress to the optimum by the potential L, defined as the maximum ONEMAX value of the current population. Let us start by mentioning that for the resulting optimisation time T, it is necessary to add μ function evaluations for the initial population, so the overall number of function evaluations is $O(\mu + T)$. In the following we will define the value for T.

In order to increase L, it is necessary to select one of the existing i individuals with L ones. Since each individual has at least $n - L$ zeroes, with probability of being selected of i/μ, then considered probability is bounded below by

$$\frac{i}{\mu} \cdot \frac{n-L}{n} \cdot \left(1 - \frac{1}{n}\right)^{n-1} \geq \frac{i(n-L)}{e\mu n}$$

Now, the objective is to increase number of individuals with the current best fitness from 1 to $\min\{\mu, n/(n-L)\}$. However, the (μ+1) EA can produce replicas of individuals. If the number of best individuals is i, the probability of selecting one of the best i individuals is at least i/μ, and the probability of leaving the remaining bits untouched of $(1-1/n)^n \geq 1/(2e)$, then probability of creating a clone of a best individual is bounded below by $i/(2e\mu)$.

If a replica is generated, this new individual replaces a worst individual and increases the number of best individuals with L ones. Now assume pessimistically that L does not increase until we have at least $\min\{\mu, n/(n-L)\}$ best individuals. The expected time for this is at most

$$\sum_{i=1}^{\lceil n/(n-L) \rceil - 1} \frac{2e\mu}{i} \leq 2e\mu \ln\left(\frac{en}{n-L}\right).$$

The expected waiting time for increasing the number of current best is

$$2e\mu \sum_{L=0}^{n-1} \ln\left(\frac{en}{n-L}\right) = 2e\mu \ln\left(\frac{e^n n^n}{n!}\right) \leq 2e\mu \log\left(e^{2n}\right) = 4e\mu n = O(\mu n)$$

as upper bound (using Stirling's formula to estimate $n! \geq (n/e)^n$, see Lemma A.18). Now, after the desired number of individuals with value L has been obtained, the time for increasing L is at most

$$\frac{e\mu n}{\min\{\mu, n/(n-L)\} \cdot (n-L)} = \frac{e\mu n}{\min\{\mu(n-L), n\}} \leq \frac{e\mu n}{\mu(n-L)} + \frac{e\mu n}{n}.$$

Hence, the expected waiting time to increase all L ones is at most

$$\sum_{L=0}^{n-1} \left(\frac{e\mu n}{\mu(n-L)} + \frac{e\mu n}{n} \right) \leq en\ln(en) + e\mu n = \mathrm{O}(\mu n + n\log n),$$

and the total expected runtime is at most $\mathrm{O}(\mu) + \mathrm{O}(\mu n) + \mathrm{O}(\mu n + n\log n) = \mathrm{O}(\mu n + n\log n)$. □

2.3.6 Markov Chains

In this section we will describe the stochastic model called *Markov chain* used to describe a sequence of possible events in which the probability of each event depends only on the state attained in the previous event. We will base the present section on the definitions and representations defined in Sections 6.2 and 7.1 from Motwani and Raghavan (1995) and Mitzenmacher and Upfal (2005), respectively.

A Markov chain M is a discrete-time stochastic process defined over a set of states S in terms of a matrix P of transition probabilities. The set S is either finite or countable infinite. Consider a sequence of random variables $X_0, X_1, \ldots$, defined on state-space S and suppose that the set of possible values of these random variables is $0, 1, \ldots, M$. It will be helpful to interpret X_t as being the state of some system at time X_t, and in accordance with this interpretation, we say that the system is in state i at time t if $X_t = i$. The sequence of random variables is said to form a *Markov chain* if, each time the system is in state i, there is some fixed transition probability—call it p_{ij}—that the system will next be in state j, given that the current state is i. That is, for all $i, j \in S$, we have $0 \leq p_{ij} \leq 1$, and $\sum_j p_{ij} = 1$.

An important property of a Markov chain is the *memorylessness property*: the future behaviour of a Markov chain depends only on its current state, and not on how it arrived at the present state. This follows from the observation that the transition probabilities p_{ij} depend only on the current state i. We will denote by X_t the state of the Markov chain at time t; thus, the sequence $\{X_t\}$ specifies the history or the evolution of the Markov chain. The memorylessness property can be stated more formally as follows:

$$\mathrm{Prob}(X_{t+1} = j \mid X_t = i, X_{t-1} = i_{t-1}, \ldots, X_1 = i_1, X_0 = i_0) = \mathrm{Prob}(X_{t+1} = j \mid X_t = i) = p_{ij}.$$

A Markov chain (indeed, a random walk) need not have a prespecified initial state; in general, its initial state X_0 is permitted to be chosen according to some probability distribution over S. Of course, an initial probability distribution includes as a special case the deterministic specification that the initial state X_0 be i. Given a distribution for the initial state X_0, we have a probability distribution for the history $\{X_t\}$. For states $i, j \in S$, define the *t-step transition probability* as $p_{ij}^t = \text{Prob}(X_t = j \mid X_0 = i)$. Given an initial state $X_0 = i$, the probability that the first transition into state j occurs at time t is denoted by r_{ij}^t and is given by

$$r_{ij}^t = \text{Prob}(X_t = j, \text{ and, for } 1 \leq s \leq t-1,\ X_s \neq j \mid X_0 = i).$$

Also, for $X_0 = i$, the probability that there is a visit to (transition into) state j at some point time $t > 0$ is denoted by f_{ij}, and is given by

$$f_{ij} = \sum_{t>0} r_{ij}^t.$$

And the expected number of time steps to reach state j starting from state i is denoted by h_{ij} and is given by

$$h_{ij} = \sum_{t>0} t \cdot r_{ij}^t$$

for $f_{ij} = 1$, and $h_{ij} = \infty$ otherwise. Now, it is convenient to arrange the transition probabilities p_{ij} in a square array as follows:

$$\begin{bmatrix} p_{00} & p_{01} & \cdots & p_{0M} \\ p_{10} & p_{11} & \cdots & p_{1M} \\ \vdots & \vdots & \ddots & \vdots \\ p_{M0} & p_{M1} & \cdots & p_{MM} \end{bmatrix}$$

Such an array is called a *matrix*. Knowledge of the transition matrix and of the distribution of X_0 enables us, in theory, to compute all probabilities of interest. For instance, the joint probability mass function of $X = 0, \ldots, X_n$ is given by

$$\begin{aligned} &\text{Prob}(X_n = i_n, X_{n-1} = i_{n-1}, \ldots, X_1 = i_1, X_0 = i_0) \\ &\quad = \text{Prob}(X_n = i_n \mid X_{n-1} = i_{n-1}, \ldots, X_0 = i_0)\text{Prob}(X_{n-1} = i_{n-1}, \ldots, X_0 = i_0) \\ &\quad = p_{i_{n-1}, i_n}\text{Prob}(X_{n-1} = i_{n-1}, \ldots, X_0 = i_0), \end{aligned}$$

and continual repetition of this argument demonstrates that the preceding is equal to

$$p_{i_{n-1},i_n} p_{i_{n-2},i_{n-1}} \cdots p_{i_1,i_2} p_{i_0,i_1} \text{Prob}(X_0 = i_0).$$

An attempt of building a general framework for analysing the average hitting times of EAs by using the theory of Markov chains was made by He and Yao (2003). However, the generality of this approach was at the cost of its applicability. Although they derived explicit solutions for some case studies, the authors recognised that in general it might very difficult to solve the proposed equations. These general limitations in deriving expected time expressions from the transition matrix highlight the necessity of introducing other randomised algorithm analysis tools that may be used as "tricks" to gather information about the Markov process without having to build the exact Markov chain model.

Another method based on Markov chain theory was presented by Sudholt (2011b). Here, the author exploited the link between runtime and mixing time and applied the *coupling* technique to derive estimations of the algorithm's runtime. In Chapter 4, we will partly use this approach and the above introduced concepts from Markov chain theory.

2.3.7 Family Trees

In this section we analyse the dynamics within the population by means of so-called *family trees*. Parts of this section are taken from Covantes Osuna and Sudholt (2018b), which will be presented in Chapter 4 for the analysis of the choice of the population size for the clearing diversity-preserving mechanism (Section 4.3).

The analysis of EAs with family trees has been introduced by Witt (2006) for the analysis of the (μ+1) EA (Algorithm 5). According to Witt, a family tree is a directed acyclic graph whose nodes represent individuals and edges represent direct parent-child relations created by a mutation-based EAs. After initialisation, for every initial individual r^* there is a family tree containing only r^*. We say that r^* is the root of the family tree $\text{T}(r^*)$. Afterwards, whenever the algorithm chooses an individual $x \in \text{T}(r^*)$ as parent and creates an offspring y out of x, a new node representing y is added to $\text{T}(r^*)$ along with an edge from x to y. That way, $\text{T}(r^*)$ contains all descendants from r^* obtained by direct and indirect mutations.

There may be family trees containing only individuals that have been deleted from the current population. As μ individuals survive in every selection, at least one tree is guaranteed to grow. A subtree of a family tree is, again, a family tree. A (directed) path within a family tree from x to y represents a sequence of mutations creating y out of x. The number of edges on a longest path from the root r^* to a leaf determines the depth of $\text{T}(r^*)$.

Witt (2006) showed how to use family trees to derive lower bounds on the optimisation time of mutation-based EAs. Suppose that after some time t the depth of a family tree $\mathrm{T}(r^*)$ is still small. Then typically the leaves are still quite similar to the root.

Lemma 2 in Witt (2006) bounds the probability for having random family trees with large depth. It states that with overwhelming probability, a family tree of the (μ+1) EA becomes no deeper than the total number of mutations performed in a single run. The tree becomes wide, but a flat tree means that on any path from the root, few mutations occur.

Here we make use of Lemma 1 in Sudholt (2009) (which is an adaptation from Lemma 2 and proof of Theorem 4 in Witt, 2006) to show that the individuals in $\mathrm{T}(r^*)$ are still concentrated around r^*. If the distance from r^* to all optima is not too small, then it is unlikely that an optimum has been found after t steps.

Lemma 2.25 (Adapted from Lemma 1 in Sudholt, 2009)**.** *Let r^* be an individual entering the population in some generation t^*. The probability that within the following t/λ generations some $y^* \in \mathrm{T}(r^*)$ with $\mathrm{H}(r^*, y^*) \geq 8t/\mu$ is $2^{-\Omega(t/\mu)}$.*

In the following we will just refer to t generations instead of t/λ since the (μ+1) EA only creates just one offspring and we will describe a general lower bound for functions with a unique global optimum. This lower bound will be useful to prove a lower bound for the (μ+1) EA on the OneMax function.

Theorem 2.26 (Adapted from Theorem 4 in Witt, 2006)**.** *Let $f : \{0,1\}^n \to \mathbb{R}$ be an arbitrary function with a unique global optimum, let $\mu = \mathrm{poly}(n)$, then the expected optimisation for the (μ+1) EA on f is $\Omega(\mu n + n \log n)$.*

Proof. Let us start proving the lower bound $\Omega(n \log n)$. For this case a small population $\mu \leq (\log n)/2$ hold. Consider an initial individual x that has a bit $i \in 1, \ldots, n$ that differs from the optimal. This i bit from individual x differs from the optimum with probability $1/2$, and with probability $(1/2)^\mu \geq n^{-1/2}$ all initial individuals in the population have this bit different from the optimal. Since this probability holds for any bit position i, the expected number of bits that are not set correctly in any individual of the initial population is bounded below by $n \cdot n^{-1/2} = n^{1/2}$. By Chernoff bounds (cf. Lemma A.15), at least $\sqrt{n}/2$ bits are different in all initial individuals with probability $1 - 2^{-\Omega(\sqrt{n})}$. Therefore, assuming that there are $\sqrt{n}/2$ different bits, the probability that at least one of these bits is never flipped within the first $t := (n-1)(\ln n/2)$ generations is bounded below by

$$1 - \left(1 - \left(1 - \frac{1}{n}\right)^{(n-1)(\ln n/2)}\right)^{\sqrt{n}/2} \geq 1 - \left(1 - \frac{1}{\sqrt{n}}\right)^{\sqrt{n}/2} \geq 1 - e^{-1/2},$$

which implies that t steps are required with probability at least $(1-e^{-1/2}-2^{-\Omega(\sqrt{n})})\cdot(n-1)(\ln n/2)=\Omega(1)\cdot\Omega(n\ln n)=\Omega(n\log n)$.

Now for the lower bound $\Omega(\mu n)$, we define a larger population size $\mu>(\log n)/2$. We consider the first $s:=c\mu n$ generations, where $c>0$ is some small constant. The idea is that in s steps, each family tree created by the (μ+1) EA does not contain nodes labelled with optimal individuals with high probability. We know that the probability to have a family tree of depth greater than $3cn$ is bounded above by $2^{-\Omega(n)}$ (Lemma 2 in Witt, 2006). This implies that the expected depth of any family tree is bounded above by $3cn$ (since $\mu=\text{poly}(n)$).

For family tress whose depth is bounded by $3cn$, each offspring is the result of standard bit mutation, the Hamming distance between any two search points on the same path or sequence is bounded by the number of bits flipped which is at most $3cn$. By applying Chernoff bounds (cf. Lemma A.15) with respect to the upper bound $4cn$, we obtain that the probability of an individual of Hamming distance at least $8cn$ to r^* emerging on a particular path is at most $e^{-4cn/3}$. Taking the union bound over all possible paths in the family tree still gives a failure probability of $2^{-\Omega(n)}$ (Lemma 2.25). We see that in only $8cn$ function evaluations the optimum is not found with probability close to 1. This implies that the expected optimisation time for the (μ+1) EA on f is at least $\Omega(\mu n)$.

The total expected runtime is at least $\Omega(\mu n+n\log n)$. □

The general lower bound in Theorem 2.26 covers a wide range of unimodal functions. Since ONEMAX is a function with a unique global optimum, Theorem 2.24 and Theorem 2.26 imply the following corollary.

Corollary 2.27. *The expected optimisation time for the* (μ+1) EA *with* $\mu=\text{poly}(n)$ *on* ONEMAX *is* $\Theta(\mu n+n\log n)$.

2.3.8 Drift Analysis

Drift analysis dates back to Hajek (1982) by studying the *stochastic drift*, i. e., the expected change of a stochastic process' value.

Definition 2.28 (Stochastic Drift)**.** *Let* $\{X_t\}_{t\geq 0}$ *be a stochastic process over a finite state space* Ω*. The drift is defined as* $\Delta(X_t,t):=\mathrm{E}[X_t-X_{t+1}\mid X_t]$.

Then it was introduced in the study of the computational complexity of EAs and randomised algorithms by He and Yao (2001). In this thesis we will only use drifts that do not depend on the current time. Hence, from now on we will write $\Delta(X_t)$ for the sake of simplicity and on the following definition. Given a Markov process $\{X_t\}_{t\geq 0}$ over a search

space S and a distance function $g : S \to \mathbb{R}_0^+$ mapping each state to a non-negative real number, the one-step drift at time t of the Markov process is defined as

$$\Delta(X_t) = g(X_t) - g(X_{t+1}).$$

The *drift* $\Delta(X_t)$ represents the random decrease in distance to the optimum obtained by the algorithm in one step at time t. This decrease is the rate of change in position with respect to time, the drift is the expected rate of change of the stochastic process. The idea behind drift analysis is quite simple. If the current process is at distance d from the optimum and at each step there is an improvement (i. e., a positive drift) towards the optimum of at least $\delta > 0$, then the optimal solution will be found in at most d/δ steps. We are interested in the first hitting time T of a target state with g-value 0. A natural example is to choose g as the Hamming distance of the population to a global optimum or in the context of typical runs, define g as a some sort of distance to the goal of the current phase.

There are several methods available in the literature that can be used to define the g-value. These methods are called *additive* (He and Yao, 2001), *multiplicative* (Doerr et al., 2012), *variable drift* (Johannsen, 2010) and *negative* (Oliveto and Witt, 2011, 2012) drift theorems. The main motivation in the development and use of this methods it is based on the idea that it is easier to estimate the drift of a random sequence rather than using a Markov chain

The additive drift will use constant bounds for the real drift, whereas a multiplicative drift requires a progress which multiplicatively depends on the current potential value. The variable drift is an generalisation of the previous two drift theorems where the drift is monotonically increasing with the current state. The negative drift is considered in cases where the expected drift is negative on some interval, then the algorithm is moving in expectation away from the optimum rather than towards it.

2.3.8.1 Additive Drift

In order to derive upper or lower bounds on $\mathrm{E}[T]$, it is not necessary that δ describes the drift exactly. If δ is a lower bound for the drift and nevertheless $\delta > 0$, then we can derive an upper bound $\mathrm{E}[T] \leq \mathrm{E}[g(x_0)]/\delta$. Symmetrically, if δ is an upper bound for the drift and $\delta > 0$, a lower bound $\mathrm{E}[T] \geq \mathrm{E}[g(X_0)]/\delta$ can be shown.

Both bounds (upper and lower) have been derived in separate studies. The upper bound can be found in Lemma 6 in Wegener and Witt (2005) and the lower bound in Lemma 12 in Jägersküpper (2007). We use Theorem 2.3.8 from Sudholt (2008) that extends the results from He and Yao (2004) with bounds for the unconditional expected first hitting time $\mathrm{E}[T]$.

Theorem 2.29 (Additive Drift, Sudholt, 2008)**.** *Consider a Markov process* $\{X_t\}_{t\geq 0}$ *with state space* S *and a function* $g : S \to \mathbb{R}_0^+$*. Let* $T := \inf\{t \geq 0 : g(X_t) = 0\}$.

1. *If there exists* $\delta > 0$ *such that for every time* $t \geq 0$ *and every state* X_t *with* $g(X_t) > 0$ *the condition* $\mathrm{E}[g(X_t) - g(X_{t+1}) \mid X_t] \geq \delta$ *holds, then*

$$\mathrm{E}[T \mid X_0] \leq \frac{g(X_0)}{\delta} \quad \textit{and} \quad \mathrm{E}[T] \leq \frac{\mathrm{E}[g(X_0)]}{\delta}.$$

2. *If there exists* $\delta \in \mathbb{R}^+$ *such that for every time* $t \geq 0$ *and every state* X_t *with* $g(X_t) > 0$ *the condition* $\mathrm{E}[g(X_t) - g(X_{t+1}) \mid X_t] \leq \delta$ *holds, then*

$$\mathrm{E}[T \mid X_0] \geq \frac{g(X_0)}{\delta} \quad \textit{and} \quad \mathrm{E}[T] \geq \frac{\mathrm{E}[g(X_0)]}{\delta}.$$

Let us show an example of its application, we analyse the (1+1) EA on the ONEMAX function. This setup will help us as a baseline for the following drift theorems.

Theorem 2.30. *The expected optimisation time of the* (1+1) EA *on* ONEMAX *is* $\mathrm{O}(n^2)$.

Proof. Let x_t be the current search point in generation t. We choose the distance function as the distance of missing ones from the optimum $g(x_t) := n - \textsc{OneMax}(x_t)$. If the initial solution x_0 has i ones, it is enough to increase the number of ones by not flipping non of the i ones (with probability $(1-1/n)^i$) and flipping one of the 0-bit is flipped (with probability $1/n$). This event has probability at least $1/(en)$ and then $g(x_{t+1}) = g(x_t) - 1$. As the ONEMAX-value cannot decrease, this implies $\mathrm{E}[g(x_t) - g(x_{t+1})] \leq 1/(en)$ if $\textsc{OneMax}(x_t) > 0$. Invoking Theorem 2.29 with $\delta := 1/(en)$ yields the upper bound

$$\mathrm{E}[T] \leq \frac{\mathrm{E}[g(X_0)]}{\delta} \leq \frac{n}{1/(en)} = e \cdot n^2 = \mathrm{O}(n^2). \qquad \square$$

2.3.8.2 Multiplicative Drift

If the potential used is not properly defined, the drift can yield too loose results for the runtime. The *multiplicative drift theorem* is inspired by the multiplicative weight decrease method (Arora et al., 2012) introduced and used to slightly improve the constants (Doerr et al., 2012). The multiplicative drift theorem is tailored towards applications where there is a logarithmic factor in the runtime bound, a slowdown that seems to appear naturally as EAs approach the optimum.

The multiplicative drift theorem was extended to also include a probability tail bound in addition to an upper bound on the expected runtime. This latest version is stated as follows (Doerr and Goldberg, 2010).

Theorem 2.31 (Multiplicative Drift, Doerr and Goldberg, 2010)**.** *Let $\{X_t\}_{t\geq 0}$ be a sequence of random variables taking values in some set S. Let $g : S \to \mathbb{R}_0^+$ and assume that $g_{\max} := \max\{g(x) \mid x \in S\}$ exists. Let $T := \min\{t \geq 0 : g(X_t) = 0\}$. If there exists $\delta > 0$ such that*

$$\mathrm{E}[g(X_{t+1}) \mid g(X_t)] \leq (1-\delta)g(X_t)$$

then

$$\mathrm{E}[T] \leq \frac{1}{\delta}(1+\ln g_{\max})$$

and for every $c > 0$

$$\mathrm{Prob}\left(T > \frac{1}{\delta}(\ln g_{\max} + c)\right) \leq e^{-c}$$

Let us show a case study for the multiplicative drift theorem for the (1+1) EA on ONEMAX.

Theorem 2.32. *The expected optimisation time of the* (1+1) EA *on* ONEMAX *is at most* $en(1+\ln n)$.

Proof. Let X_t be the number of 0-bits at time t. Each of those X_t bits has a probability of being mutated of $1/n$ and leaving the remaining 1-bits unchanged with probability $(1-1/n)^{n-X_t}$. Hence, the probability of creating an individual with 1-bit more is bounded as follows:

$$\mathrm{Prob}(X_t - X_{t+1} \mid X_t) \geq \frac{X_t}{n} \cdot \left(1-\frac{1}{n}\right)^{n-X_t} \geq \frac{X_t}{en} \geq \frac{1}{en}.$$

This follows by using $(1-1/n)^{n-X_t} \geq 1/e$ since $n - X_t < n$ and $X_t \geq 1$. Finally, this implies

$$\mathrm{E}[X_{t+1} \mid X_t] \leq \left(1-\frac{1}{en}\right) X_t.$$

Applying the multiplicative drift theorem (Theorem 2.31) with $\delta = 1/en$ and $g_{\max} = n$ we obtain a runtime of $\mathrm{E}[T] \leq en(1+\ln n)$. □

2.3.8.3 Variable Drift

The *variable drift theorem* allows any drift $\mathrm{E}[X_t - X_{t+1} \mid X_t = k] \geq h(k)$, as long as $h(k)$ is monotone increasing. This assumption is sensible, since one expects that, in general, an EA slows its progress when is approaching the optimum. However, there can be drift expressions which are not monotone increasing. The following theorem is a straightforward extension of the variable drift theorem (Johannsen, 2010) towards reaching any state smaller than some

threshold a. It can be derived with simple adaptations to the proof in Rowe and Sudholt (2014).

Theorem 2.33 (Generalised Variable Drift, Johannsen, 2010)**.** *Consider a stochastic process $X_0, X_1, \ldots$ on $\{0,1,\ldots,m\}$. Suppose there is a monotonic increasing function $h : \mathbb{R}^+ \to \mathbb{R}^+$ such that*

$$\mathrm{E}[X_t - X_{t+1} \mid X_t = k] \geq h(k)$$

for all $k \in \{a,\ldots,m\}$. Then the expected first hitting time of the set $\{0,\ldots,a-1\}$ for $a \in \mathbb{N}$ is at most

$$\frac{a}{h(a)} + \int_a^m \frac{1}{h(x)}\,\mathrm{d}x.$$

Again, we exemplify the use of the variable drift on the same problem, (1+1) EA on OneMax, to show that it outperforms both the multiplicative and additive drift.

Theorem 2.34 (Adapted from Theorem 3 from Doerr et al., 2011)**.** *The expected runtime of the* (1+1) EA *on* OneMax *is at most $en\ln n - 0.369n + \mathrm{O}(1)$.*

Proof. We apply Theorem 2.33 with X_t defined to be the number of 0-bits at time t. Now we bound the expected number of 0-bits flipped into 1-bits in one iteration. Note that the probability that the number of 1-bits increases by i is at least the probability that i 0-bits are flipped while no 1-bit is flipped. Let $X_t = x$, then we have

$$\mathrm{E}[X_t - X_{t+1} \mid X_t] \geq \frac{x}{n}\left(1-\frac{1}{n}\right)^{n-x}.$$

Note that $h(x)$ is monotone increasing in x, let it be our h-function in the Variable Drift (Theorem 2.33), so $h(x) := \frac{x}{n}\left(1-\frac{1}{n}\right)^{n-x}$, $a = 1$ and $m = n$. Assuming that initialisation is at the 0^n string $\mathrm{E}[T] \leq \mathrm{E}[T \mid X_0 = n]$ and hence

$$\begin{aligned}
\mathrm{E}[T] &\leq \frac{1}{h(1)} + \int_1^n \frac{1}{h(x)}\,\mathrm{d}x \\
&= \frac{n}{\left(1-\frac{1}{n}\right)^{n-1}} + n\left(1-\frac{1}{n}\right)^{-n}\int_1^n \frac{\left(1-\frac{1}{n}\right)^x}{x}\,\mathrm{d}x \\
&\leq \frac{n}{\left(1-\frac{1}{n}\right)^{n-1}} + n\left(1-\frac{1}{n}\right)^{-n}\int_1^n \frac{e^{-x/n}}{x}\,\mathrm{d}x \qquad (2.2)
\end{aligned}$$

Since this is a case study, we use the upper bound defined in the proof of Theorem 3 in Doerr et al. (2011)

$$n\left(1-\frac{1}{n}\right)^{-n}\int_1^n \frac{e^{-x/n}}{x}\,\mathrm{d}x = n\left(1-\frac{1}{n}\right)^{-n}\left(-\gamma + \ln n + \frac{e-1}{n} - \int_1^n \frac{e^{-x}}{x}\,\mathrm{d}x\right).$$

Note that $\left(1-\frac{1}{n}\right)^{-n} = \left(1+\frac{1}{n-1}\right)^{n} \leq e\left(1+\frac{1}{n-1}\right)$ and $\gamma \approx 0.577$ being the Euler-Mascheroni constant. Thus, by introducing this back to Equation (2.2), and by Theorem 2.33, we have

$$\begin{aligned}
\mathrm{E}[T] &\leq n\left(1-\frac{1}{n}\right)^{1-n} + ne\left(1+\frac{1}{n-1}\right)\left(-\gamma+\ln n+\frac{e-1}{n}-\int_1^n \frac{e^{-x}}{x}\,\mathrm{d}x\right) \\
&\leq n+ne\left(-\gamma+\ln n+\frac{e-1}{n}\right) \\
&= en\ln n-n(e\gamma-1)+e(e-1). \qquad \square
\end{aligned}$$

2.3.8.4 Negative Drift

If the expected drift is negative in some interval, then the algorithm is moving in expectation away from the optimum rather than towards it. Instead of deriving upper bounds to show the efficiency of an algorithm, the *negative drift* shows the inefficiency of algorithms by providing an exponential lower bound. However, this is not enough to prove exponential lower bounds on the runtime. To this end, also the probability that the process may perform large jumps towards the optimum should be proven to be low. Various drift theorems for the obtainment of exponential lower bounds have been devised (i. e., Giel and Wegener, 2003; He and Yao, 2001).

Compared to previous versions, the following theorem with simplified drift conditions was presented by Oliveto and Witt (2011, 2012). There are two simple requirements for an algorithm to be inefficient: the drift must be smaller than a negative constant and the probability of large jumps must be low.

Theorem 2.35 (Negative Drift, Oliveto and Witt, 2011, 2012)**.** *Consider a Markov process $X_0, X_1, \ldots$ on $\{0,\ldots,m\}$ with transition probabilities $p_{i,j}$ and suppose there exist integers a, b with $0 < a < b \leq m$ and $\varepsilon > 0$ such that for all $a \leq k \leq b$ the drift towards 0 is*

$$\mathrm{E}[k - X_{t+1} \mid X_t = k] < -\varepsilon \tag{2.3}$$

Further assume there exist constants $r, \delta > 0$ (i. e., they are independent of m) such that for all $k \geq 1$ and all $d \geq 1$, the probability of moving forward and backwards a distance d from k is

$$p_{k,k\pm d} \leq \frac{r}{(1+\delta)^d}, \tag{2.4}$$

where the notation "$p_{k,k\pm d} \leq x$" is a shorthand for "$p_{k,k+d} \leq x$ and $p_{k,k-d} \leq x$". Let T be the first hitting time of a state at most a, starting from $X_0 \geq b$. Let $\ell = b - a$. Then there is a

constant $c > 0$ *such that*

$$\text{Prob}\left(T \leq 2^{c\ell/r}\right) = 2^{-\Omega(\ell/r)}.$$

Theorem 2.35 is usually sufficient to prove results for the (1+1) EA. A more general version of this simplified drift theorem tailored towards the analysis of population-based EAs was introduced and applied to the analysis of an EA with fitness-proportional selection for parent selection (Neumann et al., 2009).

The negative drift theorem is used in Section 4.1 for the (μ+1) EA with a niching method called probabilistic crowding (which is basically fitness-proportional selection). In that section it is proven that this (μ+1) EA variant is not able to find the 1^n bitstring on ONEMAX and no optimum on TWOMAX for any population size μ w. o. p. in polynomial time.

2.3.9 Experimental Supplements

The main focus of this thesis is to try to close the gap between empirical and theoretical fields. We perform rigorous theoretical analyses accompanied by experimental supplements. Both fields use artificially designed functions to highlight characteristics of the studied EAs when tackling optimisation problems. The goal is to develop new ideas for the design of new variants of EAs and other search heuristics (Jansen, 2013).

Most of the analyses and comparisons made between diversity-preserving mechanisms are assessed by means of empirical investigations using complex benchmark functions and algorithmic frameworks (Sareni and Krahenbuhl, 1998; Singh and Deb, 2006). Experimental investigations face two major difficulties, *nondeterminism* and *arbitrariness of environmental conditions* (Preuss, 2015). Nondeterminism due to the stochastic nature of the algorithm can lead to varying results in quality and required effort to achieve some task. This complicates the measure and evaluation of the algorithm performance. In this sense, the definition of an outcome (success, failure or stagnation) is non-trivial, and it may result in a difficult task if the experimenter is not familiar or does not have enough experience with the algorithm.

In the case of the arbitrariness of environmental conditions, this is related to the parameters to adjust, the test problems used to test the algorithm, and the performance metric used to evaluate the algorithm. A experimental setting can only analyse a tiny subset of possible setups; in this scenario the practitioner is forced to make many restrictive decisions before the first results are obtained. Again, the definition of all these factors reside in the experience and knowledge of the experimenter.

Most of the theoretical analysis is made on simpler example functions with clear and concrete structure so that they are easy to understand. And yet they are defined in a formal way and allow for the derivation of theorems and proofs allowing us to develop our knowledge

about EAs in a sound scientific way. There are examples where empirical investigations are used to support theoretical runtime analyses (Covantes Osuna et al., 2018; Covantes Osuna and Sudholt, 2017, 2018b,c; Friedrich et al., 2009; Oliveto et al., 2014). These experiments can be used to provide information for problem dimensions that may not be covered by the theoretical results. Also, experiments can reveal insight into the constants factors and lower-order terms hidden in the asymptotic notation. In particular, experiments can provide hints whether the asymptotic runtime bounds are tight.

Both approaches are important to understand how these mechanisms impact the EA runtime and if they enhance the search for obtaining good individuals. These different expectations imply where EAs and which diversity-preserving mechanism should be used and, perhaps even more important, where they should not be used. We use the experiments for illustrative purposes. As argued in Sudholt (2008), a combination of theoretical results and experiments can deliver a more complete picture. Moreover, if visualized properly, experimental data is readable and easy to grasp. Another benefit of experiments is that the scope of a scientific work is broadened and a wider audience is addressed.

In this thesis all experiments have been conducted by the author using the Free Evolutionary Algorithm Kit (FrEAK) (Briest et al., 2004b), developed by Briest et al. (2004a). FrEAK is a free toolkit for the creation, simulation, and analysis of EAs and other randomised search heuristics within a graphical interface designed in the programming language Java, and released under the GNU General Public License (GPL). It has been designed such that it is easily extendible to incorporate phenotype search spaces, fitness functions, genotype-mappers, stopping criteria, populations models, operators like mutation, recombination, selection, and to allow the design of algorithms as if they were graphs (for more information related to the process of designing new modules, we refer the reader to Briest et al., 2004b, the Module Developer's Guide).

Moreover, the graphical interface allows the creation of *observers*; these modules can be used to show different performance measures, the individuals of the current population or other data derived from the dynamic behaviour of the algorithm. For the experiments in this thesis we have used version 0.2 of FrEAK. We have also used and developed some additional modules that have not been published.

Finally, we would like to mention that in this thesis we make use of different ways to represent the results but always trying to answer the same question: how good the algorithm works when diversity is introduced? In cases where the algorithm analysed uses several parameters, we opted to use tables instead of graphs or plots. We think that a simple, well-structured table makes it easier to track the different results for all parameters analysed than plotting several graphs with all the parameters distributed in the different axis, with several

lines overlapping each other (more difficult to see than the exact data). In some cases we have applied box plots for representing some of the statistical data (see Appendix, Section A.1) in order to observe the variance and the distribution of a population at the end of a run. We also apply the non-parametric *Mann-Whitney test* (Mann and Whitney, 1947) to test the hypothesis whether two sets of samples originate from the same probability distribution and to test whether one algorithm outperforms another one.

Overall, we have opted to sacrifice a little bit consistency for readability and clarity by representing in different ways the experimental results (when needed) in order to make it easier to appreciate the most important features of each diversity-preserving mechanism.

Chapter 3

Population Diversity in Evolutionary Algorithms

Although there is no guarantee of finding the optimal solution, approaches based on the influence of biology and life sciences such as EA, neural networks, swarm intelligence algorithms, artificial immune systems, and many others have shown to be highly practical and have provided state-of-the-art solutions to various optimisation problems (Chiong, 2009).

The advantages of nature-inspired algorithms are widely reported in literature (cf. Boussaïd et al., 2013; Chiong, 2009; Floreano and Mattiussi, 2008; Gendreau and Potvin, 2010; Rai and Tyagi, 2013; Talbi, 2009; Yang, 2010) but there has been a lack of a theoretical foundation. The assessment of nature-inspired algorithms is commonly based on experimental comparisons or problem solving oriented or by means of theoretical runtime analyses. In order to exploit nature-inspired algorithms (and any other optimisation algorithm in general) to their full potential, different statistical tests have to be carried out from experimental data to ensure a fair and meaningful comparisons between these algorithms and next to these analyses, provide theoretical foundation in order to understand the fundamental characteristics in which these nature-inspired algorithms work.

It is important to recognise that the number of parameters has a direct effect on its complexity, which complicates analysis (e. g., in the case of EAs, number of generations, iterations, population/offspring size and crossover/mutation probabilities and the dimension of the problem). Therefore, scalability for high-dimensional problems becomes an essential requirement for sophisticated optimisation approaches. Often such problems are so difficult that they can only be solved by introducing some mechanism able to visit many and/or different unexplored regions of the search space, and generate solutions that differ in various significant ways from those seen before (Gendreau and Potvin, 2010; Glover and Laguna, 1997; Lozano and García-Martínez, 2010).

There are two forces that largely determine the behaviour of an EA, *intensification* and *diversification*. EAs should be designed with the aim of effectively exploring the search space (diversification or also commonly known as exploration). The search performed by an EA should be "clever" enough to intensively explore areas in the search space with high quality solutions (intensification or also commonly known as exploitation), and to move to unexplored areas of the search space when necessary. In the following, we mention some literature reviews of several high level descriptions of intensification and diversification approaches: Battiti (1996); Glover and Laguna (1997); Mitchell (1998); Stützle (1999); Yagiura and Ibaraki (2001).

In general, providing an adequate balance between intensification and diversification becomes a very complicated task (Talbi and Bachelet, 2006). The main difference between the existing optimisation algorithms concerns the particular way in which they try to achieve this balance (Birattari et al., 2001). In fact, although most algorithms attempt to achieve this objective in their own way, it turns out that some of them show certain specialisation in intensification or diversification.

Using the clasification from Blum and Roli (2003), the intensification and diversification components occurring in optimisation algorithms can be divided in *basic* (or intrinsic) and *strategic*. The basic components are the ones defined by the basic ideas of an algorithm (mutation, crossover and selection in the case of EAs). The strategic components are composed of techniques and strategies added to the plain algorithm by the designer to improve its performance. Many of these strategies were originally developed in the context of a specific algorithm but with some adaptations or small changes added to other algorithmic approaches. Aside from the already mentioned classification made by Blum and Roli (2003), Liu et al. (2009) classified EAs into uniprocess- and multiprocess-driven approaches regarding how the balance between exploration and exploitation is achieved.

In Črepinšek et al. (2013) and Squillero and Tonda (2016) we can find a detailed review of both basic and strategic components in the family of EAs who seek to provide intensification and diversification. On the one hand, beneficial diversification properties are inherent to EAs, because they manage populations of solutions, providing a natural and intrinsic way for exploring the search space. Even more, many techniques were presented in the literature that favour diversity in population-based EAs with the aim of consolidating diversification associated with these algorithms (cf. Alba and Dorronsoro, 2005; Chaiyaratana et al., 2007; Goldberg, 1989; Koumousis and Katsaras, 2006; Lozano et al., 2005). On the other hand, some components of EAs may be specifically designed and their strategy parameters tuned, in order to provide an effective refinement. In fact, several EAs specialising in intensification

have been presented with this aim (cf. Kazarlis et al., 2001; Lozano et al., 2004; Noman and Iba, 2008).

3.1 A Review of Diversity Mechanism on Evolutionary Algorithms

In this thesis we look for runtime analyses of theoretical results on diversification components or diversity-preserving mechanisms for population-based EAs as tools to increase diversification (exploration) instead of intensification (exploitation). Although the relevance of these two concepts is commonly agreed, so far there is no unifying description to be found in the literature. In the case of EAs, diversification can be achieved by means of variation operators and intensification can be achieved by means of selection mechanisms. However, the selection mechanisms can be used to control the level of diversification and intensification, i. e., higher selection pressure can be used to orient the search towards intensification and lower selection pressure can be used to orient the search towards diversification.

In this thesis we will review methods that in some way try to achieve a balance between diversification and intensification by applying some diversity mechanism. We would like to observe whether these EAs can be potentiated or enhanced with diversity techniques. We will review, without going into too much detail, how different mechanisms have been applied to find a global optimum, to find a diverse set of optima, or, specifically in the context of multi-objective optimisation, finding Pareto optimal set of solutions (or an approximation of the Pareto front).

The main goal is to review some methods used to avoid *premature convergence* in the population-based EAs. There is no point of having a population consisting of copies of the same individual, or similar individuals. In those cases diversity mechanisms can be helpful to break these clusters of similar individuals and to promote the search to unexplored regions of the search space.

Aside from exploring the search space, a diverse population can be useful to improve the performance of the EA by improving other operators like crossover. There are better chances to create a new offspring by using different parents. Also, since the EAs work with a population of potential solutions, a diverse population can be presented to a decision maker in multi-objective optimisation where is necessary to generate solutions that often require trade-offs between different objectives. And finally in dynamic optimisation where the optimum moves over time, the diversity mechanisms can help to track new optima, and move from optima to optima without getting stuck in previous ones.

As mentioned before, many solutions have been proposed to maintain and/or promote diversity in an EA, some of the mechanisms include techniques such as eliminating duplicates, subpopulations with migration as island models, cellular EAs, and niching techniques that try to establish niches (groups or subpopulations of similar search points), and preventing niches from going extinct. In Table 3.1 we mention some mechanisms/approaches and general ideas of the techniques available to promote diversity. We also use this table to introduce the reader to the main mechanisms studied in this thesis, population-based mechanisms (non-niching) and the fitness and replacement-based mechanisms (niching). This table is a short version of Table 1 in Section 4 of Črepinšek et al. (2013). In Črepinšek et al. (2013) a more detailed description of the methods on each category is presented with a more detailed list of published work in the field of diversity mechanisms on EAs.

Table 3.1: Classification of diversity mechanisms for balancing between diversification and intensification (Črepinšek et al., 2013).

Maintenance (Non-niching)	
Population-based	Varying population size, duplicate elimination, infusion techniques (e. g., random immigrants, restarts, extinction), external archives, or migration between subpopulations.
Selection Based	Changing selection pressure or replacement restrictions.
Crossover/Mutation-based	Mating restrictions or disruptive operators.
Hybrid	Ensembles or other specific approaches.
Maintenance (Niching)	
Fitness-based	Fitness sharing in its explicit form (Goldberg and Richardson, 1987) and its implicit form (Smith et al., 1993), clearing (Pétrowski, 1996), modified clearing (Singh and Deb, 2006), clustering (Yin and Germay, 1993).
Replacement-based	Deterministic crowding (Mahfoud, 1995), probabilistic crowding (Mengsheol and Goldberg, 1999), restricted tournament selection (Harik, 1995).
Preservation-based	Species conserving GA (Li et al., 2002, 2003).
Hybrid	Ensemble of niching algorithms (Yu and Suganthan, 2010), adaptive elitist-population based GA (Liang and Leung, 2011).

Continued on next page

Table 3.1 – *Continued from previous page*

Controlling	
Through selection	Survival probability can be computed based on population diversity, or diversity can be included within those fitness functions that further drive the selection process.
Through crossover and mutation	Increase/decrease the probability of crossover and/or mutation after population diversity, fitness, and/or fitness improvements are computed.
Through changing population	After measuring population diversity, either the population size or the population alone is changed.
Learning	
Cultural learning	In combination with different machine learning techniques to learn (un)explored search areas, individuals are able to pass their knowledge to individuals in subsequent generations.
Self-organising maps	Takes into consideration not only the diversity of a current population, but also across the whole evolution process. This information is introduced into the fitness function and used to discover unexplored regions of the search space.
Binary space-partitioning tree	Avoiding previous discovered solutions, previous solutions are recorded and in the case of revisiting, random mutation within this subspace can be used to find the nearest unvisited subspace.
Estimation of distribution	Explore the search space by using machine learning techniques to learn about locations within the more promising regions.
Other Direct Approaches	
Using different subpopulations for diversification and intensification	Subpopulations are used exclusively for a particular task, e. g., one subpopulation is focused on diversification and other subpopulation is focused on intensification.
Triggers cause alternation between diversification and intensification	The use of triggers to cause alternation between diversification and intensification within the population.

Continued on next page

Table 3.1 – *Continued from previous page*

Ancestry-based	By using an *ancestry tree-based approach* for explicitly measuring diversification and intensification. If an offspring is generated with a distance bigger than a defined threshold from the parent, a new tree is formed with the offspring as a root (intensification). The process is repeated using the offspring as a root to generate new trees (diversification).

Many of the niching techniques work by modifying the selection process of individuals, taking into account not only the value of the fitness function but also the distribution of individuals in the space of genotypes or phenotypes. On the genotype level, the mechanism can try to create a diverse set of bitstrings, or on a phenotypic level, it can obtain different phenotypes by taking into consideration some form of mapping from genotypes to phenotypes. For instance, for functions of unitation (see Definition 2.12), the genotype is a bitstring, but the phenotype is given by the number of 1-bits.

Given the plethora of mechanisms to be applied, it is often not clear what the best strategy is. Which diversity mechanisms work well for a given problem, which don't, and, most importantly, why? In particular, the effect of such mechanisms have on search dynamics and performance are often not well understood. For more extensive surveys on diversity-preserving mechanisms and/or different taxonomies, we refer the reader to the previous cited works and to the surveys by Shir (2012), Črepinšek et al. (2013), Glibovets and Gulayeva (2013), and Squillero and Tonda (2016).

Most analyses and comparisons made between diversity-preserving mechanisms are assessed by means of empirical investigations (Chaiyaratana et al., 2007; Sareni and Krahenbuhl, 1998; Singh and Deb, 2006; Ursem, 2002) using real-parameter multimodal optimisation problems (Das et al., 2011; Li et al., 2013) without providing a comprehensive theory of the population dynamics and the diversity achieved. The above includes most of the methods published.

In the case of theoretical runtime analyses (Doerr et al., 2016; Friedrich et al., 2007; Gao and Neumann, 2014; Jansen and Wegener, 2005; Oliveto and Sudholt, 2014; Oliveto et al., 2014), the analysis is made using simpler example functions easier to analyse compared to the ones used in empirical investigations. These simplified functions are helpful when analysing specific characteristics of an algorithm and particular problems, e. g., the time

required to escape from a local optimum, the time required of a certain subpopulation to takeover other subpopulation, or the time needed to find one or several optima.

In theoretical runtime analysis the results help to get insight into the search behaviour of EAs in the presence or absence of diversity, and how parameters and explicit diversity mechanisms affect the performance. They in particular highlight which diversity mechanisms are effective for particular problems, and which are ineffective. More importantly, they explain why diversity mechanisms are effective or ineffective, and how to design the most effective evolutionary algorithms for the problems considered.

Let us note that the goal of this chapter is not to provide a comprehensive survey of the work done in both fields. The main goal is to provide a general review of the work done in the area of analysis of diversity mechanisms. At the same time, we do not look to replicate the formal proofs of the cited theoretical works. Instead, we combine formal theorems with informal explanations for the proofs by just explaining key ideas. In the same note, we will compare the results obtained from experimental work with results from the theoretical field when applicable. Finally, we only present selected results from the papers, or results that are simplified towards special cases for reasons of simplicity. We refer the reader to the original papers for further details.

We would like to conclude this introduction by mentioning that a similar work to this chapter has been done by Sudholt (2018). Sudholt reviews runtime analyses that have shown benefits of population diversity, this includes explicit diversity mechanisms embedded into a EA or through intrinsic EA methods. His survey reviews part of the results from the present thesis. Results from Section 4.3 and Chapter 6 are summarised in Sections 3.1.6 and 6 from Sudholt (2018), respectively. This thesis also contains material that was not available at the time the survey was written. We present theoretical results of two more niching mechanisms: probabilistic crowding (Section 4.1) and restricted tournament selection (Section 4.2). We also present an extensive empirical analysis of several diversity mechanisms in Chapter 5. Finally, we expand Sudholt's survey by presenting a review of the work done with a mechanism from the artificial immune systems field called *ageing*, and the work done in multi-objective optimisation that was not included in that survey.

Finally, since this chapter is based on the same topics as Sudholt's survey, this chapter adopts some of the structure in Sections 3.2, 3.3, and 3.4 by reviewing the same work independently. And as mentioned before, this chapter expands Sudholt's survey by providing a review of ageing mechanisms (Sections 3.2.6 and 3.3.1.2) and multi-objective optimisation (Section 3.5). Another difference of this chapter compared to Sudholt's survey is that we will focus on panmictic populations (where all individuals are potential partners like in

the (μ+1) EA), i. e., we will mention some general ideas from the results obtained using subpopulations on island models but without going into too much details.

3.2 Diversity Mechanisms for the (μ+1) EA

In the following sections we review runtime analyses on EAs with explicit diversity mechanisms for the TWOMAX and BALANCE function. We present these two functions together in this section to show that some mechanisms that are able to optimise TWOMAX are not able to optimise BALANCE and vice versa. We will make clear and highlight these observations in each of the following sections. This section is inspired by Sections 3.1 and 5.1 in Sudholts' survey, we further expand it by adding a review of the ageing mechanisms in Section 3.2.6.

Let us start with the TWOMAX function. As defined earlier in Section 2.1.1.2, TWOMAX is an ideal benchmark function for studying diversity mechanisms as it is simply structured, hence facilitating a theoretical analysis, and it is hard for EAs to find both optima as they have the maximum possible Hamming distance. Some combinatorial optimisation problems share the same structure of TWOMAX, e. g., the bipartite graph VERTEXCOVER analysed in Oliveto et al. (2008) which consists of two branches with one local optimum and the other to the minimum cover and the MINCUT instance analysed in Sudholt (2011a). The first results using diversity mechanisms on this function were obtained by Friedrich et al. (2009) and Oliveto et al. (2014).

For all results on TWOMAX, we choose to present the results from the survey done by Sudholt (2018) instead of the original work (see Friedrich et al., 2009). The reason for this is that in this thesis we take into consideration TWOMAX with two global optima instead of TWOMAX with one global optima (like in Friedrich et al., 2009). The results from Sudholt (2018) are adapted to take into consideration our definition of TWOMAX and are more suitable for comparison with our results presented in Chapter 4.

The BALANCE function (see Definition 3.1) was first introduced by Rohlfshagen et al. (2009). Following the description provided by Oliveto and Sudholt (2014), this function consists of a prefix of length $n/2$ and a suffix of the same length. The fitness of a search point is determined by the number of leading ones in the prefix and the number of 1-bits (i. e., ONEMAX) in the suffix. A globally optimal search point has the maximal value $n/2$ of leading ones (i. e., $\mathrm{LO}(a) = n/2$) in the prefix. However, before reaching the global optimum the algorithm may reach the two trap regions corresponding to search points with less that $n/16$ 0-bits or less than $n/16$ 1-bits in the suffix. The trap regions and the global optimum are separated by a region of 0-fitness of length $\sqrt{n}$ that makes it prohibitive for search heuristics to reach the optimum from the traps.

Definition 3.1 (BALANCE, Rohlfshagen et al., 2009)**.** *Let* $a,b \in \{0,1\}^{n/2}$ *and* $x = ab \in \{0,1\}^n$*. Then*

$$\text{BALANCE}(x) := \begin{cases} n^3 & \text{if } \text{LO}(a) = n/2, \text{ else} \\ |b|_1 + n \cdot \text{LO}(a) & \text{if } n/16 < |b|_1 < 7n/16, \text{ else} \\ n^2 \cdot \text{LO}(a) & \text{if } |a|_0 > \sqrt{n}, \text{ else} \\ 0 & \text{otherwise.} \end{cases}$$

Where $|x|_1 = \sum_{i=1}^{n/2} x_i$*,* $|x|_0 = n/2 - |x|_1$ *denotes the number of ones and the number of zeroes in individual x, respectively.* $\text{LO}(x) := \sum_{i=1}^{n/2} \prod_{j=1}^{i} x_j$ *counts the number of leading ones. A sketch of the* BALANCE *function is given in Figure 3.1.*

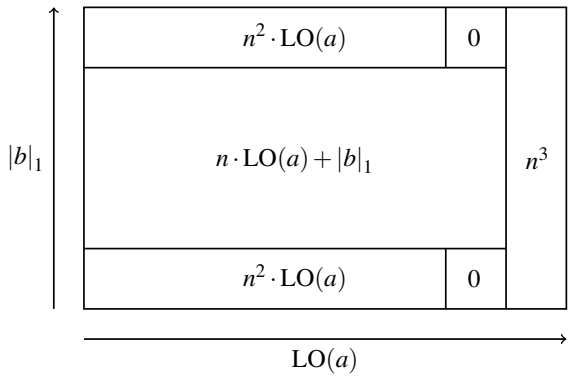


Figure 3.1: Sketch of the function BALANCE (Rohlfshagen et al., 2009).

BALANCE is used in Dynamic Optimisation Problems (DOPs), and the field concerned with the application of EAs to these classes of problems is called Evolutionary Dynamic Optimisation (EDO). In DOPs, the optimisation algorithm not only has to locate the optimum of a given problem, but also to track the optimal solution over time when the problem changes every τ generations. In BALANCE, every τ generations, the role of zeros and ones in the suffix is reversed, so that the fitness gradient switches between maximising and minimising the number of ones in the ONEMAX part.

Diversity can be used to avoid getting trapped in local optima independent of the frequency of change and to keep track of global optima, to explore and re-discover new global optima in case the previous global optimum becomes a local optima due to the change of fitness, and another local optimum becomes the new global optimum. Here we mention some runtime analyses done on the area of EDO (e. g., Dang et al., 2017b; Jansen and Zarges, 2014; Kötzing et al., 2015; Kötzing and Molter, 2012; Lissovoi and Witt, 2015; Oliveto and Zarges, 2015;

Rohlfshagen et al., 2009). In the following we describe the work done in the area of EDO with diversity mechanisms on the BALANCE function.

In order to guarantee a fair comparison on TWOMAX and BALANCE, the plain (μ+1) EA is considered, then diversity mechanisms are added to the plain (μ+1) EA. The diversity mechanisms considered for both functions include: avoiding genotype and fitness duplicates, deterministic crowding, fitness sharing (in two variants) and finally, a combination of fitness sharing and deterministic crowding is analysed in the context of (μ+1) RLS. The main goal is to determine whether populations equipped with diversity mechanisms can be robust enough to optimise the TWOMAX function on an static setting, and BALANCE on a dynamic setting.

In Section 3.2.6 we review the work done with the ageing mechanism for other test functions. Since the work done for ageing consists of several ways to apply this mechanism we will highlight some of them by providing high-level ideas of the results obtained with ageing compared to its plain version without going into too much details for each variation of ageing.

3.2.1 Plain (μ+1) EA

The plain (μ+1) EA (Algorithm 5) has already been investigated by Witt (2006) for pseudo-Boolean functions, and in (Friedrich et al., 2009, Theorem 1) for the case of TWOMAX and in Oliveto and Zarges (2015) for the case of BALANCE. With respect of TWOMAX, the selection pressure is quite low, nevertheless, the (μ+1) EA is not able to maintain individuals on both branches for a long time; the whole population is likely to converge to one of the two peaks (Theorem 3.2).

Theorem 3.2 (Theorem 1 in Sudholt, 2018)**.** *The probability that the* (μ+1) EA *with no diversity-preserving mechanisms and* $\mu = o(n/\log n)$ *finds both optima of* TWOMAX *in time* n^{n-1} *is* $o(1)$. *The expected time for finding both optima is* $\Omega(n^n)$.

The idea behind the proof of Theorem 3.2 is that, let us assume that 0^n has been found, two possible events may happen: create a clone of 0^n or find 1^n. The first event can occur if 0^n is selected as parent and no bit is flipped during mutation, i. e., a clone of 0^n enters the population. The more clones of 0^n are contained in the population, the larger the probability of this event becomes. And to create 1^n, it is necessary to flip the remaining 0-bits in an individual and no 1-bits are flipped. If the population is small, clones of 0^n will tend to takeover the whole population before the 1^n optimum is found. Once the whole population collapses to the 0^n individuals, it will be necessary to flip all the bits from one of 0^n individuals at the same time (which has probability n^{-n}) in order to create the 1^n individual, and find

both optima on TWOMAX. And even considering n^{n-1} generations, the probability of this jump converges to 0, that is $o(1)$. So the expected optimisation time is $\Omega(n^n)$.

For BALANCE, the (1+1) EA has already been analysed by Rohlfshagen et al. (2009). They show that the expected optimisation time is exponential if the frequency of change is low. Oliveto and Zarges (2015) analysed the (μ+1) EA and show that the whole population gets stuck in one of the traps before any individual reaches the global optimum.

Theorem 3.3 (Theorem 1 in Oliveto and Zarges, 2015)**.** *If $\tau > 20\mu n$ and $\mu \leq n^{1/2-\varepsilon}$ then the expected time for the* (μ+1) EA *to optimise* BALANCE *is at least $n^{\Omega(\sqrt{n})}$. If $\tau > 38\mu n^{3/2}$ and $\mu \leq n^{1/2-\varepsilon}$ then the* (μ+1) EA *requires at least $n^{\Omega(\sqrt{n})}$ steps with w. o. p.*

The main reason for this bad performance is that the population reaches the trap before it optimises the leading 1-bits prefix. Since it is easier to fall into a trap by optimising the ONEMAX suffix than reach the $n/2$ leading 1-bits in the prefix. With low frequencies of change, this is very likely to happen.

3.2.2 No Genotype Duplicates

One of the simplest ways to enforce diversity maintenance on a population-based algorithm is by avoiding duplicates. In the context of the (μ+1) EA, once the population has been initialised, new individuals are not allowed to enter the population, if there is a solution with the same genotype already contained in the population. This effect can be achieved by adding the condition "and $y \notin P_t$" to the "if" statement in the Line 6 from Algorithm 5, resulting in Algorithm 8. This kind of mechanism has already been studied in Storch and Wegener (2004) and for the case of TWOMAX in (Friedrich et al., 2009, Theorem 2).

Algorithm 8 (μ+1) EA with no genotype duplicates

1: Let $t := 0$ and initialise P_0 with μ individuals chosen uniformly at random.
2: **while** stopping criterion **not** met **do**
3: Choose $x \in P_t$ uniformly at random.
4: Create y by flipping each bit in x independently with probability $1/n$.
5: Choose $z \in P_t$ with worst fitness uniformly at random.
6: **if** $f(y) \geq f(z)$ **and** $y \notin P_t$ **then** $P_{t+1} = P_t \setminus \{z\} \cup \{y\}$ **else** $P_{t+1} = P_t$ **end if**
7: Let $t := t + 1$.
8: **end while**

This mechanism ensures that the population always contains μ different genotypes (modulo possible duplicates during initialisation). However, ensuring genotype diversity is too weak and is not able to prevent the extinction of one branch, ending with the population converging to one optimum with high probability (Theorem 3.4).

Theorem 3.4 (Theorem 2 in Sudholt, 2018)**.** *The probability that the* (μ+1) EA *with genotype diversity and* $\mu = \mathrm{o}\left(n^{1/2}\right)$ *finds optima of* TwoMax *in time* n^{n-2} *is at most* $\mathrm{o}(1)$*. The expected time for finding both optima is* $\Omega\left(n^{n-1}\right)$.

Unlike Algorithm 5, Algorithm 8 does not allow clones of the 0^n individual to enter the population. The algorithm can still create individuals similar to 0^n by flipping just one 0-bit to 1, in this sense the population can have one 0^n individual, and since the mutation can flip any of n-bits, the population can contain many search points with only a single 1-bit that are at least as fit as the current best search point. If the population size is $\mu = \mathrm{o}\left(n^{1/2}\right)$, it is likely that individuals on the 1^n branch get extinct due to the takeover of the 0^n branch. Again the argument was based on selecting 0^n as parent, with only one individual with 0^n genotype. This leads to a more restrictive condition on $\mu = \mathrm{o}\left(n^{1/2}\right)$ compared to the setting of no diversity mechanism ($\mu = \mathrm{o}(n/\log n)$).

In the context of Balance, again the diversity mechanisms is not strong enough to promote diversity leading the whole population into the traps in the same way as the plain (μ+1) EA on the Balance function. Not allowing copies in the population does not significantly change the behaviour of the algorithm.

Theorem 3.5 (Theorem 2 in Oliveto and Zarges, 2015)**.** *For the* (μ+1) EA *with no genotype duplicates, the same results from Theorem 3.3 hold.*

3.2.3 No Fitness Duplicates

Similar to avoiding duplicates in a genotype level, diversity can be enforced by avoiding duplicates in a phenotype level. According to Algorithm 9, after initialisation, if there is an individual z in the population with the same fitness as the new individual resulting from the mutation y, y replaces z (novelty is rewarded), otherwise it proceeds as the plain (μ+1) EA, choosing the worst individual z uniformly at random, and if y is at least as good as z, y replaces z. This mechanism has already been analysed in Friedrich et al. (2007) for plateaus of constant fitness and in (Friedrich et al., 2009, Theorem 3) for TwoMax. In addition, this resembles the idea of fitness diversity proposed by Hutter and Legg (2006).

Again, this simple mechanism is not able to prevent the extinction of one branch, ending with the population converging to one optimum with high probability (Theorem 3.6). In this version of TwoMax with two global optima, the population will never be able to contain both optima, so instead of achieving a population with both optima, Sudholt (2018) considered the cases where a population and the offspring contains both optima.

Algorithm 9 (μ+1) EA with no fitness duplicates

Let $t := 0$ and initialise P_0 with μ individuals chosen uniformly at random.
while stopping criterion **not** met **do**
    Choose $x \in P_t$ uniformly at random.
    Create y by flipping each bit in x independently with probability $1/n$.
    if there exists $z \in P_t$ such that $f(y) = f(z)$ **then**
        $P_{t+1} = P_t \setminus \{z\} \cup \{y\}$
    else
        Choose $z \in P_t$ with worst fitness uniformly at random.
        if $f(y) \geq f(z)$ **then** $P_{t+1} = P_t \setminus \{z\} \cup \{y\}$ **else** $P_{t+1} = P_t$ **end if**
    end if
    Let $t := t + 1$.
end while

Theorem 3.6 (Theorem 3 in Sudholt, 2018)**.** *The probability that the* (μ+1) EA *with fitness diversity and* $\mu \leq n^{O(1)}$ *finds both optima of* TwoMax *in time* 2^{cn}*,* $c > 0$ *being an appropriate constant, is at most* $o(1)$*. The expected time for finding both optima is* $2^{\Omega(n)}$*.*

In this scenario, once an optimum has been found (let us say 0^n optimum), individuals with the same fitness are not allowed to enter the population; this means that individuals on zero branch are forced to have different fitness values covering this branch. Then, there will be a competition between new generated individuals and individuals covering the zero branch. If a new individual is generated in the opposite branch (1^n), this individual replaces the old individual with the same fitness on the opposite branch (0^n). Friedrich et al. (2009) defined a potential function that captures this competition, and showed that there is a bias that favours the branch with the already found optimum. The idea is that individuals from the undiscovered optimum have to make more difficult mutations to climb up the branch, while individuals from the discovered optima need one bit-flip to create a new individual which provokes the extinction of the individuals on the opposite branch.

In contrast to TwoMax where the runtime is exponential for any population size, no fitness duplicates is effective w. o. p. on Balance independent of the frequency of change.

Theorem 3.7 (Theorem 4 in Oliveto and Zarges, 2015)**.** *Let* $\mu > n - 2(\sqrt{n} - 1)$*. Then w. o. p. the* ($\mu$+1) EA *with no fitness duplicates optimises* Balance *in time* $O(\mu n^3)$ *for arbitrary* $\tau \geq 0$*.*

The (μ+1) EA with no fitness duplicates and sufficiently large population size is able to get individuals into both traps. Once both traps are found, and the regions of 0-fitness of length $\sqrt{n}$ that separate the traps with the global optimum are found, no other individual with

the those fitness values are accepted. Then, the remaining individuals can evolve leading to the global optimum of the BALANCE function afterwards.

3.2.4 Deterministic Crowding

In this niching mechanism the offspring competes directly with its most similar parent. According to Mahfoud (1995), in a GA, all elements of the population are grouped into $\mu/2$ pairs (assuming μ to be even). Then, these groups are recombined and mutated. For each pair of offspring, two sets of parent-child tournaments are possible. Each offspring competes against the most similar parent according to a distance metric, either genotypic or phenotypic, and the offspring replace their closest parent if it is at least as good. As the (μ+1) EA does not consider crossover, the offspring competes with its only parent. Then the population contains μ lineages that evolve independently (see Algorithm 10).

Algorithm 10 (μ+1) EA with deterministic crowding

1: Let $t := 0$ and initialise P_0 with μ individuals chosen uniformly at random.
2: **while** stopping criterion **not** met **do**
3: Choose $x \in P_t$ uniformly at random.
4: Create y by flipping each bit in x independently with probability $1/n$.
5: **if** $f(y) \geq f(x)$ **then** $P_{t+1} = P_t \setminus \{x\} \cup \{y\}$ **else** $P_{t+1} = P_t$ **end if**
6: Let $t := t + 1$.
7: **end while**

It is proven in (Friedrich et al., 2009, Theorem 4) that this mechanism with a sufficiently large population is able to reach both optima with high probability in expected time $O(\mu n \log n)$ (Theorem 3.8). Here we show the modified result from Sudholt (2018) that takes into consideration finding both optima instead of only one global optimum.

Theorem 3.8 (Theorem 4 in Sudholt, 2018)**.** *The* (μ+1) EA *with deterministic crowding and* $\mu \leq n^{O(1)}$ *reaches in* TWOMAX *a population consisting of only global optima in expected time* $O(\mu n \log n)$. *In that case the population contains both global optima with probability at least* $1 - 2^{-\mu+1}$.

Since all μ lineages evolve independently, and each global optimum 0^n and 1^n can be found after hill climbing with probability $1/2$, the probability of finding an optimum is $1 - 2^{-\mu-1}$. It is only when all individuals are initialised in one branch (which happens with probability $2^{-\mu+1}$), when it is likely that only one optimum is reached. Aside from this unlucky initialisation, individuals in both branches can evolve independently, leading to an upper time bound of $O(\mu n \log n)$.

On the contrary, the (μ+1) EA with deterministic crowding on BALANCE fails, it requires exponential time to optimise the function.

Theorem 3.9 (Theorem 3 in Oliveto and Zarges, 2015)**.** *With overwhelming probability, the* (μ+1) EA *with deterministic crowding and* $\mu \leq n^{\mathrm{O}(1)}$ *requires exponential time to optimise* BALANCE *if* $\tau > 8e\mu n$.

The population of the (μ+1) EA with deterministic crowding behaves similarly to μ parallel executions of the (1+1) EA, i. e., μ individuals explore the fitness landscape independently, and this parallelism does not prevent the population from getting stuck in one of the traps.

3.2.5 Fitness Sharing

This mechanism was one of the first attempts to deal directly with the location and preservation of multiple solutions in GAs. The fitness sharing algorithm restricts the growth of one type of individuals by sharing its real fitness assignment with nearby elements in the population (Goldberg and Richardson, 1987).

The idea is to reduce the pay-off in densely populated regions. The fitness sharing algorithm restricts the growth of one type of individuals by making each individual in the population share its fitness assignment with nearby elements on the population. If there are several copies of the same individual in the population, they share their fitness by degrading an individual's pay-off due the presence of other individuals in its neighbourhood, and the amount of sharing contributed by each individual into its neighbour depends on the proximity between the individuals.

As a consequence, selection is likely to remove such clusters and to keep the individuals apart. The similarity between individuals x and y is measured by a so-called *sharing function* $\mathrm{sh}(x,y) \in [0,1]$ where a large value corresponds to a great similarity and a 0 value implies no similarity. The shared fitness $f_{\mathrm{sh}}(x,P)$ of individual x in the population P with fitness $f(x)$ is

$$f_{\mathrm{sh}}(x,P) = \frac{f(x)}{\sum_{y \in P}^{\mu} \mathrm{sh}(x,y)} \quad \text{where} \quad \mathrm{sh}(x,y) = \begin{cases} 1 - \left(\frac{\mathrm{d}(x,y)}{\sigma}\right)^{\alpha}, & \text{if } \mathrm{d}(x,y) < \sigma; \\ 0, & \text{otherwise.} \end{cases} \tag{3.1}$$

Where α is a positive constant called *scaling factor* which regulates the shape of the sharing function, $\mathrm{d}(x,y)$ is any a distance metric, either genotypic or phenotypic, and σ is called *sharing radius*. The σ parameter defines for each individual, the maximum distance over

which it has to share its fitness with other population members, i. e., individuals only share fitness if they have a distance less than σ. In the following, we will review runtime analyses for the original approach (individual-based fitness sharing), a variant called population-based fitness sharing that uses the shared fitness of the population as a selection for survival policy; the population with highest shared fitness is chosen to become the next population on the next generation. Finally a hybrid between population-based fitness sharing and deterministic crowding is also reviewed for the case of BALANCE; by using deterministic crowding in the selection for survival step, the population-shared fitness of the current population is compared against the population-shared fitness of the resulting offspring population, then the population with the largest shared fitness is selected.

Let us note that in both analyses the scaling factor has been set to $\alpha = 1$ and, sharing distance to $\sigma = n/2$ (as this is the smallest value allowing discrimination between the two branches). Also, only phenotypic distance (see Definition 2.13) is analysed since TWOMAX only depends on the number of ones. The precise sharing function is then

$$\mathrm{sh}(x,y) = \begin{cases} 1 - 2\frac{||x|_1 - |y|_1|}{n}, & \text{if } \mathrm{d}(x,y) < \sigma; \\ 0, & \text{otherwise.} \end{cases} \tag{3.2}$$

To our knowledge no runtime analyses for fitness sharing with genotypic distance on TWOMAX and on the BALANCE functions are available.

3.2.5.1 Individual-based Fitness Sharing

The original fitness sharing, where selection is based on individuals (Algorithm 11) was studied by Oliveto et al. (2014) using the sharing function described in Equation (3.2) for the case of TWOMAX.

Algorithm 11 (μ+1) EA with fitness sharing

1: Let $t := 0$ and initialise P_0 with μ individuals chosen uniformly at random.
2: **while** stopping criterion **not** met **do**
3: Choose $x \in P_t$ uniformly at random.
4: Create y by flipping each bit in x independently with probability $1/n$.
5: Let $P_t^* = P_t \cup \{y\}$.
6: Update the fitness of all individuals in P_t^* according to the fitness sharing procedure described in Equation (3.1).
7: Choose $z \in P_t^*$ with worst fitness uniformly at random.
8: **if** $f(y) \geq f(z)$ **then** $P_{t+1} = P_t^* \setminus \{z\}$ **else** $P_{t+1} = P_t^* \setminus \{y\}$ **end if**
9: Let $t := t + 1$.
10: **end while**

The analysis in Oliveto et al. (2014) showed that a population size $\mu = 2$ is not sufficient to find both optima in polynomial time. With probability $1/2 + \Omega(1)$ the population will reach the same optimum, and from there the expected time to find both optima is $\Omega\left(n^{n/2}\right)$. However, there is still a constant probability $\Omega(1)$ to find both optima in polynomial expected time $O(n \log n)$, if the two search points are initialised on different branches, and if these two search points maintain similar fitness values throughout the run.

Theorem 3.10 (Theorem 1 and 2 in Oliveto et al., 2014)**.** *The* (2+1) EA *with fitness sharing,* $\alpha = 1$*,* $\sigma = n/2$*, and phenotypic sharing with probability* $1/2 + \Omega(1)$ *will reach a population with both members in the same optimum, and then the expected time for finding both optima from there is* $\Omega\left(n^{n/2}\right)$*. However, with probability* $\Omega(1)$ *the algorithm will find both optima in time* $O(n \log n)$*.*

With $\mu \geq 3$, once the population is close enough to one optimum, individuals descending the branch heading towards lower fitness values are accepted. This allows an individual to reach the opposite branch, and after that, the expected time required to climb up the new found branch and find the remaining optimum is $O(\mu n \log n)$.

Theorem 3.11 (Theorem 3 in Oliveto et al., 2014)**.** *For any population size* $\mu \geq 3$ *the* (μ+1) EA *with fitness sharing,* $\alpha = 1$*,* $\sigma = n/2$*, and phenotypic sharing will find both optima* TwoMax *in expected time* $O(\mu n \log n)$*.*

Concerning the effects of the offspring population, increasing the offspring population of a (μ+λ) EA, with $\mu = 2$ and $\lambda \geq \mu$ cannot guarantee convergence to populations with both optima, i. e., depending on λ one or both optima can get lost, and so the expected time for finding both optima is $\Omega\left(n^{n/2}\right)$.

Theorem 3.12 (Theorem 4 in Oliveto et al., 2014)**.** *With probability* $1 - o(1)$ *the* (2+2) EA *with fitness sharing with* $\alpha = 1$*,* $\sigma = n/2$*, and phenotypic sharing will, at some point of time, reach a population with both members in the same optimum. The expected time for finding both optima from there is* $\Omega\left(n^{n/2}\right)$*.*

3.2.5.2 Population-based Fitness Sharing

This variant was introduced and analysed for the case of TwoMax in (Friedrich et al., 2009, Theorem 5) with sharing function as described in Equation (3.2). Rather than selecting individuals based on their shared fitness, selection was done on a level of populations. The goal is to select the new population out of the union of all parents and all offspring such that

it maximises the overall shared fitness of the population. In this case the shared fitness of the population is defined as

$$f(P) = \sum_{x \in P} f_{\text{sh}}(x, P). \tag{3.3}$$

The drawback of this approach is that all possible size-μ subsets of this union of size $\mu + \lambda$ need to be examined (see Algorithm 12). For large μ and λ, is computationally expensive.

Algorithm 12 (μ+1) EA with population-based fitness sharing

1: Let $t := 0$ and initialise P_0 with μ individuals chosen uniformly at random.
2: **while** stopping criterion **not** met **do**
3: Choose $x \in P_t$ uniformly at random.
4: Create y by flipping each bit in x independently with probability $1/n$.
5: Let $P_t^* = P_t \cup \{y\}$.
6: Update the population fitness of P_t^* according to the fitness sharing procedure described in Equation (3.3).
7: Choose $z \in P_t^*$ that $f(P_t^* \setminus \{z\})$ is maximised.
8: Let $P_{t+1} = P_t^* \setminus \{z\}$.
9: Let $t := t + 1$.
10: **end while**

This population-based fitness sharing approach, constructing the best possible new population amongst parents and offspring is able to find both optima in expected time $\mathrm{O}(\mu n \log n)$ for any population size $\mu \geq 2$.

Theorem 3.13 (Theorem 8 in Sudholt, 2018)**.** *The* (μ+1) EA *with population-based fitness sharing and* $\mu \geq 2$ *reaches on* TwoMax *a population containing both optima in expected optimisation time* $\mathrm{O}(\mu n \log n)$.

Let us imagine TwoMax as a straight line, then the maximum population shared fitness can be achieved with individuals with the smallest and the largest number of ones, these two individuals have the largest distance to all individuals in the population. The method promotes the creation of individuals in opposite directions (this includes individuals with worst fitness), which translates in the creation of individuals with the minimum (0^n) and maximum (1^n) number of ones. Performing a hill-climbing towards both extremes of TwoMax yields the expected time of $\mathrm{O}(\mu n \log n)$.

3.2.5.3 Population-based Fitness Sharing and Deterministic Crowding

Finally, Oliveto and Zarges (2015) analysed a hybrid version of the population-based fitness sharing and deterministic crowding mechanisms for the case of Balance. As mentioned

before, by using deterministic crowding in the selection for survival step, the population-shared fitness of the current population is compared against the population-shared fitness of the resulting offspring population, then the population with the largest shared fitness is selected.

The specifics of the algorithm are as follows. In this variant the population size is set to $\mu = 2$, Hamming distance is used as a dissimilarity metric with parameters $\alpha = 1$ and sharing radius $\sigma = n$ (basically all individuals in the population share their fitness) for the fitness sharing method, and instead of standard bit mutations, local mutations are used to flip exactly one bit chosen uniformly at random. The resulting algorithm is referred to as (2+1) RLS (Algorithm 13).

Algorithm 13 (μ+1) RLS with population-based fitness sharing and deterministic crowding

1: Let $t := 0$ and initialise P_0 with μ individuals chosen uniformly at random.
2: **while** stopping criterion **not** met **do**
3: Choose $x \in P_t$ uniformly at random.
4: Create y by flipping a randomly chosen bit in x.
5: Let $P_t^* = P_t \setminus \{x\} \cup \{y\}$.
6: **if** $f(P_t^*) \geq f(P_t)$ **then** $P_{t+1} = P_t^*$ **else** $P_{t+1} = P_t$ **end if**
7: Let $t := t + 1$.
8: **end while**

Theorem 3.14 (Theorem 5 in Oliveto and Zarges, 2015)**.** *With probability at least* $1/2 - e^{-\Omega(n)}$ *the* (2+1) RLS *with population-based fitness sharing and deterministic crowding finds the optimum of* BALANCE *in time* $\mathrm{O}(n^2)$ *for arbitrary* $\tau \geq 0$.

The proof idea is that, with probability exponentially close to $1/2$ the two individuals will never reach the traps, and if the traps are not reached, the optimum can be found in time $\mathrm{O}(n^2)$ by optimising the leading ones part in the prefix. If the sum of leading 1-bits in x and y is greater or equal than 2, then only bit-flips in the suffix increasing the Hamming distance are accepted ($f(x,y) + \mathrm{H}(x,y) > 2n$). If $\mathrm{H}(x,y) + f(x) + f(y) < 2n$, such bit flips are accepted if and only if they increase the fitness.

Now, with probability at least $1/2$, the (2+1) RLS will always have a fitness larger than $2n$ with individuals x and y having a total of at least 2 leading ones. Again, by the fitness sharing method, only bit flips on the ONEMAX section will be accepted if the Hamming distance increases. This is helpful since individuals will never meet the extreme values corresponding to a trap. The leading ones part will be optimised as it has a much larger impact on the fitness, leading to a global optimum in the claimed time. However, if τ is set to small, the algorithm will never optimise BALANCE with constant probability.

Theorem 3.15 (Theorem 6 in Oliveto and Zarges, 2015)**.** *Let* $\tau > 12n+1$. *With probability bounded below by a constant the algorithm* (2+1) RLS *with fitness sharing and crowding requires infinite time to optimise* BALANCE.

The general idea is that the algorithm starts by maximising the number of ones in the ONEMAX section, with fitness $f(x,y)+\mathrm{H}(x,y) < 2n$ leading to one of the search points to the upper trap. Once the population has reached this trap, the population will be trapped with a fitness larger than $2n$ and will start to maximise the Hamming distance which will drive the other individual the other trap. Finally once both individuals are trapped, local mutations are not able to create the global optimum from a trap and in consequence there is no way to escape from both traps.

3.2.6 Ageing

In most of the generational approaches, the parent population is replaced by the offspring population by applying survival selection. In its basic form, *ageing* can be seen as a softened version of the generational approach; individuals with a certain age are discarded after they have reached a maximal age τ to promote diversity. Ageing operators are applied in the field of artificial immune systems (AIS) to increase the diversity of the population during the optimisation process. These AIS are a class of general purpose algorithms based on the immune system of vertebrates (de Castro and Timmis, 2002).

Apart from the absence of crossover operators, the main distinguishing features of AIS compared to EAs are the hypermutation operators, and the presence of ageing operators to enhance diversity in the population. In the context of ES, one of the first proposals where the concept of ageing is introduced was in Schwefel and Rudolph (1995). Furthermore, in the context of GAs, Ghosh et al. (1996) developed an ageing genetic algorithm (aGA), where age for an individual is included into the fitness function as well as into mating. As in nature, adult individuals are considered more fit for mating. On a given problem, Ghosh et al. (1996) reported that aGA maintains more diversity in the population, while its performance has also been improved.

Kubota and Fukuda (1997) proposed two GAs with age structure (ASGA) where individuals are removed from the population when they have reached a lethal age regardless of the fitness value. They mentioned that these two ASGA have three major advantages over standard GAs: (1) prevent individuals with high fitness from taking over the population, (2) control selection pressure by the ageing process and maintain relatively high genetic diversity in a population and, (3) solve optimisation problems with a small population size.

There are several runtime analyses of ageing mechanisms introduced in EAs (Horoba et al., 2009; Jansen and Zarges, 2009, 2010a,b, 2011a,b; Oliveto and Sudholt, 2014). By introducing the ageing mechanism into the standard (μ+1) EA, we can obtain Algorithm 14. Essentially, a population of μ individuals of age 0 is initialised uniformly at random. In each generation each individual's age is increased by 1, one individual is selected for reproduction and one offspring is created due to mutation. Then, the age of the new individual is set by introducing some ageing operator. Once the age of the new individual has been set, individuals are removed from the population according to their age. These age removal operators are used in Line 5 from Algorithm 14. In AIS two main ageing mechanisms are common: *static pure ageing*, individuals are removed once they have reached a certain threshold (Cutello et al., 2007b) and *stochastic pure ageing*, individuals are removed with some probability (Cutello et al., 2003, 2007a), then taking advantages from both operators, Oliveto and Sudholt (2014) designed a third operator called *hybrid pure ageing*, individuals that have reached a certain threshold are removed with some probability.

The most common selection for replacement is as follows. If the population size is smaller than μ, then the population gets filled with individuals created uniformly at random (*birth phase*) to ensure that the population size remains throughout the run. If the population remains size μ, i. e., no individual has been removed due to the ageing operator, the replacement proceeds like the plain (μ+1) EA (an individual z with worst fitness is selected uniformly at random and removed from the population). The idea behind all these ageing approaches is to increase the diversity of the population by removing old individuals that have been in the population for a long time (Oliveto and Sudholt, 2014).

Algorithm 14 Algorithmic Framework of the (μ+1) EA with Ageing

1: **Initialisation:**
Initialise population P of size μ.
2: **Ageing, Growing older:**
Increase age of all search points in P.
3: **Variation:**
Generate new search point z.
4: **Ageing, Age of new search points:**
Decide about the age of the new search point z.
5: **Ageing, Removal due to age:**
Remove individuals with age exceeding τ.
6: **Selection for replacement:**
Decide if z is to be inserted in P. Remove or add search points as needed.
7: **Stopping:**
If stopping criterion **not** met continue at Line 2.

As can be seen from Algorithm 14, there are 4 sections where ageing plays a crucial role (Lines 2, 4, 5 and 6) and in each section the age of an individual can be used to perform each operation. Due to the high modularity of the EAs, many methods can be added in each line of Algorithm 14. For example, if the offspring is created as a result of two parents by using variation operators in Line 3, Jansen and Zarges (2010a) defined an algorithm to use crossover, mutation, and static pure ageing. And now that an offspring is created by two parents, static pure ageing is modified to take into consideration 2 parents.

Jansen and Zarges (2010b) took the previous idea, and made a step further. Due to crossover, each new search point has two parents as origin, so what age has to be inherited if no improvement was made? This question was addressed by providing three variants; the age of the offspring gets the age of the older parent, the age of the offspring gets the age of the best parent according to the fitness function value, or setting the age of the offspring with the age of the worst parent according to the fitness function value.

Finally, in Jansen and Zarges (2011a) different methods for selection for replacement are defined. In some cases all individuals with the most frequently age within the current selection (including z) are selected for replacement. Another approach is that an element with the minimal age distance to the new search point is selected uniformly at random for removal. Finally, the set of search points whose age occurs fewest within the current selection of search points (including z) are selected for replacement.

Due to the large number of results for ageing-based operators, for simplicity we will focus on the method for setting the age of new search points and how to remove individuals in the population due to age (static, stochastic and hybrid pure ageing for removal due to age). Algorithm 15 is commonly named as *static pure ageing*, the age of the offspring y is set to the same age as the parent x unless it has better fitness, in this case, its age is set to 0 (i. e., if $f(y) > f(x)$ then $y_{\text{age}} := 0$). Note that there are another 2 ways to set the age of an offspring called *evolutionary ageing* and *genotypic ageing* analysed in Jansen and Zarges (2009). In the following we will just mentioned the results obtained by these two methods compared to static pure ageing without going into details.

Algorithm 15 Static Pure Ageing (Age of New Search Points)

Require: Two search points, the parent x and the offspring y.
Ensure: The age of the offspring y.
1: **if** $f(y) > f(x)$ **then** $y_{.\text{age}} := 0$ **else** $y_{.\text{age}} := x_{.\text{age}}$ **end if**

Other results will just be mentioned when needed. We will focus on general ideas like the importance of the τ parameter and how the ageing operator can outperform other randomised search heuristics without these mechanisms, and other randomised search heuristics with

restarts (the complete population is replaced by new search points generated uniformly at random). We will also mention some other ageing-based mechanisms without going to deep into the definition of each operator, instead we will provide a high level description of them, we refer the reader to the source paper for formal definition and specifics of the results.

3.2.6.1 Static Pure Ageing

In static pure ageing for removal due to age operator, a threshold τ is used to indicate the deterministic life time of individuals. When individuals reach an age of $\tau+1$ generations, they are automatically removed from the population. An experimental analysis for this operator was carried out by Castrogiovanni et al. (2007). The static pure ageing for removal mechanism is formalised in Algorithm 16. Note that the (μ+1) EA is identical to the (μ+1) EA with static pure ageing operator when setting the maximal age $\tau = \infty$.

Algorithm 16 Static Pure Ageing (Removal Due to Age)

Require: Set of search points P.
Ensure: Set of search points P without individuals with age $> \tau$.
1: Remove all individuals in P with age $> \tau$.
2: **return** P

Horoba et al. (2009) analysed the influence of the parameter τ in the population. They show that the choice of the maximal age τ can decide between polynomial and exponential optimisation time. Hence, a good choice for the parameter τ can be essential as well as difficult since the range of suitable values for τ may be extremely small. The results are divided in two analyses: situations where a large maximal age is needed and the analysis where the maximal age must not be too large. By setting τ big enough, it may allow the algorithm to reach the global optimum while setting τ small enough may lead to exponential time due to the constant birth and death cycle. Hence, a good choice of the maximal age τ is essential since the range of suitable values for τ may be extremely small.

In Jansen and Zarges (2009) the influence on the performance of static pure ageing, evolutionary ageing, and genotypic ageing is analysed. The main focus of the analysis is the performance of these ageing operators on two example functions, one with a fitness landscape containing a local optimum called $\textsc{LocalOpt}_k$ (see Definition 3.16). And the other with a fitness landscape containing a plateau that contains n points spanning a large Hamming distance of $n-1$ between the first and the last point on the plateau (see Definition 3.17).

Definition 3.16 (LocalOpt$_k$, Oliveto and Sudholt, 2014)**.** *For $n \in \mathbb{N}$ and $2 \leq k = O(1)$ the function* LocalOpt$_k : \{0,1\}^n \to \mathbb{R}$ *is defined for $x \in \{0,1\}^n$ by*

$$\text{LocalOpt}_k(x) := \begin{cases} n \cdot (i \cdot k + 1) & \text{if } x \in \{1^{i \cdot k} 0^{n-i \cdot k} \mid i \in \{1,2,\dots \lfloor n/k \rfloor\}\}, \\ n \cdot (i+1) & \text{if } x \in \{0^{n-i} 1^i \mid i \in \{1,2,\dots \lfloor n/2 \rfloor\}\}, \\ \text{ZeroMax}(x) & \text{otherwise.} \end{cases}$$

A sketch of the function is given in Figure 3.2.

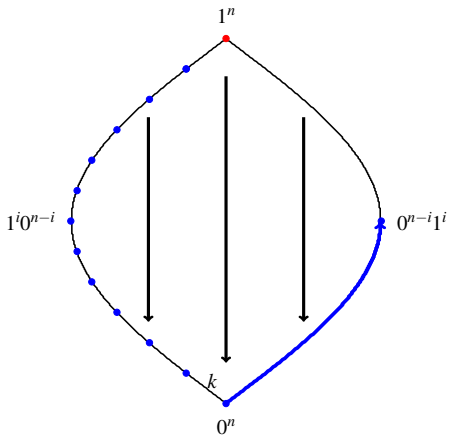


Figure 3.2: Sketch of the function LocalOpt$_k$ (Oliveto and Sudholt, 2014).

Following the description from Oliveto and Sudholt (2014), most of the search space LocalOpt$_k$ follows ZeroMax guiding the search towards the 0^n bitstring. From there an "easy-to-find-path" is found if the i rightmost 0-bits are flipped into 1-bits (i. e., $i \in \{1,2,\dots,\lfloor n/2 \rfloor\}$) and a "hard-to-find" path is found if exactly $i \cdot k$ leftmost bits are flipped into 1-bits (i. e., $i \in \{1,2,\dots,\lfloor n/k \rfloor\}$). The easy path leads to a local optimum represented by the $0^{n/2}1^{n/2}$ bitstring with fitness value $n \cdot (\lfloor n/2 \rfloor + 1)$. The hard leads to the global optimum 1^n of value $n \cdot (\lfloor n/k \rfloor \cdot k + 1)$.

And for the case of the Plateau function, like in LocalOpt$_k$, in the vast majority of the search space the fitness values guide the search towards the 0^n bitstring. The n points $\{1^i 0^{n-i} \mid i \in \{0,1,2,\dots,n-1\}\}$ form the plateau. While the number of points on the plateau is rather small, the Hamming distance between the first and the last point of the plateau equals $n-1$ and is really large. Since there are no hints in which direction better search points can be found, and since all points not on the plateau have worse fitness value, usually the algorithm will perform a random walk on the plateau until they happen to find the unique global optimum 1^n.

Definition 3.17 (PLATEAU, Jansen, 2013)**.** *For $n \in \mathbb{N}$, the function* PLATEAU $: \{0,1\}^n \to \mathbb{R}$ *is defined for $x \in \{0,1\}^n$ by*

$$\textsc{Plateau}(x) := \begin{cases} n & \text{if } x \in \{1^i 0^{n-i} \mid i \in \{0,1,2,\ldots,n-1\}\}, \\ n+1 & \text{if } x = 1^n, \\ \textsc{ZeroMax}(x) & \text{otherwise.} \end{cases}$$

A graphical representation of the PLATEAU *function is given in Figure 3.3.*

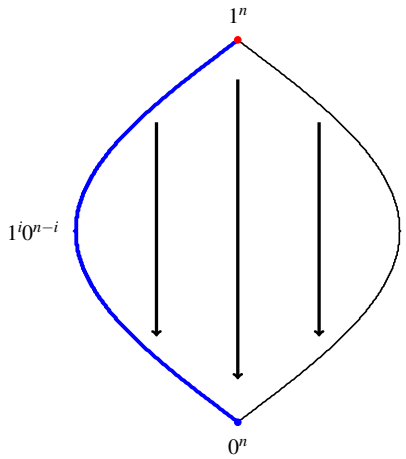


Figure 3.3: Sketch of the function PLATEAU (Jansen, 2013).

Jansen (2002) showed that the (1+1) EA cannot find the global optimum efficiently on a variant of LOCALOPT$_k$ specifically designed for population with very small μ. With high probability the algorithm finds the easy path and ends up being stuck on the local optimum while a (1+1) EA with restarts can optimise this LOCALOPT$_k$ variant in polynomial time. The results for the (μ+1) EA with evolutionary ageing on LOCALOPT$_k$ show that the algorithm requires a very large expected optimisation time. For the case of static pure ageing and genotypic ageing, if the maximal age τ is sufficient large, the population will gather in the local optimum with probability close to 1 (for not too large μ). Since no new search points with larger function values can be created unless the global optimum is found, the algorithm only creates new search points that inherit their age. Creating a copy of a current best search point is much faster that finding the global optimum. So, no new search points enter the population. Thus, after some time, the complete population is replaced by new search points generated uniformly at random (equal to a restart). Since the path to the global optimum is found with not too small probability $\Omega\left(1/n^{k-1}\right)$, on average after $\mathrm{O}\left(n^{k-1}\right)$ restarts the path to the global optimum is found.

Theorem 3.18 (Theorem 2 and 5 in Jansen and Zarges, 2009)**.** *For $n,k \in \mathbb{N}$ with $k = \mathrm{O}(1)$, any number of search points $\mu \in \mathbb{N}$, and a maximal age $\tau \in \mathbb{N}$ with $\tau = \Omega\big(n^{k+1} + \mu n \log n\big)$. Algorithm 14 with static pure ageing or genotypic ageing has expected optimisation time $\mathrm{O}\big(\tau n^{k-1}\big)$ on* LocalOpt$_k$.

The general idea is that, there is always a restart if all search points are removed due to their age simultaneously. This happens once the local optimum has been found and copies of the current best search point are been created. So the proof takes the time required to reach the local optimum and the population consist of copies of the local optimum, then after τ generations $\big(\mathrm{O}\big(\tau + n^2 + \mu n \log n\big)\big)$ a restart is done. Multiplying it with the expected time to reach the global optimum $\mathrm{O}\big(n^{k-1}\big)$, the upper bound on the expected optimisation time is $\mathrm{O}\big(\tau n^{k-1} + n^{k+1} + \mu n^k \log n\big)$. With $\tau = \Omega\big(n^{k+1} + \mu n \log n\big)$ the bound simplifies to $\mathrm{O}\big(\tau n^{k-1}\big)$. In this case there is not a real advantage in using ageing over a simple restart strategy.

For the case of Plateau, the expected optimisation time of the (1+1) EA is $\Theta\big(n^3\big)$ (Theorem 7 in Jansen and Wegener, 2001) and for the (μ+1) EA is $\mathrm{O}\big(\mu n^3\big)$ (Theorem 3 in Witt, 2006). For the case of the performance of the ageing operators on Plateau, the proofs ideas use the family trees method (see Section 2.3.7). For evolutionary ageing and genotypic ageing, the expected optimisation time of the (μ+1) EA with these ageing operators and maximal age $\tau = \omega(\log n \cdot (n + \mu \log n))$, is $\mathrm{O}\big(\mu n^3\big)$

Evolutionary ageing and genotypic ageing does not change much compared to the corresponding algorithm without ageing on Plateau given that the maximal age τ is sufficiently large. For the case of static pure static ageing, descendants of plateau points that are on the plateau inherit the age of their parents, which may lead to the extinction of the whole population on the plateau if the maximal age τ is not sufficiently large.

Theorem 3.19 (Theorem 4 in Jansen and Zarges, 2009)**.** *For $n \in \mathbb{N}$, let $\alpha(n) = \omega(1)$ and $\alpha(n) = \mathrm{O}\Big((n/\ln)^{4/3}\Big)$. Then for $\mu, \tau \in \mathbb{N}$ with $\mu = \mathrm{poly}\,(n)$, $\tau = \omega(\log n \cdot (n + \mu \log n))$ and $\tau = \mathrm{O}\big(n^3/(\alpha(n) \ln n)\big)$, with probability $1 - n^{-\Omega\left(\sqrt{\alpha(n)}\right)}$, the optimisation time of the (μ+1) EA with static pure ageing on* Plateau *is $n^{\Omega\left(\sqrt{\alpha(n)}\right)}$.*

In this case, individuals can still reach the plateau but the main difference is that individuals inherits the age of the parent once they have reached the plateau. The resulting random process corresponds to a fair random walk which leads to the constant extinction of the population before any progress is achieved. And in the same way as LocalOpt$_k$, the use of ageing does not translate into improvements compared to its plain version.

We conclude this section by mentioning the work done by Oliveto and Sudholt (2014) on static pure ageing on the Balance function (see Definition 3.1). The following theorem

shows that the (μ+1) EA with static pure ageing as a removal operator fails for any value of τ. If τ is not large enough to allow the individuals to improve their fitness before reaching age τ, then no local optimum will be found at all. For larger values of τ it is shown that w. o. p. the trap is reached before the global optimum. Consequently, when the individuals are eventually restarted they end up in the trap all over again w. o. p.

Theorem 3.20 (Theorem 8 in Oliveto and Sudholt, 2014)**.** *Let* $\varphi > 38\mu n^{3/2}$ *and* $\mu \leq n^{1/2-\varepsilon}$. *Then the* ($\mu$+1) EA *with static pure ageing requires at least* $2^{\Omega(\sqrt{n})}$ *generations to optimise* BALANCE *w. o. p. for every value* τ.

3.2.6.2 Stochastic Pure Ageing

In stochastic pure ageing individuals are given an expected life time. In each generation an individual has a probability to survive of $p_{\text{live}} = e^{-\frac{c}{\tau'}}$ (i. e., hence a probability to die of $p_{\text{die}} = 1 - e^{-\frac{c}{\tau'}}$) where c is usually a constant and τ' the main parameter operator. Hence, the expected life time of an individual is $\mathrm{E}[\tau] = 1/p_{\text{die}}$. The stochastic pure ageing mechanism is formalised in Algorithm 17.

Algorithm 17 Stochastic Pure Ageing (Removal Due to Age)

Require: Set of search points P.
Ensure: Set of search points P with removed individuals with probability p_{die}.
1: **for each** $x \in P$ **do**
2: Let $P = P \setminus \{x\}$ with probability p_{die}.
3: **end for**
4: **return** P

Jansen and Zarges (2009) have shown how a (μ+1) EA cannot find the global optimum of the LOCALOPT$_k$ function efficiently because there is a high probability that the algorithm finds the easy path and ends up being stuck on the local optimum. On the other hand, from Theorem 3.18, we know that the (μ+1) EA with static pure ageing (age assignment for new individuals and removal operator) can optimise LOCALOPT$_k$ efficiently. Static pure ageing can implicitly restart the population allowing it to escape from the local optimum and to eventually find the path leading to the global optimum.

In Oliveto and Sudholt (2014) it is shown that stochastic pure ageing can optimise LOCALOPT$_k$ if the population size is not too large in a similar fashion. First, the time reaching the local optimum is considered. If the wrong local optimum $0^{\lceil n/2 \rceil}1^{\lceil n/2 \rceil}$ is reached, the (μ+1) EA can escape by performing a restart where all individuals die in the same generation. Once a local optimum with leading ones form is reached, there is a good chance for the (μ+1) EA to climb up these local optima until the global optimum is reached.

Theorem 3.21 (Theorem 4 in Oliveto and Sudholt, 2014)**.** *The expected optimisation time on* LocalOpt$_k$ *of the* (μ+1) EA *with stochastic pure ageing, population size* $\mu = \mathrm{O}(1)$ *and*

$$p_{\mathrm{die}} = \min\left\{\frac{1}{320\mu^2}, \frac{1}{48\mu}\cdot\left(\frac{1}{c\mu n^{k+1}}\right)^{1/\mu}\right\}$$

for a sufficiently large constant $c > 0$*, is at most*

$$\mathrm{O}\left(\mu n^{k+1} + n^{k-1}\cdot p_{\mathrm{die}}^{-\mu-1}\right) = \mathrm{O}\left(n^{2k+(k+1)/\mu}\right).$$

For large population sizes, stochastic pure ageing fails to work efficiently. Oliveto and Sudholt (2014), showed that the effects of stochastic pure ageing can range from a population evolving without guidance of the fitness to take exponential time to escape from the local optimum due to its inability to reproduce a complete restart.

3.2.6.3 Hybrid Pure Ageing

In Oliveto and Sudholt (2014), the authors observed that stochastic pure ageing is effective in achieving implicit restarts and in escaping from local optima beyond restarts only if the population size is not too large, so in order accomplish the same results but with larger populations, they introduce a hybrid mechanism which attempt of taking advantages from the static and stochastic pure ageing operators. In hybrid pure ageing, the operator does not act on an individual until its lifetime reaches a deterministic value τ. All the individuals with age greater than τ survive with probability p_{live} and are removed from the population with probability $p_{\mathrm{die}} = 1 - p_{\mathrm{live}}$ like in stochastic ageing. This allows, even in large populations, both complete restarts and the possibility for some individuals to survive after they have escaped a local optimum. The hybrid pure ageing mechanism is formalised in Algorithm 18.

Algorithm 18 Hybrid Pure Ageing (Removal Due to Age)

Require: Set of search points P.
Ensure: Set of search points P with removed individuals with probability p_{die}.
1: **for each** $x \in P$ **do**
2: **if** $x_{.\mathrm{age}} > \tau$ **then** let $P = P \setminus \{x\}$ with probability p_{die} **end if**
3: **end for**
4: **return** P

The (μ+1) EA with hybrid pure ageing for LocalOpt$_k$ is analysed in Oliveto and Sudholt (2014). The basic idea follows the same line of though as the one used in the proof for Theorem 3.18.

Theorem 3.22 (Theorem 7 in Oliveto and Sudholt, 2014)**.** *Let* $k = \mathrm{O}(1)$*,* $p_{\text{live}} = 1/\mu$ *and* $\tau = \omega\big(\log n \cdot (n^k + \mu n) + \mu^3\big)$*. The* $(\mu{+}1)$ EA *with hybrid pure ageing optimises the function* $\textsc{LocalOpt}_k$ *in expected time* $\Omega\big(\tau n^{k-1} + n^{k+1}\big)$.

First, it is assumed that the local optimum is found before the global optimum. Once the local optimum is reached, the time required for the whole population to collapse on the local optimum is taken into consideration. Then, the time required for all individuals to have the same age is considered. Since only copies are accepted and these inherit their parent's age all the population will reach age $\tau + 1$ at the same generation, and each individual will die with probability p_{die}. At this moment, it is necessary to wait for a "genuine restart", i. e., the whole population dies in the same generation escaping from the local optimum. If a genuine restart happens, after $\mathrm{O}\big(n^{k-1}\big)$ expected number of genuine restarts, the best individual will reach the global optimum before the age τ is reached.

For the case of Balance, the $(\mu{+}1)$ RLS with hybrid pure ageing instead of the $(\mu{+}1)$ EA is analysed in Oliveto and Sudholt (2014). The proof idea takes into consideration that the trap is found before the global optimum by all individuals but with constant probability the individuals reach an age $\tau + 1$ and there will be at least one individual with exactly $n/16 + 1$ zeroes (i. e., it will escape the trap if it flips a 1-bit of the suffix and survives the restart). Again, it is necessary to wait till a survivor escapes from the trap and the survivor reach the global optimum by making improvements before τ generations.

Theorem 3.23 (Theorem 10 in Oliveto and Sudholt, 2014)**.** *Let* $\mu \leq n^{1/2-\varepsilon}$*,* $\tau > c\mu n \ln n$ *and c a large enough constant. Then the expected time for the* $(\mu{+}1)$ RLS *with hybrid pure ageing to optimise the* Balance *function is* $\mathrm{O}\big(\mu^2 n^3\big)$ *for any frequency of change* $\varphi > 0$*.*

3.3 Diversity Benefits Crossover

This section is based on Section 4 from Sudholt (2018), we adopt a similar structure but we present the results in a different way and we expand Section 4.5 in Sudholt (2018) by introducing the results from ageing and crossover in Section 3.3.1.

The general framework of an EA with crossover is described in Algorithm 19. Parents are selected based on some policy (uniformly at random) and the crossover operator is performed with probability p_c; this operator can be any of the operators mentioned in Chapter 1, like uniform crossover or k-point crossover. Mutation is performed with mutation rate p (normally $p = 1/n$ unless stated otherwise). The replacement policy selects the μ best individuals and in case of ties, some tie-breaking rule can be used like breaking ties uniformly at random.

One of the major concerns related to crossover is to define how this procedure can be helpful in the evolution process. The area of runtime analysis has been trying to provide

Algorithm 19 Scheme of a (μ+λ) GA with mutation rate p, and crossover with probability p_c for maximising $f : \{0,1\} \to \mathbb{R}$

Let $t := 0$ and initialise P_0 with $\mu \in \mathbb{N}$ individuals chosen uniformly at random.
while stopping criterion **not** met **do**
    Let $P' = \emptyset$.
    for $i = 1$ **to** λ **do**
        Choose $p \in [0,1]$ uniformly at random.
        if $p \leq p_c$ **then**
            Select two parents x_1, x_2.
            Let $y :=$ crossover(x_1, x_2).
        else
            Select a parent y.
        end if
        Create y' by flipping each bit in y independently with probability p.
        Add y' to P'.
    end for
    Let P_{t+1} contain the μ best individuals from $P_t \cup P'$; break ties according to a specified tie-breaking rule.
    Let $t := t + 1$.
end while

answers to the questions: how crossover is able to promote diversity? And how helpful is crossover in an optimisation problem? The first theoretical analysis towards these answers was done by Jansen and Wegener (2002) (we will review these results in Section 3.3.1.1). Another study in the same direction was done by Jansen and Wegener (2005). The authors showed that a (μ+1) GA was able to optimise 2 example functions (called *real royal road functions*) in polynomial expected optimisation time while mutation-based EAs needed exponential expected optimisation time w. o. p.

The main feature of this (μ+1) GA is that a simple rule to increase the diversity on the population is added to the replacement policy: once an offspring is added to the population (if has a better fitness than an individual with worst fitness of the current population), then a multi-set W of individuals with worst fitness is selected, and from W another set W' is formed from the individuals with the largest number of copies in W. Then, one element in W' is removed from the current population uniformly at random.

Another simple strategy that showed to be better than mutation-based EAs was the *greedy* (2+1) GA with uniform crossover and standard bit mutation for the case of hill climbing (ONEMAX) analysed by Sudholt (2012). This greedy GA chooses two parents with the best fitness uniformly at random, then crossover and mutation are applied, then it proceeds as the standard (μ+1) EA. Sudholt showed that this greedy (2+1) GA with uniform crossover

and standard bit mutation is at least as twice fast as its plain version with just standard bit mutation (up to small order terms). Then, Sudholt (2017) extended the idea to a more general analytical framework that applies to all (μ+λ) GA (including the greedy (2+1) GA), subject to mild conditions. One feature of this (μ+λ) GA, was its tie-breaking on the selection for survival step; better individuals are selected with the individuals with the fewest duplicates. Again, it is shown that this (μ+λ) GA with uniform crossover and standard bit mutation is at least twice as fast as every EA with only standard bit mutations on ONEMAX.

Corus and Oliveto (2017) showed that it still possible to achieve speed-ups on ONEMAX without some diversity-based tie-breaking rule. By always applying uniform crossover (each bit of the offspring is chosen from each parent with probability $1/2$), an offspring is created and standard bit mutation (with mutation rate c/n) is applied to this offspring. Finally the best μ individuals are selected from the $\mu+1$ individuals and ties are broken uniformly at random.

As can be seen, in order for crossover to work, some diversity has to be achieved (like simple greedy strategies or removing duplicates mentioned previously) before crossover shows its full potential. In the following sections we will focus on works that show that crossover can outperform mutation-based EAs by using explicit diversity-mechanisms embedded into a GA (Section 3.3.1) and by means of diversity emerging by the implicit mechanism: mutation (Section 3.3.2).

3.3.1 Escaping Local Optima with Diversity Mechanisms and Crossover

In this section we will review how some GAs with diversity mechanisms. In Section 3.3.1.1 we focus our attention on the performance of the (μ+1) GA with duplicate elimination, duplicate minimisation, deterministic crowding, convex hull and Hamming distance maximisation, and population-based fitness sharing and how these mechanisms with crossover can escape from local optima (Dang et al., 2016b). In Section 3.3.1.2 we mention some results that show how ageing and crossover can improve the performance of a (μ+1) EA compared to a plain (μ+1) EA or equipped with restart strategies.

3.3.1.1 Diversity Mechanisms and Crossover

Unlike the (μ+1) EA that only uses mutation, the (μ+1) GA uses crossover and mutation operators. Dang et al. (2016b) introduced diversity mechanisms into this algorithm and they provide an analysis of its performance on the JUMP_k function (see Definition 3.24, Jansen and Wegener, 2002).

Definition 3.24 (JUMP$_k$, Jansen, 2013)**.** *Let $n \in \mathbb{N}$ and $k \in \{1,2,\ldots,n\}$ be given. The fitness function* JUMP$_k : \{0,1\}^n \to \mathbb{R}$ *is defined by*

$$\text{JUMP}_k(x) := \begin{cases} n - \text{ONEMAX}(x) & \text{if } n-k < \text{ONEMAX}(x) < n, \\ k + \text{ONEMAX}(x) & \text{otherwise.} \end{cases}$$

A graphical representation of JUMP$_k$ *is given in Figure 3.4.*

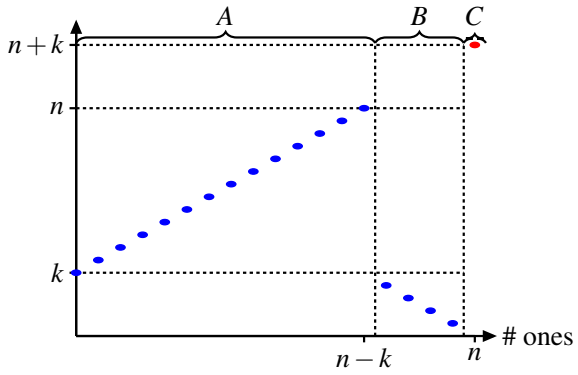


Figure 3.4: Sketch of the function JUMP$_k$ with $n = 18$ and $k = 5$ (Jansen, 2013).

As shown in Figure 3.4, the JUMP$_k$ function is divided in 3 sections, the section denoted by A is directed into the direction of the 1^n bitstring but that leads to a locally optimal plateau with fitness $n-k$. Section B is defined by the points with more $n-k$ but less than n 1-bits, the function value is given by $n - \text{ONEMAX}(x)$ (the values vary between 1 and $k-1$) with smaller function values than all bitstrings not in this part. Finally, the unique global optimum 1^n forms the third and last part of the search space denoted by C.

The main goal of any EA on this function is to overcome the fitness valley with Hamming distance k from the optimum. Jansen and Wegener (2002) showed that mutation-only algorithms such as the (1+1) EA require expected time $\Theta\left(n \log n + n^k\right)$, which is the expected time to reach the plateau plus the expected time required to mutate exactly the k bits simultaneously in order to reach the global optimum. For large $k \geq 2$ this time becomes $\Theta\left(n^k\right)$. A simple (μ+1) GA with uniform crossover only needs time $\text{O}\left(\mu n^2 k^3 + 4^k/p_c\right)$. This time is $\text{O}\left(4^k/p_c\right)$ for large k, and hence significantly faster than mutation-only EAs.

Kötzing et al. (2011) later showed that the (μ+1) GA with no genotype duplicates by mutation and with/without mutation after crossover also works with very small populations ($\mu \geq 2$ and $\mu \leq \text{poly}(n)$) and not too big crossover probability ($p_c = k/n$) on JUMP$_k$. The expected optimisation time in Kötzing et al. (2011) analysis is $\text{O}\left(\mu n \log n + e^{6k} \cdot \mu^{k+2} \cdot n\right)$, if $\mu = 2$ and $k = c \log\log n$ for a sufficiently small c, the expected time simplifies to $\text{O}(n \log n)$.

As can be noticed from the previous results, it is possible to achieve speedups on the optimisation time by using crossover and mutation. However, if all individuals in the population are very similar, crossover will create a similar offspring, so in order for crossover to work, some degree of diversity has to be achieved before applying the operator. Dang et al. (2016b) used diversity mechanisms to improve the population diversity. The goal was to study how diversity helps to escape local optima and how crossover can be enhanced by promoting diversity in the population. The (μ+1) GA uses explicit diversity mechanisms in the tie-breaking rule for the survival selection. The main results are summarised in Table 3.2 (Table 1 in Dang et al., 2016b).

Table 3.2: Overview of the main results from Dang et al. (2016b). The table represents the run time bound for best-possible population size $\mu = 2$ for each diversity mechanism. The results for deterministic crowding only holds for $p_c = k/n$, for the rest of the mechanisms, the results hold for constant crossover probability $p_c < 1$.

Mechanism	**General case**	$\mathbf{k = 2}$	$\mathbf{k = 4}$
Duplicate elimination (Theorem 3.1)	$O\left(n\log n + n^{k-1}\right)$	$O(n\log n)$	$O\left(n^3\right)$
Duplicate minimisation (Theorem 3.2)	$O\left(n\log n + n^{k-1}\right)$	$O(n\log n)$	$O\left(n^3\right)$
Deterministic crowding (Theorem 3.3)	$O\left(n\log n + ne^{5k}\right)$	$O(n\log n)$	$O(n\log n)$
Convex hull max. (Theorem 4.1)	$O\left(n^2\log n + 4^k\right)$	$O\left(n^2\log n\right)$	$O\left(n^2\log n\right)$
Hamming distance max. (Theorem 4.2)	$O\left(n\log n + nk\log k + 4^k\right)$	$O(n\log n)$	$O(n\log n)$
Fitness sharing (Theorem 4.4)	$O\left(n\log n + nk\log k + 4^k\right)$	$O(n\log n)$	$O(n\log n)$

More general results can be found in the respective theorems. Another contribution made from Dang et al. (2016b) is that the upper bounds do not rely on small crossover probabilities (like in Jansen and Wegener, 2002 and Kötzing et al., 2011), in this case all diversity mechanisms mentioned in Table 3.2 work for more realistic constant probabilities $p_c \leq 1$ (except for deterministic crowding). In the following we will explain the tie-breaking rules based on each diversity mechanism, along with the main ideas behind each analysis.

In *duplicate elimination* as a tie-breaking mechanism, if there are duplicates of the worst individual, one is selected for removal so that the number of duplicated individuals decreases. If no duplicates with the worst individual are found, a least-fit individual is selected uniformly at random for removal. The analysis it is based on the idea that once the whole population has reached the plateau in expected time $O(\mu n + n\log n)$ and after $O\left(\mu^2 n\right)$ generations in expectation, there will be $c\mu$ duplicates in the population (with $0 < c < 1$ being an arbitrary constant). In this scenario crossover can create a surplus of ones, so that mutations only has to flip the remaining at most $k-1$ zeroes, so the expected number of generations until the optimal string appears is $\Omega\left(n^{k-1}\right)$.

For *duplicate minimisation*, if there is a large number of duplicates in the population, an individual from this subpopulation is selected for removal. In this sense the largest group of duplicates cannot increase over time. In this case it is just necessary to wait until the size of this group is at most $c\mu$ in expected time $O(\mu n)$. Then, the same waiting time for flipping at most $k-1$ bits from duplicate elimination apply.

In the case of *deterministic crowding*, the offspring always survive. If just mutation is used, the parent is always removed and if crossover is used, one of the parents is selected uniformly at random and removed from the population while the offspring gets to the next generation. The main idea to create individuals with not a single 0 in common due to mutation (no crossover), and these individuals get chosen for crossover that succeeds in creating the optimum.

Convex hull maximisation focusses in the maximisation of the number of bit positions where the population contains both a 0 and an 1 some individual. First, it is necessary to wait until a maximum amount of diversity is reached, i. e., in expected time $O\big(\mu n^2 \log n\big)$. Now, since all two individuals have no 0s in common, an optimal individual can be formed by crossover with probability $1/2^{2k}$ with a waiting time $4^k \cdot p_c^{-1}$.

Hamming distance maximisation removes an individual from the population such that the overall Hamming distance of the population is maximised without that individual. Similar to convex hull maximisation, the algorithm reaches a population of maximal diversity in expected time $O\big(\mu^2 kn \log(\mu k)\big)$ and by repeating the same arguments from convex hull maximisation the bound is obtained.

For *fitness sharing*, the idea of population-based fitness sharing is used, the population with the maximum shared fitness is the one that passes to the next generation. With $\sigma \geq 2k$, population-based fitness sharing turns out to be equivalent to maximising the total Hamming distance between pairs of search points, obtaining the same results. We conclude this section by mentioning that this is not the only result concerning population-based fitness sharing and crossover; Fischer and Wegener (2005) and Sudholt (2005) have shown that GAs with this diversity mechanism perform better than mutation-based EAs for the Ising model (Ising, 1925).

The Ising model was first presented by Naudts and Naudts (1998) as subject for the investigation of GAs and EAs. In its general form it is NP-hard to solve and consists of an undirected graph $G = (V,E)$, $V = \{1,\ldots,n\}$ and a search point $x = (x_1,\ldots,x_n)$ represents a colouring of V. An edge $e = \{u,v\}$ contributes the value $f_e(x) := w(e) \cdot x_u \cdot x_v$ to the fitness where $w(e)$ is the weight of the edge and $x_u, x_v \in \{-1,+1\}$. The fitness of an individual x is the sum of all $f_e(x)$ and has to be maximised. In case where $w(e) = 1$ for all $e \in E$ by a affine transformation, the state space can be changed to $\{0,1\}^n$ and the fitness $f_e(x)$ equals

the number of monochromatic edges leading to the 0^n and 1^n being optimal and being the only optimal states for connected graphs.

Fischer and Wegener (2005) showed that population-based fitness sharing enhances crossover for the *Ising model on the ring*, an one-dimensional undirected graph than can have one of two colours (or states) 0 or 1. For this fitness function, the value corresponds to the number of monochromatic edges, and all colourings where all connected components have the same colour are global optima. One particular characteristic of this ring model is that it contains large plateaus that the GA or EA has to overcome in order to find one of the global optimum. As mentioned by Sudholt (2018), let us say that there is one individual with the following genotype structure 0001111000, a mutation flipping: 0001111000, 0001111000, 0001111000 or 0001111000 are fitness-neutral, i. e., a plateau with no effect on the fitness. For this it is necessary to change complete blocks to create an improvement. Fischer and Wegener (2005) showed that the expected optimisation time for the (1+1) EA on the Ising model on the ring is $\Theta\left(n^3\right)$ while a (2+2) GA using two-point crossover with population-based fitness sharing with Hamming distance function and parameters $\sigma = n$ and $\alpha = 1$ can optimise the Ising model on the ring in expected time $\mathrm{O}\left(n^2\right)$.

As mentioned in Section 3.2.5 for the case of the TwoMax function, population-based fitness sharing promotes the creation of individuals in opposite directions. For this case it is not different, dissimilar individuals will have a higher shared-fitness. If the two-point crossover chooses properly where to place the cutting points from complementary individuals, it is possible to invert whole blocks.

Sudholt (2005) investigated another instance of the Ising model, *complete binary trees*. In this problem class the building blocks form subtrees, and subtrees of the same color represent building blocks of good solutions. Finally, let us note that this class of problem has difficult local optima, for example when the two subtrees of the root are coloured with different colours. Sudholt (2005) proved that the $(\mu+\lambda)$ EA with mutation rate p_m needs at least expected time $2^{\Omega(n)}$ to find a global optimum on the complete binary tree Ising class while the (2+2) GA with two-point crossover and population-based fitness sharing with parameters $\sigma = n$, $\alpha = 1$, Hamming distance, and probability $p_c = 1/2$ can find the optimal in the complete binary tree class in expected time $\mathrm{O}\left(n^3\right)$.

Again, population-based fitness sharing promotes the creation of individuals with maximum Hamming distance between them. In the case of x and y being complementary, two-point crossover can effectively substitute subtrees to increase the fitness. In this problem class there is no plateau; this means that any change in the Hamming distance has an effect on the fitness value of the populations, i. e., instead of individuals, the replacement is done at the level of populations, e. g., $P = \{x, y\}$ (the parent and current population) and $P' = \{x', y'\}$

(the offspring population), the population P' replaces P if the shared fitness is $f(P') \geq f(P)$. In the analysis Sudholt (2005) shows that there are several situations where the fitness can improve by performing more complex operations (mutation and/or crossover); however, these steps have probability $\Omega\left(1/n^2\right)$, leading to the overall time bound of $\mathrm{O}\left(n^3\right)$.

3.3.1.2 Ageing and Crossover

As mentioned in Section 3.2.6.1, in Horoba et al. (2009) it is shown that the proper setting for τ can make the difference between very inefficient and efficient search. In Jansen and Zarges (2009), the authors have pointed out that in many cases ageing may be replaced by a suitable restart mechanism that has the same beneficial effects on the performance of the search heuristic. A question left open was: is there a real advantage of using ageing over restarts? Since restarts are easier to implement and computationally simpler and cheaper in comparison to ageing, in Jansen and Zarges (2010a), the authors showed that an EA with crossover, mutation, and static pure ageing can efficiently optimise their example function f (see Definition 8 in Jansen and Zarges, 2010a) while other randomised search heuristics without crossover and ageing, and with just restarts fail completely.

In this case, an offspring can be produced not just by mutation but also by crossover. Variation creates one new search point y by means of k-point crossover and standard bit mutations. The crossover operator cuts two search points into $k+1$ pieces using k randomly selected crossover points. The two search points are selected uniformly at random. If no crossover is carried out (with probability $1-p_c$) one search point is selected uniformly at random and undergoes standard bit mutation. In this mutation each of the n bits is flipped independently with probability $1/n$. Now, the "classic" static pure ageing has to be changed to deal with two parents, so if the fitness of the offspring is better than the fitness of both parents, the age of the offspring is set to 0, if it is worst, the age of one the parents (selected uniformly at random) is used to define the age of the offspring.

So in order to show that the new (μ+1) EA with crossover and ageing operators works better than simple restarts, Jansen and Zarges (2010a) make use of a function simply called $f : \{0,1\}^n \to \mathbb{R}$ (see Definition 8 in Jansen and Zarges, 2010a), with structure similar to $\textsc{LocalOpt}_k$, with a vast majority of the points on the search space leading to the 0^n bitstring. It has a path of Hamming neighbours of the form 1^i0^{n-i} leading to a local optimum at $1^{n/4}0^{3n/4}$, with the difference that the points of the form $1^{n/4}0^{n/4}q$ with $q \in \{0,1\}^{n/2}$ and $\textsc{OneMax}(q) \geq n/12$ are defined as the set of all global optima of the f function (cf. Figure 1 in Jansen and Zarges, 2010a).

By allowing partial restarts, i. e., some individuals are removed due to its age and some individuals remain on the path 1^i0^{n-i} or at the local optimum $1^{n/4}0^{3n/4}$, the set of optima

points can be reached by applying crossover to new generated individuals and individuals from the path; the structure of f is tailored in such way that it can be optimised by crossover and ageing. General search heuristics without crossover will find it difficult to optimise this function efficiently, and a complete restart will always maintain a cycle between new born individuals and individuals stuck on the local optimum. In the following we just mention the upper and lower bounds on the expected optimisation time of the (μ+1) EA with k-point crossover and static pure ageing on the f function. The proof ideas follow from the arguments used from previous results and from the previous paragraph.

Theorem 3.25 (Theorem 10 in Jansen and Zarges, 2010a)**.** *Consider the function f. The* (μ+1) EA *with static pure ageing with population size $\mu \in \mathbb{N}$ with $\mu \geq 2$ and $\mu = n^{\mathrm{O}(1)}$, crossover probability p_c with $0 < \varepsilon \leq p_c \leq 1-\varepsilon < 1$ for a positive constant ε, $k = \mathrm{O}(1)$ crossover points in k-point crossover, and maximal age $\tau = \omega(\mu n \log \mu)$ has expected optimisation time $\mathrm{O}\big(\mu \cdot \big(\tau + n^2 + \mu n \log n\big)\big)$.*

Theorem 3.26 (Theorem 11 in Jansen and Zarges, 2010a)**.** *Consider the function f. The* (μ+1) EA *with static pure ageing with population size $\mu \in \mathbb{N}$ with $\mu \geq 2$ and $\mu = n^{\mathrm{O}(1)}$, crossover probability p_c with $0 < \varepsilon \leq p_c \leq 1-\varepsilon < 1$ for a positive constant ε, $k = \mathrm{O}(1)$ crossover points in k-point crossover, and any maximal age τ has expected optimisation time $\Omega\big(\tau + n^2 + \mu n \log n\big)$.*

We would like to mention that Jansen and Zarges (2010b) use the same function analysed by Jansen and Zarges (2010a) but with different ways of setting the age of the offspring. Due to crossover, each new search point has two parents as origin, so what age has to be inherited if no improvement was made? This question is addressed in Jansen and Zarges (2010b) by providing three variants that follow the general scheme given by static pure ageing for two parents. In *age-based static pure ageing* the age of the offspring is set using the age of the older parent. *Optimistic value-based static pure ageing*, the age of the offspring is set by using the age of the best parent according to the fitness function value or *pessimistic value-based static pure ageing*, setting the age of the offspring with the age of the worst parent according to the fitness function value.

We conclude the results obtained by ageing and crossover by mentioning that in Jansen and Zarges (2011b) the same f function is analysed and the same algorithmic framework is used with the same variants for static pure ageing, but this time different selection for replacements are introduced. *Most frequent replacement*, all individuals with the most frequently age within the current selection are selected for replacement. In *smallest age distance replacement*, an element with the minimal age distance to the new search point is selected uniformly at random for removal. In *random replacement* a search point is

selected from replacement uniformly at random and in *fewest replacement*, the set of search points whose age occurs fewest within the current selection of search points. Each of this replacement mechanisms offer different degrees of diversity with respect to age.

3.3.2 Escaping Local Optima with High Mutation Rates and Crossover

In Section 3.3 and 3.3.1 we mentioned examples where some degree/method for diversity is introduced to the plain GA to speed up hill climbing on ONEMAX and to speed up the escape from local optima on JUMP_k, respectively. Now, in this section we introduce the work done by Dang et al. (2017a), in which the authors show that crossover, when followed by high mutation rates, may lead to a sudden burst of diversity.

As mentioned before, the work done by Jansen and Wegener (2002) and Kötzing et al. (2011) relied on mutations to provide diversity with small crossover probability. Both approaches focussed on creating diversity through a sequence of mutations, relying on crossover to create the optimum once sufficient diversity has been created. In Dang et al. (2016a) it is shown that diversity can also be created and speed-ups can be achieved by introducing frequent applications of crossover followed by standard mutation rates but restricted to very short jumps ($k = \text{O}(1)$). Then, Dang et al. (2017a) extended the analysis done in Dang et al. (2016a) by introducing higher mutation rates and generalising the results to a much larger class of JUMP_k, only requiring $k = \text{o}(n)$. The resulting optimisation time using standard mutation rate with $k = \text{o}(n)$ is stated as follows.

Theorem 3.27 (Theorem 2 in Dang et al., 2017a)**.** *The expected optimisation time of the* (μ+1) GA *with* $p_c = 1$ *and* $\mu \leq \kappa n$*, for some constant* $\kappa > 0$ *on* JUMP_k*, with* $k = \text{o}(n)$ *is* $\text{O}\Big(\mu n\sqrt{k}\log\mu + n^k/\mu + n^{k-1}\log\mu\Big)$.

For $\mu = \kappa n$ and $k \geq 3$, the bound simplifies to the dominant term $\text{O}\big(n^{k-1}\log n\big)$, achieving a speedup of order $\Omega(n/\log n)$ compared to the expected time of $\Theta\big(n^k\big)$ for the (1+1) EA (Jansen and Wegener, 2002). The analysis follows the same ideas from previous results; once the population has reached a fitness of $n-k$ (the plateau), in the worst case scenario, there is no diversity: all individuals are identical. Selection and crossover cannot create different offspring from the parents. But with mutation, new local optima can be created by flipping a 1-bit back to 0 from the $n-k+1$ bits available. The algorithm now can spend time creating new individuals at the plateau. Finally, once several local optima have been created by mutation, crossover has a chance to create a surplus of 1-bits, and then the global optimum can be found by flipping the at most $k-1$ remaining 0-bits to 1.

Now higher mutation rates $(1+\delta)/n$ for a constant $\delta > 0$ leads to an increment on the diversity, i. e., the larger the mutation rate, the easier it is to promote the emergence and

maintenance of diversity. This leads to the following improved upper bound, which for reasonably small μ and for $k \geq 3$, simplifies to the dominant term $\mathrm{O}\left(n^{k-1}\right)$, achieving a speed-up of order n compared to the expected time of the (1+1) EA.

Theorem 3.28 (Theorem 3 in Dang et al., 2017a)**.** *The* (μ+1) GA *with mutation rate* $(1+\delta)/n$*, for a constant* $\delta > 0$*, and population size* $\mu \geq ck\ln n$ *for a sufficiently large constant* $c > 0$*, has for* $k = \mathrm{o}(n)$ *expected optimisation time* $\mathrm{O}\left(n\sqrt{k}\mu\log\mu + \mu^2 + n^{k-1}\right)$ *on* $\textsc{Jump}_k$.

3.4 Diversity in Island Models

As mentioned in Section 3.1, the main interest of this chapter (and this thesis) is on panmictic populations, but because of its importance as a diversity mechanism, we will review some of the results obtained with island models. This section is loosely based on Section 3.2 from Sudholt (2018) but without going into too much detail. For a more detailed survey on this diversity mechanism and on runtime analyses on parallel EAs we refer the reader to Sudholt (2015).

Due to their highly parametrisable nature (problem representation, operators, evolution mode, etc.), EAs are specifically adequate for the analysis of hybrid and parallel approaches (Cotta-Porras, 1998). As mentioned before, some of the mechanisms include the partition of the population into subpopulations; in the case of island models, each island can run an EA and with the help of a migration policy communicate each other by sending/receiving selected search points.

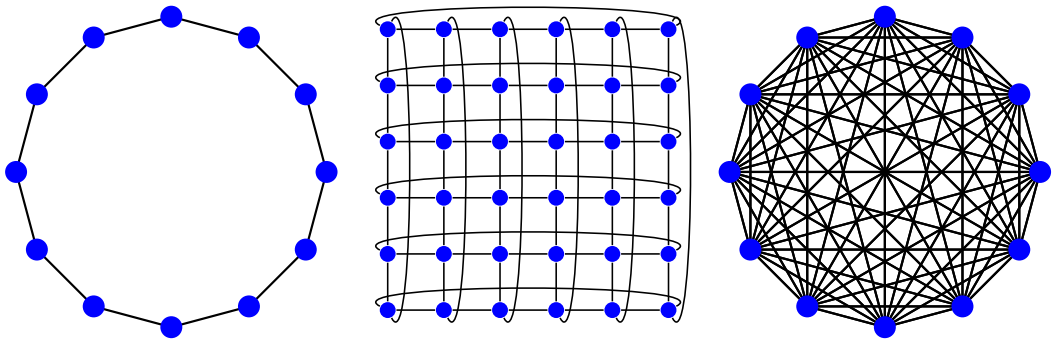
Each island can run an EA on parallel hardware in various ways. It is possible to parallelise specific operations, or to parallelise the evolutionary process itself. These island models communicate with other islands based on a topology that connects the islands, and migration involves sending solutions to all neighbouring islands. Often periodic migration is used: migration happens every τ iterations (*migration interval*). The solutions sent are then considered for inclusion on the receiving island according to a selection process.

One of the major advantages is that the islands evolve independently, and the flow of genetic information in the whole system is slowed down (compared to having one large population). In the same way as selection pressure in panmictic populations can be used to control diversification and intensification, the migration interval τ, the migration policy, the number of individuals to be migrated, or the selection schemes for emigration and immigration can be used to increase diversity and to prevent (or at least delay) premature convergence. Algorithm 20 shows a general scheme of a basic island model.

Algorithm 20 Structure of an island model with migration interval τ (Sudholt, 2015)

Let $t := 0$.
Initialise a population made up of subpopulations or islands, $P^t = \{P_1^t, \ldots, P_m^t\}$.
while stopping criterion **not** met **do**
    for each island i **do in parallel**
        if $(t \bmod \tau) == 0$ **then**
            Send selected individuals from island P_i^t to selected neighbouring islands.
            Receive immigrants I_i^t from islands for which islands P_i^t is a neighbour.
            Replace P_i^t by a subpopulation resulting from a selection among P_i^t and I_i^t.
        end if
        Produce P_i^{t+1} by applying reproduction operators and selection to P_i^t.
    end for
    Let $t := t + 1$.
end while

Figure 3.5 sketches some of the most common topologies, common topologies include unidirectional rings (a ring with directed edges only in one direction), bidirectional rings, torus or grid graphs, hypercubes, scale-free graphs (De Felice et al., 2011), random graphs (Giacobini et al., 2005), and complete graphs. As can be seen in Figure 3.5, some of the topologies require more time to spread the information and reach a desired vertex like rings and torus graphs, hypercubes, complete graphs, and many scale-free graphs can spread the information faster due to the large amount of connected vertices.



(a) Ring (b) 2D Torus (c) Complete Graph

Figure 3.5: Sketches of common topologies.

The first theoretical result where communication between islands plays a key role was presented in Lässig and Sudholt (2010, 2013). Lässig and Sudholt showed that communication between islands is necessary to optimise the family of problems LOLZ with n, z, b and ℓ as parameters of the function. An island model running (1+1) EA with a correct setting of the migration interval and for any migration topology that is not too sparse, is able to find the

optimum of LOLZ in polynomial time with high probability. On the other hand, independent islands running (1+1) EA and the (μ+1) EA with large population require exponential time with high probability.

The function LOLZ is based on the function LEADINGONES and its symmetric counterpart LEADINGZEROS that counts the number of leading zeroes. The function is defined in such a way that the maximum contribution provided by the leading zeroes cannot exceed a limit called z, i. e., the fitness cannot be increased by adding more leading zeroes, leading to a local optimum. The algorithm is encouraged to either establish a prefix of ones or a prefix of zeroes (both with probability close to $1/2$), and as defined by z on the long run, gathering leading zeroes is a bad choice since at some point this decision becomes irreversible and it will be necessary to switch from one prefix to the other to escape from the local optimum.

The function is composed by several blocks denoted by b and each block contains ℓ bits, each block has to be optimised one by one, from left to right. Once the first block consists only of leading ones, the fitness depends on the correct decision made in the second block. Only by making all decisions correctly, the EA will be able to find the global optimum with a prefix of $b\ell$ leading ones (cf. Table 1 in Sudholt, 2018 for an example of this function).

For the case of the (μ+1) EA, it is not possible to optimise LOLZ efficiently, w. o. p. A population-based EA tends to gather leading bits of the same value; sooner or later it will make a wrong decision and get stuck in some local optimum. The same holds for independent runs, if the islands do not communicate, each island behaves like a population-based EA.

An island model with a suitable parametrisation can optimise the function LOLZ efficiently. If a set of islands have made the correct decision, the migration policy and topology can help in the dissemination of "good" individuals to other islands. This helps to maintain good individuals for a long period of time with high probability. In this case it is necessary to set the migration policy properly so that migration only transmits good individuals; if migration happens to soon it may be possible to transmit individuals with leading zeroes. For the case of the topology, it is necessary to define a topology able to spread the right information quickly. Lässig and Sudholt (2013) have defined the following desired property for a topology: a topology G is called *well-expanding* if there is a constant $\varepsilon > 0$ such that the following holds. For every subset $V' \subseteq V$ with $|V'| \leq |V|^{\varepsilon}$ we have $|N(V')| \geq (2+\varepsilon)|V'|$. Lässig and Sudholt (2013) give the following positive result by considering an island model running a (1+1) EA with a well-expanding topology and migration interval $\tau = n^{5/3}$.

Theorem 3.29 (Theorem 3 in Lässig and Sudholt, 2013)**.** *Consider an island model where each island runs a* (1+1) EA *with migration on a well-expanding migration topology with* $\tau = n^{5/3}$ *and* $\mu = \text{poly}(n)$ *subpopulations, accepting a best search point among all immigrant and the resident individual. Let the function* LOLZ *be parametrised according to* $\ell = 2\tau/n =$

$2n^{2/3}$, $z = \ell/4 = n^{2/3}/2$, *and* $b \leq n^{1/6}/16$. *If the migration counter* t *starts at* $\tau/2 = n^{5/3}/2$ *then w. o. p. the algorithm finds a global optimum within* $\mathrm{O}(b\ell n) = \mathrm{O}\left(n^2\right)$ *generations.*

The main proof idea is that each island makes its own decision for the first block (from left to right). Some may start gathering leading zeroes and some may start gathering leading ones (with high probability at least one island start gathering leading ones). Then, once a migration happens (the migration interval is tuned such that between two migrations all islands will be starting to optimise the same new block, excluding islands that got stuck on previous blocks), individuals with leading ones are strictly better than individuals with leading zeroes, so the leading ones individuals takeover the leading zeroes individuals if the topology is not too sparse. In this case there will be many islands that have made the right decision, and the individuals can continue towards the next block. The idea is that this process is repeated block after block, with "good" decisions made and communicated to other islands until some island makes it to the end of the last block and finds a global optimum with high probability.

The definition of the migration interval τ and the setting of the topology are crucial for the proper propagation of the information. If these two settings are chosen wrong, takeover may happen; if migration happens too frequently, the island model is subjected to genetic drift in the same fashion as the (μ+1) EA population and if migration is slow, promising individuals can die out depending on the maximum degree of the topology.

The island model has also been analysed for the case of the $\textsc{Jump}_k$ function by Dang et al. (2016b); their island model uses a particular topology called *single-receiver model* (Watson and Jansen, 2007). This model was introduced to prove that a constructed royal road function with a building block structure could be solved efficiently by crossover. It was also used in Neumann et al. (2011) where it was shown that crossover during migration can be effective, for constructed functions as well as for instances of the VERTEXCOVER problem. We refer the reader to Neumann et al. (2011); Watson and Jansen (2007) and Section 46.5.4 from Sudholt (2015) for details.

The topology is formed by $\mu + 1$ islands, μ of them running a (1+1) EA in parallel and the remaining island is called *receiver*. In each iteration 2 of the μ islands are selected uniformly at random, then copies of their respective best-so-far individuals are sent to the receiver, then crossover is performed and the resulting offspring replaces the resident if it has higher fitness. The run is stopped once the receiver island produces the optimum. The main idea of the analysis is similar to the ones in deterministic crowding. These islands will generate individual with 0s in different positions or reduce the number of bit positions where they have 0s in common. Then by crossover, the receiver will have good chances of creating

the optimum from the 2 migrants with the same bound as Hamming distance maximisation and population-based fitness sharing.

For the case of island models for combinatorial optimisation, Lässig and Sudholt (2014) considered island models for sorting (as maximisation of sortedness), shortest paths and the Eulerian cycle problem. The main goal for this analysis was to show how general-purpose island models with EAs perform when applied to a broad range of problems.

The sorting problem was first considered as an optimisation problem and analysed in Scharnow et al. (2005) in the context of the (1+1) EA. Given a sequence of n distinct elements from a totally ordered set, sorting is the problem of maximising the sortedness. The main goal is to find the unknown optimal permutation π_{opt} such that $\big(\pi_{\text{opt}}(1),\ldots,\pi_{\text{opt}}(n)\big)$ is the sorted sequence with respect to some unknown criterion. The search space is the set of all permutations π on $\{1,\ldots,n\}$. The fitness function $f_{\pi_{\text{opt}}}(\pi)$ describes the sortedness of $(\pi(1),\ldots,\pi(n))$ with respect to $\big(\pi_{\text{opt}}(1),\ldots,\pi_{\text{opt}}(n)\big)$. The main results of this analysis of parallel executions of the (1+1) EA, show that it is possible to achieve linear speedups when certain restrictions on the number of islands are defined and used in comparison to the standard (1+1) EA. These results also show that the bounds improve with the density of topologies.

Scharnow et al. (2005) also analysed the single source shortest path problem (SSSP) in the context of the (1+1) EA. An of the SSSP is given by an undirected connected graph with vertices $\{1,\ldots,n\}$ and a distance matrix $D = (\mathrm{d}_{ij})_{1\leq i,j\leq n}$ where $\mathrm{d}_{ij} \in \mathbb{R}_0^+ \cup \{\infty\}$ defines the length value for given edges from node i to node j. The goal is to find the shortest path from a node s to each other node $1 \leq i \leq n-1$.

A solution is represented as a *shortest paths tree*, a tree rooted at s with directed shortest paths to all other vertices. A search point x is defines as a vector of length $n-1$, where each position i describes the predecessor node x_i of node i in the shortest path tree. The results in Lässig and Sudholt (2014) show that a parallel implementation with λ islands also allows for linear speedups. The maximum number of islands guaranteed to yield linear speedups depends on the topology and the mutation operator used.

An undirected graph is called *Eulerian* if it is connected and all vertices have an even degree. Given an undirected, loopless Eulerian graph, the goal is to find and Eulerian cycle, i. e., find a traversal of the graph on which each edge is traversed exactly once. This problem can be solved in linear computation time using Hierholzer's algorithm and has already been analysed in the context of EAs (Doerr et al., 2007a; Doerr and Johannsen, 2007; Doerr et al., 2007b; Neumann, 2008).

Lässig and Sudholt (2014) showed that if the migration policy is set large enough, parallelisation can help to make the right decision through independent evolution. The

authors showed that island models can lead to a superlinear speedup on problems from combinatorial optimisation but this good performance can only be achieved if migration is rarely used, or if independent runs are used. On the contrary, if the migration interval is set too small (this include the number of islands and the topology), the island model rapidly loses diversity. Only strictly better immigrants are considered for inclusion, with all the islands performing independent random walks, then there is a constant probability that the some island propagates its solution throughout the whole island model, before any other island can make an improvement. This example shows that the choice of the migration interval can make a difference between exponential and logarithmic speedups.

Finally, for the case of island model on DOPs we can find the work done by Lissovoi and Witt (2017) on the MAZE function. This function was introduced by Kötzing and Molter (2012) and the goal is to track the optimum while this changes in phases of t steps. The authors showed that the (1+1) EA is not able to keep track of the optimum and requires with high probability an exponential amount of time to find the optimum. Lissovoi and Witt (2017) analysed this function in the context of parallel EAs. By using an island model with communication occurring within regular intervals, islands allow efficient tracking of the optimum of the MAZE function. We refer to Kötzing and Molter (2012); Lissovoi and Witt (2017) for more details and formal definition of MAZE.

3.5 Diversity for Multi-Objective Optimisation

When dealing with large Pareto fronts, MOEAs try to spread the individuals in the population over the whole Pareto front with a set of solutions representing the different trade-offs with respect to the given objective functions. The number of these trade-offs can be exponential with regard to the problem size, which implies that not all trade-offs can be computed efficiently. In this case, one may be interested in good approximations of the Pareto front consisting of a not too large set of Pareto-optimal solutions (see Definition 2.15). The application of a wide range of diversity mechanism can help to achieve this goal (Fonseca and Fleming, 1995). We refer to Brockhoff's survey (Brockhoff, 2012) for a review of further theoretical results.

3.5.1 Diversity for Approximating Pareto-Optimal Sets

A popular diversity strategy is to use some density estimator to favour individuals in less crowded regions of the objective space (Laumanns et al., 2001). A well-known diversity

strategy is the δ-dominance approach: partitioning the objective space into boxes and restricting the population to at most one individual per box (Laumanns et al., 2002).

Horoba and Neumann (2010) focussed on how such diversity mechanisms influence the approximation ability of MOEAs, specifically on GSEMO. Here, we present the results from Horoba and Neumann (2010) that show the usefulness of such diversity mechanisms. They point out for each diversity mechanism a typical situation, which explains when and how the considered diversity mechanism is crucial to obtain a good approximation of the Pareto front of the given problem.

When the number of Pareto-optimal objective vectors grows exponentially with the problem size, it is not possible to obtain the whole front efficiently. In this case, one may be interested in the time to obtain a good approximation of the Pareto front and to examine in which situations the use of a diversity mechanism can help to achieve this goal.

To judge the quality of an approximation, Horoba and Neumann (2010) use the additive ε-dominance measure (see Laumanns et al., 2002). A set of objective vectors T (or a set of corresponding search points) is called an *ε-approximation* of f if and only if, there is for each objective vector $v \in f(\{0,1\}^n)$ at least one objective vector $u \in T$ that ε-dominates v. If it is just required an approximation of the Pareto front, it might be beneficial to avoid storing similar individuals in the population of GSEMO. The objective search space is divided into boxes and each box stores at most one individual.

An individual x is mapped to a box index vector $b(x) = (b_1(x), \cdots, b_m(x))$ with $b_i(x) := \lfloor f_i(x)/\delta \rfloor$ where $\delta \in \mathbb{R}^+$ determines the size of the boxes. This idea is incorporated into the plain GSEMO resulting into the Global Diversity Evolutionary Multi-Objective Optimiser (GDEMO) (see Algorithm 21). The population of GDEMO constitutes an δ-approximation of the so far sampled decision vectors, the dominance with respect to the box index vector indices δ-dominance $b(x) \succeq b(y)$ then $x \succeq_\delta y$ (see Horoba and Neumann, 2010 for a detailed definition).

Algorithm 21 GDEMO

1: Choose an initial solution $s \in \{0,1\}^n$ uniformly at random.
2: Determine $f(s)$ and initialize $P := \{s\}$.
3: **while** stopping criterion **not** met **do**
4: Choose s uniformly at random from P.
5: Define s' by flipping each bit in s independently with probability $1/n$.
6: **if** s' is **not** dominated **or** ε-approximation dominated by any individual in P **then**
7: Add s' to P, and remove all individuals weakly ε-approximation dominated by s' from P.
8: **end if**
9: **end while**

Also, Horoba and Neumann (2010) investigate a simplified version of the algorithm SPEA2 (Zitzler et al., 2001), which relies on a so-called density estimator. Let Q be a given set of search points. The $\text{rank}_Q(x)$ of a search point $x \in Q$ is given by the number of search points in Q that dominate x, i. e., $\text{rank}_Q(x) := |\{y \in Q \mid y \succ x\}|$. Additionally, a metric on the objective space is taken into account. In Horoba and Neumann (2010) the maximum metric $\text{d}(u,v) := \max_{i\in\{1,\dots,m\}}|u_i - v_i|$ where u and v are objective vectors is used. Let $\text{d}_Q(x) := (\text{d}_Q^0(x), \dots, \text{d}_Q^{|Q|-1}(x))$ where $\text{d}_Q^k(x)$ denotes the distance $\text{d}(f(x), f(y))$ from $x \in Q$ to its k-th nearest neighbour $y \in Q$ with respect to d. The archive truncation selects a search point $x \in Q$ with the lowest $\text{d}_Q(x)$ value with respect to the lexicographic order from the search points with the highest $\text{rank}_Q(x)$ value for removal (Algorithm 22).

Algorithm 22 Selection for Removal

Require: Set of search points Q.
Ensure: Search point $z \in Q$.
1: Set $Q' = \arg\max_{x\in Q} \text{rank}_Q(x)$.
2: Set $Q'' = \arg\min_{x\in Q'} \text{d}_Q(x)$.
3: **return** $z \in Q''$ chosen uniformly at random.

Using this procedure, the simplified version of the SPEA2 is named Rank- And Distance-based Evolutionary Multi-Objective Optimiser (RADEMO) (see Algorithm 23). Now the goal for any algorithmic approach is to achieve an additive ε-approximation of a given problem where $\varepsilon \in \mathbb{R}^+$. All the problems examined depend on a parameter ε and the goal is to examine whether the algorithm is able to achieve an ε-approximation of the Pareto-optimal set in polynomial time.

Algorithm 23 RADEMO

1: Initialise P with μ individuals chosen uniformly at random.
2: **while** stopping criterion **not** met **do**
3: Choose s uniformly at random from P.
4: Define s' by flipping each bit in s independently with probability $1/n$.
5: Choose an individual $z \in P \cup s'$ for removal using Algorithm 22
6: Let $P = P \setminus \{z\} \cup \{s'\}$.
7: **end while**

3.5.1.1 Large Pareto Fronts

Horoba and Neumann (2010) discuss how diversity mechanisms can be helpful to achieve an ε-approximation of an exponentiallly large Pareto front. The authors considered the bi-objective example function LF_ε (see Section 3 from Horoba and Neumann, 2010 for a formal

definition and for a sketch of the function), where both objectives behave complementary. The results show that both diversity mechanisms may help to achieve a good approximation of an exponentially large Pareto front.

For the case of the plain GSEMO on LF_ε, the algorithm produces many Pareto-optimal objective vectors with roughly $n/4$ 1-bits in the first half of the bitstring. However, to achieve an ε-approximation it is necessary that for each i, $0 \leq i \leq n/2$, a solution with i 1-bits in the first half on the bitstring is obtained. This implies that at least $n/2+1$ search points are necessary to achieve an ε-approximation.

Theorem 3.30 (Theorem 1 in Horoba and Neumann, 2010)**.** *The time until GSEMO has reached an ε-approximation of* LF_ε *is* $2^{\Omega\left(n^{1/4}\right)}$ *with probability* $1-2^{-\Omega\left(n^{1/4}\right)}$.

With GDEMO, an ε-approximation can be obtained efficiently when choosing δ (the size of the boxes) value properly. This has the effect that the algorithm keeps for each fixed 1-bit in the first half exactly one individual in the population. On the contrary, if δ is defined small, GDEMO becomes GSEMO which shows that the right choice of δ is crucial for dealing with large Pareto fronts.

Theorem 3.31 (Theorem 2 in Horoba and Neumann, 2010)**.** *Choosing $\delta = \varepsilon$ as box size, the algorithm GDEMO achieves an ε-approximation of* LF_ε *in expected time* $\mathrm{O}\left(n^2 \log n\right)$.

For RADEMO, the density estimator ensures in a natural way a spread over the Pareto front of LF_ε. To achieve an ε-approximation of LF_ε, $(n/2)+1$ points are necessary. Therefore, with a population size of at least $n/2+1$, RADEMO constructs an ε-approximation efficiently.

Theorem 3.32 (Theorem 3 in Horoba and Neumann, 2010)**.** *Choosing $\mu \geq n/2+1$ as population size, the algorithm RADEMO achieves an ε-approximation of* LF_ε *in expected time* $\mathrm{O}(\mu n \log n)$.

3.5.1.2 On the Choice of δ

From the previous results, it is observed that the behaviour of GDEMO varies with respect to δ. Horoba and Neumann (2010) then analyse the right choice of δ and the ability of GDEMO to achieve a good approximation by using the bi-objective function small front SF_ε (see Section 4.1 from Horoba and Neumann, 2010 for a formal definition and for a sketch of the function).

For the case of GSEMO on SF_ε, the algorithm is able to compute the Pareto front efficiently. The proof idea is simply based on the number of b mutations needed to create an individual on the Pareto front.

Theorem 3.33 (Theorem 6 in Horoba and Neumann, 2010)**.** *The expected time until GSEMO has computed the Pareto front of* SF_ε *is* $\mathrm{O}\left(n^2 \log n\right)$.

The proof idea for GDEMO on SF_ε is based on the phenomenon that an individual x with $|x|_1 < n/2$ ($|x|_1 > n/2$) tends to have more (less) 1-bits inducing a drift towards the middle section of the front. Since GDEMO is limited to at most one individual per box, it takes a long time to reach the outskirts of the Pareto front, which are necessary for an ε-approximation (Horoba and Neumann, 2008).

Theorem 3.34 (Theorem 4 in Horoba and Neumann, 2010)**.** *Choosing* $\delta \geq \varepsilon$ *as box size, the time until the algorithm GDEMO has achieved an* ε*-approximation of* SF_ε *is* $2^{\Omega(n)}$ *with probability* $1 - 2^{-\Omega(n)}$.

For the middle part of the Pareto front of SF_ε it holds that all distances between neighbouring objective vectors are equal. In addition, the objective vectors corresponding to the search points 0^n and 1^n have a large distance to all other objective vectors. This helps the algorithm RADEMO to achieve an ε-approximation of SF_ε as the density estimator enforces the algorithm to produce solutions that have a large distance in the objective space. The next theorem shows that RADEMO obtains an ε-approximation efficiently if the population size is at least 2.

Theorem 3.35 (Theorem 5 in Horoba and Neumann, 2010)**.** *Choosing* $\mu \geq 2$ *as population size, the algorithm RADEMO has achieved an* ε*-approximation of in expected time* $\mathrm{O}(\mu n \log n)$.

3.5.1.3 Distance Measure of the Density Estimator

Now, the size of the population is analysed for the case of RADEMO; if the size of the population is not large enough, the optimisation time can be affected negatively. For this case the bi-objective example function TF_ε is analysed (see Section 4.2 from Horoba and Neumann (2010) for a formal definition and for a sketch of the function).

GSEMO with quite small Pareto front of TF_ε can be efficiently computed. Again, the proof consists of the time needed to reach the Pareto front (in expected optimisation time $\mathrm{O}\left(n^3\right)$). After the first Pareto-optimal individual is added to the population, it takes into consideration the time until the last Pareto-optimal search point is found in exptected optimisation time $\mathrm{O}\left(n^3\right)$.

Theorem 3.36 (Theorem 9 in Horoba and Neumann, 2010)**.** *The expected time until the GSEMO has computed the Pareto front of* TF_ε *is* $\mathrm{O}\left(n^3\right)$.

With GDEMO, if there is no solution of 1^i0^{n-i} in the population, the population size is 1 and the algorithm maximises the number of 0-bits. No steps increasing the number of 0-bits are accepted in this case as such search points are dominated by the current one constituting the population. This implies that after an expected number of $O(n \log n)$ steps the population consists of an individual from 1^i0^{n-i}. Afterwards, the individual starts a random walk on 1^i0^{n-i}. The population has converged to an ε-approximation if an individual of the second box has been obtained. This happens after an expected number of $O(n^3)$ steps (Jansen and Wegener, 2001).

Theorem 3.37 (Theorem 8 in Horoba and Neumann, 2010)**.** *Choosing $\delta = \varepsilon$ as box size, the algorithm GDEMO has achieved an ε-approximation of* TF_ε *in expected time* $O(n^3)$.

The next theorem shows that RADEMO does not achieve an ε-approximation of TF_ε within polynomial time w. h. p. if the size of the population is not too large. The main idea of the proof is that the individuals spread out over $\{(\varepsilon/4 + i \cdot 2\varepsilon/n, \varepsilon/4 - i \cdot \varepsilon/n) \mid 0 \leq i \leq n/4\}$ in an almost equally spaced manner before the Pareto front is reached. Thereafter RADEMO's diversity mechanism prevents the algorithm from spreading out on the Pareto front. Hence, RADEMO does not obtain the objective vectors in the top left part of the Pareto front, which are necessary to achieve an ε-approximation.

Theorem 3.38 (Theorem 7 in Horoba and Neumann, 2010)**.** *Choosing $2 \leq \mu = O\left(n^{1/3-c}\right)$ as population size where $0 \leq c \leq 1/3$ is a constant, the time until the algorithm RADEMO has achieved an ε-approximation of* TF_ε *is $2^{\Omega(n^c)}$ with probability $1 - 2^{-\Omega(n^c)}$.*

We conclude this section by providing an overview of the results mentioned previously. Table 3.3 shows how the different diversity strategies used by the MOEAs can help to achieve a good approximation of the Pareto-optimal set.

Table 3.3: Overview over the main results from Horoba and Neumann (2010). All exponential times hold w. o. p. The bounds come with restrictions on the population size μ and the size of the boxes δ.

Function	**GSEMO**	**GDEMO**	**RADEMO**
LF_ε	$2^{\Omega(n^{1/2})}$	$O(n^2 \log n)$	$O(\mu n \log n)$
SF_ε	$O(n^2 \log n)$	$2^{\Omega(n)}$	$O(\mu n \log n)$
TF_ε	$O(n^3)$	$O(n^3)$	$2^{\Omega(n^c)}$

3.6 Conclusions

In this chapter we have shown many theoretical results where the performance of an EA can be enhanced by promoting diversity into the population by means of explicit and/or implicit diversity mechanisms in different kind of ploblems (global exploration in static and dynamics optimisation problems, and multi-objective optimisation problems). As mentioned at the beginning of the chapter, balancing between diversity and intensification is a very complicated and important task. Surveys on diversity mechanisms (Blum and Roli, 2003; Črepinšek et al., 2013; Liu et al., 2009; Shir, 2012; Squillero and Tonda, 2016; Sudholt, 2018; Talbi, 2002) reveal a multitude of approaches to enhance and promote diversity, yet it is often unclear which of these mechanisms perform well, and why.

The explicit mechanisms reviewed here range from simple ones like avoiding genotype of phenotypes (when used as a tie breaking rule or selection for survival), dividing the population into islands or niching mechanisms like deterministic crowding or fitness sharing (in 3 different variants), ageing and many others.

All the mechanisms reviewed here have been compared with its plain version (when available) and we have seen that diversity can be beneficial for enhancing the global exploration capabilities of EAs. In many cases diversity mechanisms can be highly effective for the considered problems, speeding up the expected or typical optimisation time by constant factors, polynomial factors, or even exponential factors. Nevertheless, results from the example functions reviewed here, have shown that some mechanisms that work for one function may not work on other functions, and vice versa.

Another interesting result shown by these results is that crossover can make use of a diverse population to work effectively. This diversity can be provided by the methods mentioned before or emerge naturally through independent mutations (this includes different mutation rates). Finally, all these theoretical results provide a detailed description of how these diversity mechanisms work well (or not) and why these mechanisms are able to overcome certain problems while their plain versions struggle, or are not able to find satisfactory solutions to the analysed example problems.

Part II

Runtime Analysis of Diversity Mechanisms on Multimodal Optimisation

Chapter 4

Runtime Analysis of Niching Mechanisms on TwoMax

This chapter is based on the following publications:

1. Covantes Osuna, E. and Sudholt, D. (2017). Analysis of the Clearing Diversity-Preserving Mechanism. In *Proceedings of the 14th ACM/SIGEVO Conference on Foundations of Genetic Algorithms*, FOGA '17, pages 55–63. ACM.

2. Covantes Osuna, E. and Sudholt, D. (2018b). On the Runtime Analysis of the Clearing Diversity-Preserving Mechanism. *Evolutionary Computation*. To appear. Preprint available from http://arxiv.org/abs/1803.09715.

3. Covantes Osuna, E. and Sudholt, D. (2018c). Runtime Analysis of Probabilistic Crowding and Restricted Tournament Selection for Bimodal Optimisation. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '18, pages 929–936. ACM. **Nominated for a best paper award in the track 'Genetic Algorithms'**.

One particular way for diversity maintenance are niching methods, based on the mechanics of natural ecosystems (Shir, 2012). A niche can be viewed as a subspace in the environment that can support different types of life. A specie is defined as a group of individuals with similar features capable of interbreeding among themselves but that are unable to breed with individuals outside their group. Species can be defined as similar individuals of a specific niche in terms of similarity metrics. In EAs the term niche is used for the search space domain, and species for the set of individuals with similar characteristics.

Niching methods have been developed to reduce the effect of genetic drift resulting from the selection operator in standard EAs, to maintain the population diversity, and allow the EA to investigate many peaks in parallel, thus avoiding getting trapped in local optima (Sareni

and Krahenbuhl, 1998). Premature convergence is one of the major difficulties in evolutionary algorithms, the population converging to a sub-optimal individual before the fitness landscape is explored properly. Real optimisation problems often lead to multimodal domains and so require the identification of multiple optima, either local or global (Sareni and Krahenbuhl, 1998; Singh and Deb, 2006). In multimodal optimisation problems, there exist many attractors for which finding a global optimum can become a challenge to any optimisation algorithm. A diverse population can deal with these multimodal problems as it can explore several hills in the fitness landscape simultaneously.

Niching methods often try to promote diversity by modifying the selection process of individuals, taking into account not only the value of the fitness function but also the distribution of individuals in the space of genotypes or phenotypes (Glibovets and Gulayeva, 2013). Many niching techniques have been introduced to form and maintain multiple, diverse, final solutions for an exponential to infinite time period with respect to population size, whether these solutions are of identical fitness or of varying fitness (Črepinšek et al., 2013; Glibovets and Gulayeva, 2013; Shir, 2012; Squillero and Tonda, 2016). Given such a variety of mechanisms to choose from, it is often not clear which mechanism is the best choice for a particular problem.

Most of the analyses and comparisons made between niching methods are assessed by means of empirical investigations using benchmark functions (Sareni and Krahenbuhl, 1998; Singh and Deb, 2006). Theoretical runtime analyses have been performed that rigorously quantify the expected time needed to find one or several global optima (Covantes Osuna and Sudholt, 2017, 2018b,c; Oliveto et al., 2014). Both approaches are important to understand how these mechanisms impact the EA runtime and if they enhance the search for good individuals. These different expectations imply where EAs and which niching mechanism should be used and, perhaps even more importantly, where they should not be used.

Previous theoretical studies (Covantes Osuna and Sudholt, 2017, 2018b,c; Friedrich et al., 2009; Oliveto et al., 2014) compared the expected runtime of different diversity mechanisms when embedded in a simple baseline EA, the (μ+1) EA (see Algorithm 5). All mechanisms were considered on the well-known bimodal function TwoMax (see Definition 2.11)[1].

TwoMax was chosen because it is simply structured, hence facilitating a theoretical analysis, and it is hard for EAs to find both optima as they have the maximum possible Hamming distance. The results allowed for a fair comparison across a wide range diversity

[1]In Friedrich et al. (2009) an additional fitness value for 1^n was added to distinguish between a local optimum 0^n and a unique global optimum. There the goal was to find the global optimum, and all approaches had a baseline probability of $1/2$ of climbing up the right branch by chance. We use the same approach as Covantes Osuna and Sudholt (2017, 2018b,c); Oliveto et al. (2014), and consider the original definition of TwoMax and the goal of finding both global optima. The discussion and presentation of previous work from Friedrich et al. (2009) is adapted to our setting. We refer to Sudholt (2018) for details.

mechanisms, revealing that some mechanisms like fitness diversity or avoiding genotype duplicates, perform badly, while other mechanisms like fitness sharing, clearing or deterministic crowding perform surprisingly well. Table 4.1 summarises previous work for the (μ+1) EA with diversity mechanisms on TwoMax (details can be found in Section 3.2). Some mechanisms succeed in finding both optima on TwoMax efficiently, that is, in (expected) time $O(\mu n \log n)$. Others have a very low success probability.

Table 4.1: Overview of runtime analyses for the (μ+1) EA with different diversity mechanisms on TwoMax. The success probability is the probability of finding both optima within (expected) time $O(\mu n \log n)$. Conditions include restrictions on the population size μ, the sharing/clearing radius σ, the niche capacity κ, window size w, and $\mu' := \min(\mu, \log n)$. Results adapted from Covantes Osuna and Sudholt (2018c).

	Diversity Mechanism	**Success prob.**	**Conditions**
PL	Plain (μ+1) EA	$o(1)$	$\mu = o(n/\log n)$
NGD	No Genotype Duplicates	$o(1)$	$\mu = o(\sqrt{n})$
NFD	No Fitness Duplicates	$o(1)$	$\mu = \text{poly}(n)$
PC	Probabilistic Crowding	$2^{-\Omega(n)}$	all μ
DC	Deterministic Crowding	$1-2^{-\mu+1}$	all μ
RTS	Restricted Tournament Selection	$\geq 1-2^{-\mu'+3}$	$w \geq 2.5\mu \ln n$
PFS	Population-based Fitness Sharing ($\sigma = n/2$)[1]	1	$\mu \geq 2$
FS	Individual-based Fitness Sharing ($\sigma = n/2$)[1]	1	$\mu \geq 3$
CL	Clearing ($\sigma = n/2$)	1	$\mu \geq \kappa n^2$

[1] Fitness sharing uses phenotypic sharing based on the number of ones.

We contribute to this line of work by studying the performance of three classical niching mechanisms, *probabilistic crowding*, *restricted tournament selection* and *clearing*. The three methods are well-known techniques as covered in tutorials and surveys for diversity-preserving mechanisms (Črepinšek et al., 2013; Glibovets and Gulayeva, 2013; Shir, 2012; Squillero and Tonda, 2016) and compared in empirical investigations (Sareni and Krahenbuhl, 1998; Singh and Deb, 2006). However, we are lacking a good understanding of when and why they perform well and how they compare to diversity mechanisms analysed previously.

In *probabilistic crowding*, the offspring compete against their most similar parent and the survivor is chosen with a probability proportional to their fitness. The idea is to use a low selection pressure to prevent the loss of niches of lower fitness (Mengsheol and Goldberg, 1999). Probabilistic crowding has been used for multimodal optimisation (Ballester and Carter, 2004; Mengsheol and Goldberg, 1999; Mengshoel et al., 2014; Mengshoel and Goldberg, 2008), in its plain form as well as in variants in which a scaling factor has been introduced into the replacement policy.

Restricted tournament selection (RTS) is a modification of the classical tournament selection for multimodal optimisation that exhibits niching capabilities. RTS selects two elements from the population uniformly at random (u. a. r.) to undergo recombination and mutation to produce two new offspring. The offspring compete with their closest individual from w (*window size*) more individuals selected u. a. r. from the population, and the best individual is selected. This form of tournament restricts an entering individual from competing with others too different from it (Harik, 1995). RTS has been analysed empirically for the classical comparison between crowding mechanisms for multimodal optimisation as a replacement strategy (García-Martínez et al., 2012; Qu and Suganthan, 2010). Recent applications for engineering problems with multimodal domains include facility layout design (García-Hernández et al., 2015) and the design of product lines (Tsafarakis, 2016) with reported better results compared to the other variants without RTS.

Clearing preserves the fitness of the best individuals in each niche (called *winners*), while resetting the fitness of all the other individuals of the same niche to the minimum fitness value possible (Pétrowski, 1996) and has been analysed empirically for both of its existing variants (Sareni and Krahenbuhl, 1998; Singh and Deb, 2006).

Our contribution is to provide a rigorous theoretical runtime analysis for the three niching mechanisms in the context of the $(\mu+1)$ EA on TwoMax, to rigorously assess their performance in comparison to other diversity mechanisms. In addition, our goal is to provide insights into the working principles of these mechanisms to enhance our understanding of their strengths and weaknesses.

For the $(\mu+1)$ EA with probabilistic crowding, we show in Section 4.1 that the mechanism is unable to evolve solutions of significantly higher fitness than that obtained during initialisation (or, equivalently, through random search), even when given exponential time. The reason is that fitness-proportional selection between parent and offspring results in an almost uniform choice as both have very similar fitness, hence fitness-proportional selection degrades to uniform selection for replacement. For the $(\mu+1)$ EA with restricted tournament selection, we show in Section 4.2 that the mechanism succeeds in finding both optima of TwoMax in the same way as deterministic crowding, provided that the window size w is chosen large enough. However, if the window size is too small then it cannot prevent one branch taking over the other, leading to exponential runtimes with high probability. In Section 4.3 we show that, for the case of small niches, the $(\mu+1)$ EA with clearing can optimise all functions of unitation when the distance function and parameters like the clearing radius σ, the niche capacity κ (how many winners a niche can support) and μ are chosen appropriately. In the case of large niches, that is, with a clearing radius of $\sigma = n/2$, it is able to find both optima of TwoMax.

Before we start with the runtime analysis for the three niching mechanisms, let us show the following time bound, which assumes that the (μ+1) EA never decreases the best fitness on a considered branch of TwoMax. This bound defines the expected time until an optimum is found when an individual is initialised on one branch. This bound is used in the following theoretical analysis, we will show in the proof of Theorem 4.4 that this assumption is met with high probability. We also use this time bound for experimental analysis as a stopping criterion.

Lemma 4.1. *Consider one branch of* TwoMax *and a* (μ+1) EA *with a replacement selection where the best fitness of all individuals on this branch never decreases. If the* (μ+1) EA *is initialised with at least one individual on the branch then the optimum of the branch is found within time* $2e\mu n \ln n$ *with probability at least* $1-1/n$ *and in expectation.*

Proof. We apply the multiplicative drift theorem with tail bounds (Theorem 2.31) to random variables X_t that describe the Hamming distance of the closest individual to the targeted optimum. Note that $X_0 \leq n/2$ as we start with an individual on the considered branch and the optimum has been found once $X_t = 0$.

The probability of selecting an individual with Hamming distance X_t is at least $1/\mu$. In order to create a better individual, it is sufficient that one of the X_t differing bits is flipped and the other bits remain unchanged. Each bit flip has a probability of being mutated of $1/n$ and the remaining bits remain unchanged with probability $(1-1/n)^{n-1}$. Hence, the probability of creating an individual with a smaller Hamming distance is bounded as follows:

$$\text{Prob}(X_{t+1} < X_t \mid X_t) \geq \frac{1}{\mu} \cdot \frac{X_t}{n} \cdot \left(1-\frac{1}{n}\right)^{n-1} \geq \frac{X_t}{\mu e n}.$$

This implies

$$\text{E}[X_{t+1} \mid X_t] \leq \left(1-\frac{1}{e\mu n}\right) X_t.$$

Applying the multiplicative drift theorem yields that the time till the optimum is found is at most $e\mu n \cdot (\ln(n/2)+\ln n) \leq 2e\mu n \ln n$ with probability at most $1/n$ and in expectation. □

4.1 Probabilistic Crowding

We start by presenting the (μ+1) EA using probabilistic crowding in the same fashion as deterministic crowding in Friedrich et al. (2009). Recall that in probabilistic crowding, the offspring compete against the most similar parent according to a distance metric and the survivor wins proportionally according to their fitness. Without crossover, this means that

the mutant y competes against its parent x using fitness-proportional selection. Then the probability of the mutant y winning is given by $\frac{f(y)}{f(x)+f(y)}$, where f is the fitness function. The resulting (μ+1) EA is shown in Algorithm 24.

Algorithm 24 (μ+1) EA with probabilistic crowding

1: Let $t := 0$ and initialise P_0 with μ individuals chosen uniformly at random.
2: **while** stopping criterion **not** met **do**
3: Choose $x \in P_t$ uniformly at random.
4: Create y by flipping each bit in x independently with probability $1/n$.
5: Choose $r \in [0,1]$ uniformly at random.
6: **if** $r \leq \frac{f(y)}{f(y)+f(x)}$ **then** $P_{t+1} = P_t \setminus \{x\} \cup \{y\}$ **else** $P_{t+1} = P_t$ **end if**
7: Let $t := t+1$.
8: **end while**

There are several related theoretical analyses for fitness-proportional selection for the case of the OneMax function. The *Simple Genetic Algorithm (SGA)* has been analysed with fitness-proportional selection for parent selection (Neumann et al., 2009; Oliveto et al., 2014; Oliveto and Witt, 2015).

Most relevant to this work is the work by Happ et al. (2008), who analysed a variant of the (1+1) EA using fitness-proportional selection and showed that it needs exponential time to evolve a fitness of at least $(1+\varepsilon)n/2$ on OneMax with high probability. Their algorithm can be seen as a special case of the (μ+1) EA with probabilistic crowding for $\mu = 1$. Our result is similar to the result in Happ et al. (2008), but it holds for arbitrary population sizes μ and it applies to both OneMax and TwoMax. The proof uses more modern techniques from drift analysis (Oliveto and Witt, 2011) that were not available to Happ et al. (2008). In the following lemma we define in expectation how the individual accepted for replacement moves away from the individual selected for mutation.

Lemma 4.2. *Let x be the selected parent, y be the offspring, and $z \in \{x,y\}$ be the individual selected for survival. For $f =$ OneMax if $f(x) \geq (1+\delta)n/2$ for some positive $\delta = \delta(n)$ that may depend on n then*

$$\mathrm{E}[f(z) - f(x) \mid x] \leq -\frac{\delta}{2} + \Theta\left(\frac{1}{n}\right).$$

The statement also holds for TwoMax *if $f(x) \geq n/2 + \log n$.*

Proof. We first analyse the expected fitness of the mutant y before survival selection. Compared to its parent x, in expectation at least $(1+\delta)n/2 \cdot 1/n = (1+\delta)/2$ bits flip from 1 to

0, and at most $(1-\delta)n/2 \cdot 1/n = (1-\delta)/2$ bits flip from 0 to 1. Hence

$$\mathrm{E}[f(y) - f(x) \mid x] \leq (1-\delta)/2 - (1+\delta)/2 = -\delta. \tag{4.1}$$

We now use this inequality to analyse the fitness difference $f(z) - f(x)$ after survival selection. Observe that this difference is 0 in case $z = x$. Hence only generations where y is selected for survival contribute to $\mathrm{E}[f(z) - f(x) \mid x]$. The latter can be written as follows.

$$\mathrm{E}[f(z) - f(x) \mid x] = \sum_{d=-\infty}^{\infty} \mathrm{Prob}(f(y) - f(x) = d \mid x) \cdot d \cdot \frac{f(y)}{f(x)+f(y)}$$

Using that with $d = f(y) - f(x)$,

$$\frac{f(y)}{f(x)+f(y)} = \frac{(f(x)+f(y))/2 + d/2}{f(x)+f(y)} = \frac{1}{2} + \frac{d/2}{f(x)+f(y)} = \frac{1}{2} + \Theta\left(\frac{d}{n}\right),$$

we get

$$\begin{aligned}
\mathrm{E}[f(z) - f(x) \mid x] &= \sum_{d=-\infty}^{\infty} \mathrm{Prob}(f(y) - f(x) = d \mid x) \cdot d \cdot \left(\frac{1}{2} + \Theta\left(\frac{d}{n}\right)\right) \\
&= \frac{1}{2} \sum_{d=-\infty}^{\infty} \mathrm{Prob}(f(y) - f(x) = d \mid x) \cdot d \\
&\qquad + \Theta\left(\frac{1}{n}\right) \sum_{d=-\infty}^{\infty} \mathrm{Prob}(f(y) - f(x) = d \mid x) \cdot d^2.
\end{aligned}$$

The first sum is $\mathrm{E}[f(y) - f(x)]/2$ by definition of the expectation, and we already know from (4.1) that $\mathrm{E}[f(y) - f(x)]/2 \leq -\delta/2$. The second sum is at most $\mathrm{Prob}(f(y) - f(x) = d \mid x) \leq 1/(|d|!)$ as it is necessary to flip at least $|d|$ bits, which has probability at most $\binom{n}{|d|}(1/n)^{|d|} \leq 1/(|d|!)$. Thus

$$\begin{aligned}
\mathrm{E}[f(z) - f(x) \mid x] &\leq -\frac{\delta}{2} + \Theta\left(\frac{1}{n}\right) \sum_{d=-\infty}^{\infty} \frac{1}{|d|!} \cdot d^2 \\
&\leq -\frac{\delta}{2} + \Theta\left(\frac{1}{n}\right) \cdot 2 \sum_{d=1}^{\infty} \frac{1}{d!} \cdot d^2 \\
&= -\frac{\delta}{2} + \Theta\left(\frac{1}{n}\right)
\end{aligned}$$

as $\sum_{d=1}^{\infty} \frac{1}{d!} \cdot d^2 = \sum_{d=1}^{\infty} \frac{d}{(d-1)!} = \sum_{d=0}^{\infty} \frac{d+1}{d!} = \sum_{d=0}^{\infty} \frac{d}{d!} + \sum_{d=0}^{\infty} \frac{1}{d!} = \sum_{d=1}^{\infty} \frac{1}{(d-1)!} + \sum_{d=0}^{\infty} \frac{1}{d!} = 2\sum_{d=0}^{\infty} \frac{1}{d!} = 2e$.

The statement also holds for TWOMAX if $f(x) \geq n/2 + \log n$ as the algorithm only ever notices a difference to ONEMAX in case at least $\log n$ bits flip in one mutation. Since this only occurs with probability at most $1/(\log n)! = n^{-\Omega(\log\log n)}$, and the fitness difference between ONEMAX and TWOMAX is at most $n/2$, this only accounts for an additive error term of $n/2 \cdot n^{-\Omega(\log\log n)} = n^{-\Omega(\log\log n)}$ in the expectation for ONEMAX, and this error term is absorbed in the $\Theta(1/n)$ term. □

Lemma 4.2 gives an important lesson. Assume that the survivalist z was chosen uniformly between x and y, then we would have

$$\mathrm{E}[f(z) - f(x) \mid x] \leq \frac{1}{2} \cdot \mathrm{E}[f(y) - f(x) \mid x] + \frac{1}{2} \cdot \mathrm{E}[f(x) - f(x) \mid x] = -\frac{\delta}{2}$$

using (4.1) and $\mathrm{E}[f(x) - f(x) \mid x] = 0$. Lemma 4.2 states that compared to this setting, a fitness-proportional selection of z only gives a vanishing bias of $\Theta(1/n)$. In other words, Lemma 4.2 quantifies the observation that in the considered context, fitness-proportional selection is very similar to uniform selection. We now use Lemma 4.2 to prove a strong negative result on the performance of the (μ+1) EA with probabilistic crowding. To this end, we will use the negative drift theorem (Oliveto and Witt, 2011, 2012, also called *simplified drift theorem*, see Theorem 2.35).

Note that the expected ONEMAX value of a search point chosen uniformly at random is $n/2$. We also show in Section 4.2.1 that the expected TWOMAX value of a uniform random search point is $n/2 \pm \Theta(\sqrt{n})$. These values also represent equilibrium states for sequences of mutations in the absence of selection. The following theorem shows that the (μ+1) EA with probabilistic crowding does not evolve any solutions of significantly higher fitness than these values, even given exponential time.

Theorem 4.3. *With probability* $1 - 2^{-\Omega(n)}$ *the* (μ+1) EA *with probabilistic crowding on either* $f = \text{ONEMAX}$ *or* $f = \text{TWOMAX}$ *will not have found a search point with fitness at least* $(1+\varepsilon)n/2$ *in* 2^{cn} *function evaluations, for every population size* μ*, every constant* $\varepsilon > 0$ *and a small enough constant* $c > 0$ *that may depend on* ε*.*

Proof. We assume that $f = \text{ONEMAX}$ as TWOMAX can be handled in the same way. We may also assume that $\mu = 2^{o(n)}$ as if $\mu \geq 2^{c'n}$ for any constant $0 < c' < 1$, the statement follows immediately (for $c := c'$) as the first $2^{c'n}$ search points contain an optimal search point only with probability at most $2 \cdot 2^{-n} \cdot 2^{c'n} = 2^{-\Omega(n)}$ as $c' < 1$. Note that in the absence of crossover, probabilistic crowding evolves μ independent lineages as any offspring only competes directly with its parent. We show that the probability of any fixed lineage reaching a fitness of at least $(1+\varepsilon)n/2$ in 2^{cn} generations is $2^{-\Omega(n)}$. Taking the union bound over

all lineages yields that the probability of reaching such a fitness is bounded by $\mu \cdot 2^{-\Omega(n)} = 2^{o(n)} \cdot 2^{-\Omega(n)} = 2^{-\Omega(n)}$, which implies the claim.

Now focus on one lineage. By standard Chernoff bounds (see Lemma A.15), the probability of initialising the lineage with an initial search point of fitness at least $(1+\varepsilon/2)n/2$ is $2^{-\Omega(n)}$. If this rare failure event does not happen, the lineage needs to increase an initial fitness from a value at most $(1+\varepsilon/2)n/2$ to a value at least $(1+\varepsilon)n/2$ in order to achieve a fitness of $(1+\varepsilon)n/2$. We apply the negative drift theorem to the fitness of the current individual in our lineage to show that this does not happen in 2^{cn} generations with probability $1-2^{-\Omega(n)}$. The interval chosen will be from $a := (1+\varepsilon/2)n/2$ to $b := (1+\varepsilon)n/2$; note that it has length $\varepsilon n/4$.

Let x be the selected parent, y be the offspring, and $z \in \{x,y\}$ be the individual selected for survival. We establish the two conditions of the negative drift theorem. The first condition (2.3) for search points with fitness at least $a := (1+\varepsilon/2)n/2$ follows from Lemma 4.2 with $\delta := \varepsilon/2$, yielding a drift of at most $-\varepsilon/4 + \Theta(1/n) = -\Omega(1)$. The second condition (2.4) follows easily from properties of standard bit mutation: the fitness difference $|f(z)-f(x)|$ is clearly bounded by the number of flipping bits. The probability of flipping d bits in a standard bit mutation is at most $1/(d!) \leq 2/2^d$ for all $d \geq 1$. This proves the second condition when choosing $r := 2$ and $\delta := 1$. Invoking the negative drift theorem yields that the probability of one lineage reaching a search point with fitness at least $(1+\varepsilon)n/2$, starting with a fitness at most $(1+\varepsilon/2)n/2$, in $2^{c'\varepsilon n/2}$ steps, for some constant $c' > 0$, is at most $2^{-\Omega(\varepsilon n/4)} = 2^{-\Omega(n)}$. Choosing $c := c'\varepsilon/4$ completes the proof for OneMax.

The same proof can be used for TwoMax with minor modifications: note that if the number of ones is $k \leq n/2$, a fitness difference of d can be achieved by increasing or decreasing the number of ones by d, provided $k+d \leq n/2$ or by creating an offspring with $n-k-d$ ones on the opposite branch. Since $k+d \leq n/2 \leq n-k-d$, the probability for the latter event is no larger than that of the former. The same holds symmetrically for $k \geq n/2$. Hence all transition probabilities are bounded by twice the previous bound for OneMax and the second condition can be fulfilled by doubling r and choosing $c := c'\varepsilon/8$. Then the result follows as for OneMax. □

4.1.1 Experimental Analysis

We provide an experimental analysis as well in order to see how closely the theory matches the empirical performance for reasonable problem sizes. Our analysis is focused on the (μ+1) EA with probabilistic crowding for the TwoMax function. We consider exponentially increasing population sizes $\mu \in \{2,4,8,\ldots,1024\}$ for a problem size $n = 100$ and for 100 runs.

Since we are interested in proving how good/bad this mechanism is, we define the following outcomes and stopping criteria for each run. *Success*, both optima of TwoMax have been reached, i. e., the run is stopped if the population contains both 0^n and 1^n. *Failure*, once the run has reached $10\mu n \ln n$ generations and the population does not contain both optima. By Lemma 4.1, this time period is long enough to allow any reasonable (μ+1) EA variant to find one or two global optima with high probability (unless the best fitness on a branch drops frequently). We report the mean of successes and failures for the 100 runs.

For probabilistic crowding (Algorithm 24), and as proved in Theorem 4.3, for all μ sizes, the method is not able to optimise TwoMax. In all runs the algorithm failed to reach even one optimum, let alone reaching both. Since the algorithm is not able to find any optimum of TwoMax, we ran additional experiments for $n \in \{32, 64, 128, \ldots, 16384\}$ and population size $\mu = 32$ to observe how far the best lineages evolve from $n/2$ and/or how close the best individuals get to reach an optimum. In Figure 4.1, we show the best individuals obtained in each of the 100 runs and its variance. As soon as n increases, the best fitness in the population starts to concentrate around $n/2$ and reaching a fitness of $(1+\varepsilon)n/2$ becomes very difficult for all constants $\varepsilon > 0$ as n grows. Even the best outliers start to get closer and closer to the average of the population.

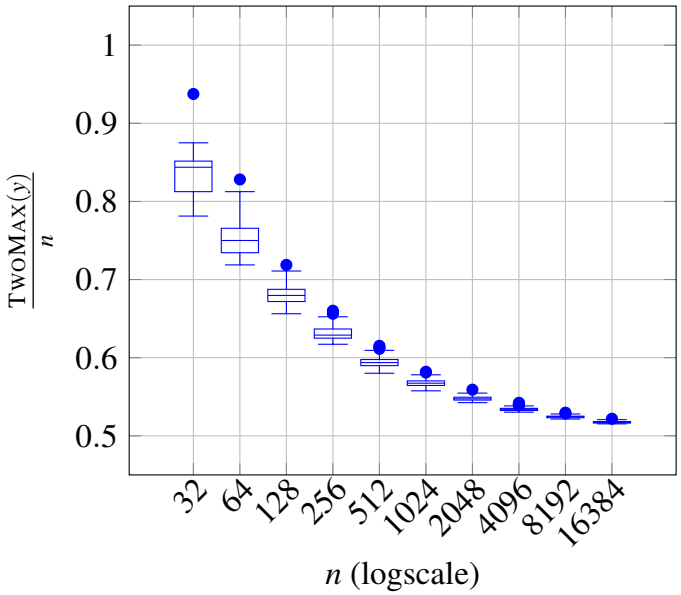


Figure 4.1: The normalised best fitness TwoMax$/n$ reached among 100 runs at the time both optima were found or the $t = 10\mu n \ln n$ generations have been reached on TwoMax for $n \in \{32, 64, 128, \ldots, 16384\}$ by the (μ+1) EA with probabilistic crowding with $\mu = 32$.

4.1.2 Conclusions

We rigorously proved that probabilistic crowding fails miserably; it is not even able to evolve search points that are significantly better than those found by random search, even when given exponential time. The reason is that fitness-proportional selection for survival selection works very similar to uniform selection, and then the algorithm performs an almost blind search on μ independent lineages.

Our results highlight the importance of scaling the fitness, as done in Ballester and Carter (2004); Mengshoel et al. (2014); Mengshoel and Goldberg (2008). An open question is whether fitness scaling would enable probabilistic crowding to find both optima on TWOMAX, and if so, how much the fitness needs to be scaled. A proof where fitness scaling has helped for a variant of the Simple GA on ONEMAX was given in Neumann et al. (2009). We are confident that the proof arguments used here can also be used to analyse more advanced versions of crowding (Galan and Mengshoel, 2010; Mengshoel et al., 2014).

4.2 Restricted Tournament Selection

In restricted tournament selection (RTS) a new offspring competes with the closest element from w (*window size*) more members selected uniformly at random from the population, and the better individual from this competition is selected. The (μ+1) EA with RTS (Algorithm 25) is defined in a similar way as deterministic crowding in Section 3.2.4 and as probabilistic crowding in Section 4.1.

In Algorithm 25 an individual x is selected uniformly at random as a parent and a new individual y is created in the mutation step. Since we are not considering crossover and only one individual is created, w individuals are selected uniformly at random with replacement and stored in a temporary population P_t^*. Then in Line 6 an individual z is selected from population P_t^* with the minimum distance from y (ties are broken uniformly at random), and if the individual y has a fitness at least as good as z, y replaces z.

As distance functions $\mathrm{d}(y,z)$ we consider genotypic distance ($\mathrm{H}(x,y) := \sum_{i=1}^{n} |x_i - y_i|$), and phenotypic distances ($\mathrm{d}(x,y) := ||x|_1 - |y|_1|$) as in Covantes Osuna and Sudholt (2017); Friedrich et al. (2009); Oliveto et al. (2014).

4.2.1 Large Window Sizes Are Effective

Now, let us start with the theoretical analysis for TWOMAX with a positive result for RTS. The following shows that, if w is chosen very large, the (μ+1) EA with RTS behaves almost like the (μ+1) EA with deterministic crowding.

Algorithm 25 (μ+1) EA with restricted tournament selection

Let $t := 0$ and initialise P_0 with μ individuals chosen uniformly at random.
while stopping criterion **not** met **do**
    Choose $x \in P_t$ uniformly at random.
    Create y by flipping each bit in x independently with probability $1/n$.
    Select w individuals uniformly at random from P_t and store them in P_t^*.
    Choose $z \in P_t^*$ with minumum distance to y.
    if $f(y) \geq f(z)$ **then** $P_{t+1} = P_t \setminus \{z\} \cup \{y\}$ **else** $P_{t+1} = P_t$ **end if**
    Let $t := t + 1$.
end while

Theorem 4.4. *If $\mu = o(\sqrt{n}/\log n)$ and $w \geq 2.5\mu \ln n$ then the* (μ+1) EA *with restricted tournament selection using genotypic or phenotypic distance finds both optima on* TwoMax *in time $O(\mu n \log n)$ with probability at least $1 - 2^{-\mu'+3}$, where $\mu' := \min(\mu, \log n)$.*

Note that the probability $1 - 2^{-\mu'+3}$ is close to the success probability $1 - 2^{-\mu+1}$ for deterministic crowding, if $\mu \leq \log n$, apart from a constant factor in front of the $2^{-\mu'+3} = 4 \cdot 2^{-\mu'+1}$ term. For both, the success rate converges to 1 very quickly for increasing population sizes. For restricted tournament selection our probability bound is capped at $1 - 2^{-\log n+3} = 1 - 8/n$ as there is always a small probability of an unexpected takeover occurring.

In order to prove Theorem 4.4, we first analyse the probability of initialising a population such that there are individuals on each branch with a safety gap of σ to the border between branches. This safety gap will be used to exclude the possibility that the best individual on one branch creates offspring on the opposite branch.

Lemma 4.5. *Consider the population of the* (μ+1) EA *on* TwoMax *and for some μ and safety gap $\sigma = n$. The probability of having at least one initial search point with at most $n/2 - \sigma$ ones and one search point with at least $n/2 + \sigma$ ones is at least*

$$1 - 2\left(\frac{1 + 2\sigma \cdot \sqrt{2/n}}{2}\right)^{\mu} \geq 1 - 2^{-\mu+1}(1 + o(1))$$

where the inequality holds if $\sigma\mu = o(\sqrt{n})$.

Proof. Using (Doerr, 2018, Lemma 4.9) for binomial coefficients, a random variable X with binomial distribution $\mathrm{Bin}(n, 1/2)$, for all $z \in [0, n]$ we have

$$\mathrm{Prob}(X = z) \leq \mathrm{Prob}(X = \lfloor n/2 \rfloor) \leq 2^{-n} \cdot \binom{n}{\lfloor n/2 \rfloor} \leq \sqrt{2/n}.$$

So the probability that an individual x is initialised inside the safety gap is at most

$$p_\sigma := \text{Prob}(n/2 - \sigma < |x|_1 < n/2 + \sigma) \leq 2\sigma \cdot \sqrt{2/n}.$$

Now let us define the probability that an individual x is initialised on the outer regions with $|x|_1 \leq n/2 - \sigma$ ones (0^n branch) or $|x|_1 \leq n/2 + \sigma$ ones (1^n branch) as p_0 and p_1, respectively. Note that both p_0 and p_1 are symmetric, and $p_0 + p_1 := 1 - p_\sigma$, and by rewriting we obtain $p_0 := \frac{1-p_\sigma}{2}$ (the same for p_1) with its complement being $1 - \frac{1-p_\sigma}{2} = \frac{1+p_\sigma}{2}$.

So the probability of having no individual with at most $n/2 - \sigma$ ones is $(1-p_1)^\mu = \left(\frac{1+p_\sigma}{2}\right)^\mu$, and the same holds for having no individual with at least $n/2 + \sigma$ ones. Hence the probability of being initialised as stated in the statement of the lemma is at least

$$1 - 2\left(\frac{1+p_\sigma}{2}\right)^\mu = 1 - 2^{-\mu+1} \cdot (1+p_\sigma)^\mu.$$

Plugging in p_σ and using the inequality $1 + x \leq e^x$ as well as $\sigma\mu = o(\sqrt{n})$ we simplify the last term as

$$(1+p_\sigma)^\mu \leq e^{2\sigma\mu\sqrt{2/n}} = e^{o(1)} = \frac{1}{e^{-o(1)}} \leq \frac{1}{1-o(1)} = 1 + o(1),$$

and by plugging all together we have $1 - 2^{-\mu+1}(1+o(1))$. □

Using Lemmas 4.5 and 4.1, we can now prove Theorem 4.4.

Proof of Theorem 4.4. According to Lemma 4.5, with probability $1 - 2^{-\mu+1}(1+o(1))$ the initial population contains at least one search point with at most $n/2 - \log n$ ones and at least one search point with at least $n/2 + \log n$ ones. We assume in the following that this has happened. The probability of mutation flipping at least $\log n$ bits is at most $1/(\log n)! = n^{-\Omega(\log\log n)}$. Taking the union bound over $O(\mu n \log n)$ steps still gives a superpolynomially small error probability. In the following, we work under the assumption that mutation never flips more than $\log n$ bits.

We call two search points *close* if their genotypic distance is at most $\log n$. Due to our assumption on mutations, every newly created offspring is close to its parent. Note that on TWOMAX the phenotypic distance of any two search point is bounded from above by the genotypic distance, hence close search points also have a phenotypic distance of at most $\log n$. Note that, whenever the tournament contains a search point that is close to the new offspring, either the offspring or a close search point will be removed. If this always happens, the best individual on any branch cannot be eliminated by an offspring on the opposite branch; recall

that initially, the best search points on the two branches have phenotypic distance at least $2\log n$, and this phenotypic distance increases if the best fitness on any branch improves. When genotypic distances are being used, the genotypic distance is always at least $2\log n$.

Since each offspring has at least one close search point (its parent), the probability that the tournament does not contain any close search point is at most $(1-1/\mu)^w \leq e^{-w/\mu} = e^{-2.5\ln n} = 1/n^{2.5}$. So long as the best individual on any branch does not get replaced by any individuals on the opposite branch, the conditions of Lemma 4.1 are met. Applying Lemma 4.1 to both branches, by the union bound the probability of both optima being found in time $2e\mu n\ln n$ is at least $1-2/n$. The probability that in this time a tournament occurs that does not involve a close search point is $\mathrm{O}(\mu n\log n)\cdot 1/n^{2.5} = \mathrm{o}(1/n)$ as $\mu = \mathrm{o}(\sqrt{n}/\log n)$.

All failure probabilities sum up to (assuming n large enough)

$$\frac{2}{n} + \mathrm{o}\left(\frac{1}{n}\right) + 2^{-\mu+1}(1+\mathrm{o}(1)) + \frac{\mathrm{O}(\mu n\log n)}{n^{-\Omega(\log\log n)}} \leq \frac{4}{n} + 2^{-\mu+2} \leq 2^{-\mu'+3}$$

where the last inequality follows as $2^{-\mu} \leq 2^{-\mu'}$ and $1/n \leq 2^{-\mu'}$. □

In Theorem 4.4 we chose w so large that every tournament included the offspring's parent with high probability. Then the (μ+1) EA behaves like the (μ+1) EA with deterministic crowding (Friedrich et al., 2009), leading to similar success probabilities (see Table 4.1).

A success probability around $1-2^{-\mu+1}$ is best possible for many diversity mechanisms as with probability $2^{-\mu+1}$ the whole population is initialised on one branch only (for odd n), and then it is likely that only one optimum is reached. Methods like fitness sharing and clearing obtain success probabilities of 1 by more aggressive methods that can force individuals to travel from one branch to the other by accepting worse search points along the way. The performance of restricted tournament selection is hence best possible amongst all mechanisms that do not allow worse search points to enter the population.

4.2.2 Small Window Sizes Can Fail

We now turn our attention to small w. If the w is small in comparison to μ, the possibility emerges that the tournament only contains individuals that are far from the offspring. In that case even the closest individual in the tournament will be dissimilar to the offspring, resulting in a competition between individuals from different "niches" (i. e., sets of similar individuals). The following theorem and its proof show that this may result in one branch taking over the other branch, even when the branch to get extinct is very close to a global optimum. The resulting expected optimisation time is exponential.

Theorem 4.6. *Let $\mu \leq n/8$. The probability that the* $(\mu+1)$ EA *with restricted tournament selection with $w \geq 2$ and either genotypic or phenotypic distances finds both optima on* TWO-MAX *in time n^{n-1} is at most $O(\mu^w/n)$. If $\mu \leq \varepsilon n^{1/w}$ for a sufficiently small constant $\varepsilon > 0$ then the expected time for finding both optima is $\Omega(n^n)$.*

Note that the probability of finding both optima in n^{n-1} generations is $o(1)$ if $w = O(1)$ and μ grows slower than the polynomial $n^{1/w}$. It also holds if $w \leq c(\ln n)/\ln\ln n$ for some constant $0 < c < 1$ and $\mu = O(\log n)$ as then $n^{1/w} = e^{(\ln n)/w} \geq e^{(\ln\ln n)/c} = (\ln n)^{1/c} = \omega(\log n)$, which shows $\mu^w/n = o(1)$.

Proof of Theorem 4.6. The analysis follows the proof of (Friedrich et al., 2009, Theorem 1). We assume that the initial population contains at most one global optimum as the probability of both optima being found during initialization is at most $\mu \cdot 2^{-n} = O(\mu^w/n)$.

We consider the first point of time at which the first optimum is being bound. Without loss of generality, let us assume that this is 0^n. Then we show that with high probability copies of 0^n takeover the population before the other optimum 1^n is found.

Let i be the number of copies of the 0^n individuals in the population, then a good event G_i (good in a sense of leading towards extinction as we are aiming at a negative result) is to increase this number from i to $i+1$. For this it is just necessary to create copies of one of the i individuals. For $n \geq 2$ we have $\text{Prob}(G_i) \geq \frac{i}{\mu} \cdot \left(1 - \frac{1}{n}\right)^n \cdot \left(\frac{\mu-i}{\mu}\right)^w \geq \frac{i}{4\mu} \cdot \left(\frac{\mu-i}{\mu}\right)^w$ since it suffices to select one out of i individuals and to create a copy of the selected individual, and to select w times individuals from the remaining $\mu - i$ individuals. On the other hand, a bad event B_i is to create an 1^n individual in one generation. This probability is clearly bounded by $\text{Prob}(B_i) \leq \frac{1}{n}$ as every individual with at least one zero bit has to flip said bit to create 1^n. Together, the probability that the good event G_i happens before the bad event B_i is

$$\begin{aligned}\text{Prob}(G_i \mid G_i \cup B_i) &\geq \frac{\text{Prob}(G_i)}{\text{Prob}(G_i) + \text{Prob}(B_i)} \geq \frac{\frac{i}{4\mu} \cdot \left(\frac{\mu-i}{\mu}\right)^w}{\frac{i}{4\mu} \cdot \left(\frac{\mu-i}{\mu}\right)^w + \frac{1}{n}} \\ &= 1 - \frac{\frac{1}{n}}{\frac{i}{4\mu} \cdot \left(\frac{\mu-i}{\mu}\right)^w + \frac{1}{n}} \geq 1 - \frac{4\mu}{in \cdot ((\mu-i)/\mu)^w}.\end{aligned}$$

The probability that the i individuals takeover the population before 1^n is found is therefore at least

$$\prod_{i=1}^{\mu} \text{Prob}(G_i \mid G_i \cup B_i) \geq \prod_{i=1}^{\mu} \left(1 - \frac{4\mu}{in \cdot ((\mu-i)/\mu)^w}\right).$$

Using $\frac{4\mu}{n} \leq \frac{1}{2}$ and $1-x \geq e^{-2x}$ for $x \leq \frac{1}{2}$, we obtain

$$\begin{aligned}\prod_{i=1}^{\mu}\left(1-\frac{4\mu}{in\cdot((\mu-i)/\mu)^w}\right) &\geq \prod_{i=1}^{\mu}\exp\left(-\frac{8\mu}{in\cdot((\mu-i)/\mu)^w}\right)\\ &= \exp\left(-\frac{8\mu}{n}\cdot\sum_{i=1}^{\mu-1}\frac{1}{i\cdot((\mu-i)/\mu)^w}\right)\\ &= \exp\left(-\frac{8\mu}{n}\cdot\mu^w\sum_{i=1}^{\mu-1}\frac{1}{i\cdot(\mu-i)^w}\right).\end{aligned}$$

Note that the summands are non-increasing with w. So the worst case is having the smallest possible $w \geq 2$, so we can bound this sum from above in the following way:

$$\begin{aligned}\sum_{i=1}^{\mu-1}\frac{1}{i\cdot(\mu-i)^2} &\leq \sum_{i=1}^{\lfloor\mu/2\rfloor}\frac{1}{i\cdot(\mu-i)^2}+\sum_{i=\lceil\mu/2\rceil}^{\mu-1}\frac{1}{i\cdot(\mu-i)^2}\\ &\leq \sum_{i=1}^{\lfloor\mu/2\rfloor}\frac{1}{i\cdot(\mu/2)^2}+\sum_{i=\lceil\mu/2\rceil}^{\mu-1}\frac{1}{\mu/2\cdot(\mu-i)^2}\\ &\leq \frac{4}{\mu^2}\sum_{i=1}^{\lfloor\mu/2\rfloor}\frac{1}{i}+\frac{2}{\mu}\sum_{i=1}^{\infty}\frac{1}{i^2}=\mathrm{O}\left(\frac{1}{\mu}\right)\end{aligned}$$

as $\sum_{i=1}^{\lfloor\mu/2\rfloor}\frac{1}{i}=\mathrm{O}(\log\mu)$ and $\sum_{i=1}^{\infty}1/i^2=\pi^2/6$. Together we have

$$\prod_{i=1}^{\mu}\mathrm{Prob}(G_i \mid G_i\cup B_i) \geq \exp\left(-\frac{8\mu}{n}\cdot\mu^w\cdot\mathrm{O}\left(\frac{1}{\mu}\right)\right) \geq 1-\mathrm{O}\left(\frac{\mu^w}{n}\right).$$

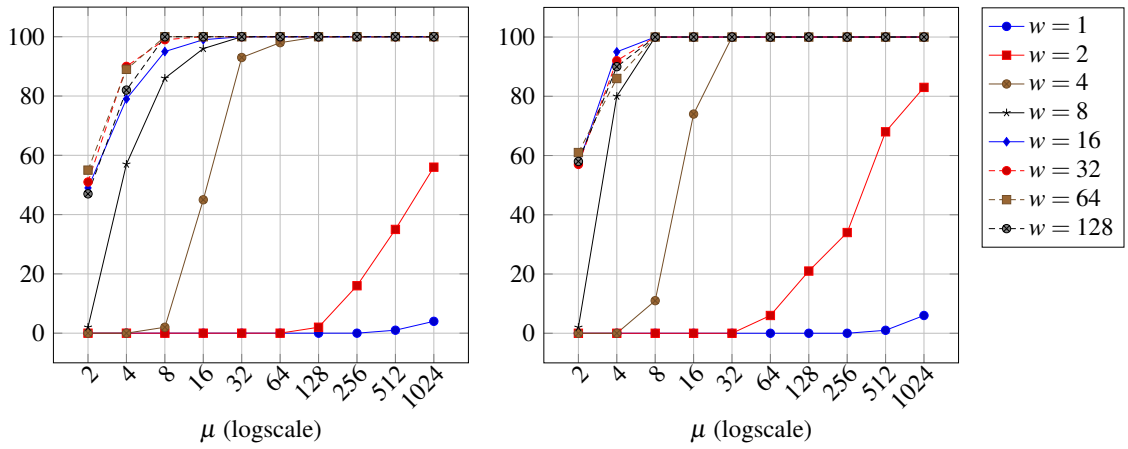
Once the population consists only of copies of 0^n, a mutation has to flip all n bits to find the 1^n optimum. This event has probability n^{-n} and, by the union bound, the probability of this happening in a phase consisting of n^{n-1} generations is at most $\frac{1}{n}=\mathrm{O}(\mu^w/n)$. The sum of all failure probabilities is $\mathrm{O}(\mu^w/n)$, which proves the first claim. For the second claim, observe that the conditional expected optimization time is n^n once the population has collapsed to copies of 0^n individuals. As this situation occurs with probability at least $1-\mathrm{O}(\mu^w/n)=\Omega(1)$ if the constant ε in $\mu \leq \varepsilon n^{1/w}$ is sufficiently small, the unconditional expected optimization time is $\Omega(n^n)$. □

4.2.3 Experimental Analysis

We provide an experimental analysis as well in order to see how closely the theory matches the empirical performance for reasonable problem sizes, and to investigate a wider range

of parameters, where the theoretical results are not applicable. Here we focus our analysis on the (μ+1) EA with restricted tournament selection for the TwoMax function. We consider the same experimental setting defined for probabilistic crowding: population sizes $\mu \in \{2,4,8,\ldots,1024\}$, problem size $n = 100$ and for 100 runs. The same outcomes and stopping criteria defined in Section 4.1.1. *Success*, the population contains both 0^n and 1^n in the population. *Failure*, once the run has reached $10\mu n \ln n$ generations and the population does not contain both optima. We report the mean of successes and failures for the 100 runs.

In the case of RTS (Algorithm 25), we ran experiments for $w \in \{1,2,4,8,\ldots,1024\}$, however we only plot results up to $w = 128$ as the results for large w were very similar. Figure 4.2 shows that for small values of w and μ the algorithm is not able to maintain individuals on both branches of TwoMax for a long period of time, as predicted by Theorem 4.6. It is only when the population size is set to $\mu = 1024$ (where Theorem 4.6 does not apply any more since $\mu > n/8$) the algorithm is able to maintain individuals on both branches before the takeover happens. When setting, for example, $w \geq 8$ and $\mu \geq 32$ the algorithm was able to find both optima with both genotypic and phenotypic distances. It is possible to observe a trade-off between w and μ: larger w allow for a smaller population size μ to be used. Such a trade-off was also indicated by the probability bound $O(\mu^w/n)$ from Theorem 4.6. Our experiments show that RTS works well for much smaller window sizes than those required in Theorem 4.4. As a final remark, the method seems to behave fairly similarly with respect to both distance functions.



(a) Genotypic (b) Phenotypic

Figure 4.2: The number of successful runs measured among 100 runs at the time both optima were found on TwoMax or $t = 10\mu n \ln n$ generations have been reached for $n = 100$ with the (μ+1) EA with restricted tournament selection with $\mu \in \{2,4,8,\ldots,1024\}$, $w \in \{1,2,4,8,\ldots,128\}$, genotypic and phenotypic distance.

4.2.4 Conclusions

The performance of restricted tournament selection seems to vary a lot with the parameters involved. We have shown that if μ and w are set too small, one subpopulation may get extinct. But if w is large enough then RTS behaves similarly to deterministic crowding. For both the probability of finding both optima is close to $1-2^{-\mu+1}$, hence converging to 1 very quickly as μ grows. It still an open problem to theoretically analyse the population dynamics of RTS for intermediate values for w. Our experiments show that RTS can optimise TWOMAX for smaller w than the one required in Theorem 4.4.

4.3 Clearing

Clearing is a niching method inspired by the principle of sharing limited resources within a niche (or subpopulation) of individuals characterised by some similarities. Instead of evenly sharing the available resources among the individuals of a niche, the clearing procedure supplies these resources only to the best individual of each niche: the winner. The winner takes all rather than sharing resources with the other individuals of the same niche as it is done with fitness sharing (Pétrowski, 1996).

Like in fitness sharing, the clearing algorithm uses a dissimilarity measure given by a threshold called clearing radius σ between individuals to determine if they belong to the same niche or not. The basic idea is to preserve the fitness of the individual that has the best fitness (also called dominant individual), while it resets the fitness of all the other individuals of the same niche to zero[2]. With such a mechanism, two approaches can be considered. For a given population, the set of winners is unique. The winner and all the individuals that it dominates are then fictitiously removed from the population. Then the algorithm proceeds in the same way with the new population which is then obtained. Thus, the list of all the winners is produced after a certain number of steps.

On the other hand, the population can be dominated by several winners. It is also possible to generalise the clearing algorithm by accepting several winners chosen among the niche capacity κ (best individuals of each niche defined as the maximum number of winners that a niche can accept). Thus, choosing niching capacities between one and the population size offers intermediate situations between the maximum clearing ($\kappa = 1$) and a standard EA ($\kappa \geq \mu$).

[2]We tacitly assume that all fitness values are larger than 0 for simplicity. In case of a fitness function f with negative fitness values we can change clearing to reset fitness to $f_{\min} - 1$, where $f_{\min}$ is the minimum fitness value of f, such that all reset individuals are worse than any other individuals.

Empirical investigations made in Pétrowski (1996, 1997a,b); Sareni and Krahenbuhl (1998); Singh and Deb (2006) mentioned that clearing surpasses all other niching methods because of its ability to produce a great quantity of new individuals by randomly recombining elements of different niches, controlling this production by resetting the fitness of the poor individuals in each different niche. Furthermore, an elitist strategy prevents the rejection of the best individuals.

We incorporate the clearing method into Algorithm 5, resulting in Algorithm 26. The idea behind Algorithm 26 is: once a population with μ individuals is generated, an individual x is selected and changed according to mutation. A temporary population P_t^* is created from population P_t and the offspring y, then the fitness of each individual in P_t^* is updated according to the clearing procedure shown in Algorithm 27.

Algorithm 26 (μ+1) EA with clearing

1: Let $t := 0$ and initialise P_0 with μ individuals chosen uniformly at random.
2: **while** stopping criterion **not** met **do**
3: Choose $x \in P_t$ uniformly at random.
4: Create y by flipping each bit in x independently with probability $1/n$.
5: Let $P_t^* = P_t \cup \{y\}$.
6: Update $f(P_t^*)$ with the clearing procedure (Algorithm 27).
7: Choose $z \in P_t$ with worst fitness uniformly at random.
8: **if** $f(y) \geq f(z)$ **then** $P_{t+1} = P_t^* \setminus \{z\}$ **else** $P_{t+1} = P_t^* \setminus \{y\}$ **end if**
9: Let $t := t + 1$.
10: **end while**

Each individual is compared with the winner(s) of each niche in order to check if it belongs to a certain niche or not, and to check if its a winner or if it is cleared. Here $\mathrm{d}(P[i], P[j])$ is any dissimilarity measure (distance function) between two individuals $P[i]$ and $P[j]$ of population P. Finally, we keep control of the niche capacity defined by κ. For the sake of clarity, the replacement policy will be the one defined in Witt (2006): the individuals with best fitness are selected (set of winners) and individuals coming from the new generation are preferred if their fitness values are at least as good as the current ones (novelty is rewarded).

Finally, as dissimilarity measures, we consider genotypic or Hamming distance (see Definition 2.5). As TwoMax is a function of unitation, we have adopted the same approach as in previous work (Friedrich et al., 2009; Oliveto et al., 2014) for the phenotypic distance function (see Definition 2.13).

Algorithm 27 Clearing

Require: Set of search points P.
Ensure: Set of winners and cleared individuals in P.

```
1: Sort P according to fitness of individuals by decreasing values.
2: for i := 1 to |P| do
3:     if f(P[i]) > 0 then
4:         winners := 1
5:         for j := i + 1 to |P| do
6:             if f(P[j]) > 0 and d(P[i], P[j]) < σ then
7:                 if winners < κ then winners := winners + 1 else f(P[j]) := 0 end if
8:             end if
9:         end for
10:    end if
11: end for
12: return P.
```

4.3.1 Small Niches

In this section we prove that the (μ+1) EA with phenotypic clearing and a small niche capacity is not only able to achieve both optima of TwoMax but is also able to optimise all functions of unitation with a large enough population, while genotypic clearing fails in achieving such a task (hereinafter, we will refer as phenotypic or genotypic clearing to Algorithm 27 with phenotypic or genotypic distance function, respectively).

4.3.1.1 Phenotypic Clearing

First it is necessary to define a very important property of clearing, which is its capacity of preventing the rejection of the best individuals in the (μ+1) EA, and once μ is defined large enough, clearing and the population size pressure will always optimise any function of unitation.

Note that on functions of unitation all search points with the same number of ones have the same fitness, and for phenotypic clearing with clearing radius $\sigma = 1$ all search points with the same number of ones form a niche. We refer to the set of all search points with i ones as niche i. In order to find an optimum for any function of unitation, it is sufficient to have all niches i, for $0 \leq i \leq n$, being present in the population.

In the (μ+1) EA with phenotypic clearing with $\sigma = 1$, $\kappa \in \mathbb{N}$ and $\mu \geq (n+1) \cdot \kappa$, a niche i can only contain κ winners with i ones. The condition on μ ensures that the population is large enough to store individuals from all possible niches.

Lemma 4.7. *Consider the* (μ+1) EA *with phenotypic clearing with* $\sigma = 1$, $\kappa \in \mathbb{N}$ *and* $\mu \geq (n+1) \cdot \kappa$ *on any function of unitation. Then winners are never removed from the population, i. e., if* $x \in P_t$ *is a winner then* $x \in P_{t+1}$.

Proof. After the first evaluation with clearing, individuals dominated by other individuals are cleared and the dominant individuals are declared as winners. Cleared individuals are removed from the population when new winners are created and occupy new niches. Once an individual becomes a winner, it can only be removed if the size of the population is not large enough to maintain it, as the worst winner is removed if a new winner reaches a new better niche. Since there are at most $n+1$ niches, each having at most κ winners, if $\mu \geq (n+1) \cdot \kappa$ then there must be a cleared individual amongst the $\mu+1$ parents and offspring considered for deletion at the end of the generation. Thus, a cleared individual will be deleted, so winners cannot be removed from the population. □

The behaviour described above means that with the defined parameters and sufficiently large μ to occupy all the niches, we have sufficient conditions for the furthest individuals (individuals with the minimum and maximum number of ones in the population) to reach the opposite edges. Now that we know that a winner cannot be removed from the population by Lemma 4.7, it is just a matter of finding the expected time until 0^n and 1^n are found.

Because of the elitist approach of the (μ+1) EA, winners will never be replaced if we assume a large enough population size. In particular, the minimum (maximum) number of ones of any search point in the population will never increase (decrease). We first estimate the expected time until the two most extreme search points 0^n and 1^n are being found, using arguments similar to the well-known fitness-level method (Section 2.3.5).

Lemma 4.8. *Let* f *be a function of unitation and* $\sigma = 1$, $\kappa \in \mathbb{N}$ *and* $\mu \geq (n+1) \cdot \kappa$. *Then, the expected time for finding the search points* 0^n *and* 1^n *with the* (μ+1) EA *with phenotypic clearing on* f *is* $O(\mu n \log n)$.

Proof. First, we will focus on estimating the time until the 1^n individual is found (by symmetry, the same analysis applies for the 0^n individual). If the current maximum number of ones in any search point is i, it has a probability of being selected of at least $1/\mu$. In order to create a niche with $j > i$ ones, it is sufficient that one of the $n-i$ zeroes is flipped into a 1-bit and the other bits remain unchanged. Each bit flip has a probability of being changed (mutated) of $1/n$ and the probability of the other bits remaining unchanged is $(1-1/n)^{n-1}$. Hence, the probability of creating some niche with $j > i$ ones is at least

$$\frac{1}{\mu} \cdot \frac{n-i}{n} \cdot \left(1 - \frac{1}{n}\right)^{n-1} \geq \frac{n-i}{\mu e n}.$$

The expected time for increasing the maximum number of ones, i, is hence at most $\frac{\mu en}{n-i}$ and the expected time for finding 1^n is at most

$$\sum_{i=0}^{n-1} \frac{\mu en}{n-i} = \mu en \sum_{i=1}^{n} \frac{1}{i} \leq \mu en \ln n = \mathrm{O}(\mu n \log n).$$

Where $\mathrm{H}_n = \sum_{i=1}^{n} 1/i$ is known as the *harmonic number* (Lemma A.17). Adding the same time for finding 0^n proves the claim. □

Once the search points 0^n and 1^n have been found, we can focus on the time required for the algorithm until all intermediate niches are discovered.

Lemma 4.9. *Let f be any function of unitation, $\sigma = 1$, $\kappa \in \mathbb{N}$ and $\mu \geq (n+1) \cdot \kappa$, and assume that the search points 0^n and 1^n are contained in the population. Then, the expected time until all niches are found with the* (μ+1) EA *with phenotypic clearing on f is* $\mathrm{O}(\mu n)$.

Proof. According to Lemma 4.7 and the elitist approach of (μ+1) EA, winners will never be replaced if we assume a large enough population size and by assumption we already have found both search points 0^n and 1^n. As long as the algorithm has not yet found all niches with at least $n/2$ ones, then there must be an index $i \geq n/2$ such that the population does not cover the niche with i ones, but it does cover the niche with $i+1$ ones. We can mutate an individual from niche $i+1$ to populate niche i. The probability of selecting an individual from niche $i+1$ is at least $1/\mu$, and since it is just necessary to flip one of at least $n/2$ 0-bits with probability $1/n$, we have a probability of at least $1/2$ to do so, and a probability of leaving the remaining bits untouched of $(1-1/n)^{n-1} \geq 1/e$. Together, the probability is bounded from below by $1/(2\mu e)$. Using the level-based argument used before for the case of the niches, the expected time to occupy all niches with at least $n/2$ ones is bounded by

$$\sum_{i=n/2}^{n-1} \frac{2\mu e}{1} \leq 2\mu en = \mathrm{O}(\mu n).$$

A symmetric argument applies for the niches with fewer than $n/2$ ones, leading to an additional time of $\mathrm{O}(\mu n)$. □

Theorem 4.10. *Let f be a function of unitation and $\sigma = 1$, $\kappa \in \mathbb{N}$ and $\mu \geq (n+1) \cdot \kappa$. Then, the expected optimisation time of the* (μ+1) EA *with phenotype clearing on f is* $\mathrm{O}(\mu n \log n)$.

Proof. Now that we have defined and proved all conditions where the algorithm is able to maintain every winner in the population (Lemma 4.7), to find the extreme search points (Lemma 4.8) and intermediate niches (Lemma 4.9) of the function f, we can conclude that the total time required to optimise the function of unitation f is $\mathrm{O}(\mu n \log n)$. □

4.3.1.2 Genotypic Clearing

In the case of genotypic clearing with $\sigma = 1$, the (μ+1) EA behaves like the diversity-preserving mechanism called *no genotype duplicates*. The (μ+1) EA with no genotype duplicates rejects the new offspring if the genotype is already contained in the population. The same happens for the (μ+1) EA with genotypic clearing and $\sigma = 1$ if the population is initialised with μ mutually different genotypes (which happens with probability at least $1 - \binom{\mu}{2}2^{-n}$). In other words, conditional on the population being initialised with mutually different search points, both algorithms are identical. In Friedrich et al. (2009, Theorem 2), it was proved that the (μ+1) EA with no genotype duplicates and $\mu = o\left(n^{1/2}\right)$ is not powerful enough to explore the landscape and can be easily trapped in one optimum of TwoMax. Adapting Friedrich et al. (2009, Theorem 2) to the goal of finding both optima and noting that $\binom{\mu}{2}2^{-n} = o(1)$ for the considered μ yields the following.

Corollary 4.11. *The probability that the* (μ+1) EA *with genotypic clearing,* $\sigma = 1$ *and* $\mu = o\left(n^{1/2}\right)$ *finds both optima on* TwoMax *in time* n^{n-2} *is at most* $o(1)$. *The expected time for finding both optima is* $\Omega\left(n^{n-1}\right)$.

As mentioned before, the use of a proper distance is really important in the context of clearing. In our case, we use phenotypic distance for functions of unitation, which has been proved to provide more significant information at the time it is required to define small differences (in our case small niches) among individuals in a population, so the use of that knowledge can be taken into consideration at the time the algorithm is set up. Otherwise, if there is no more knowledge related to the specifics of the problem, genotypic clearing can be used but with larger niches as shown in the following section.

4.3.2 Large Niches

While small niches work with phenotypic clearing, Corollary 4.11 showed that with genotypic clearing small niches are ineffective. This makes sense as for phenotypic clearing with $\sigma = 1$ a niche with i ones covers $\binom{n}{i}$ search points, whereas a niche in genotypic clearing with $\sigma = 1$ only covers one search point. In this section we turn our attention to larger niches, where we will prove that cleared search points are likely to spread, move, and climb down a branch.

We first present general insights into these population dynamics with clearing. These results capture the behaviour of the population in the presence of only one winning genotype x^* (of which there may be κ copies). We estimate the time until in this situation the population evolves a search point of Hamming distance d from said winner, for any $d \leq \sigma$,

or for another winner to emerge (for example, in case an individual of better fitness than x^* is found).

These time bounds are very general as they are independent of the fitness function. This is possible since, assuming the winners are fixed at x^*, all other search points within the clearing radius receive a fitness of 0 and hence are subject to a random walk. We demonstrate the usefulness of our general method by an application to TwoMax with a clearing radius of $\sigma = n/2$, where all winners are copies of either 0^n or 1^n. The results hold both for genotypic clearing and phenotypic clearing as the phenotypic distance of any point x to 0^n (1^n, resp.) equals the Hamming distance of x to 0^n (1^n, resp.).

4.3.2.1 Large Population Dynamics with Clearing

We assume that the population contains only one winner genotype x^*, of which there are κ copies. For any given integer $0 \leq d \leq \sigma$, we analyse the time for the population to reach a search point of Hamming distance at least d from x^*, or for a winner different from x^* to emerge. To this end, we will study a potential function φ that measures the dynamics of the population. Let

$$\varphi(P_t) = \sum_{x \in P_t} \mathrm{H}(x, x^*)$$

be the sum of all Hamming distances of individuals in the population to the winner x^*. The following lemma shows how the potential develops in expectation.

Lemma 4.12. *Let P_t be the current population of the* $(\mu{+}1)$ EA *with genotypic clearing on any fitness function such that the only winners are κ copies of x^* and $\mathrm{H}(x, x^*) < \sigma$ for all $x \in P_t$. Then if no winner different from x^* is created, the expected change of the potential is*

$$\mathrm{E}[\varphi(P_{t+1}) - \varphi(P_t) \mid P_t] = 1 - \frac{\varphi(P_t)}{\mu}\left(\frac{2}{n} + \frac{\kappa}{\mu - \kappa}\right).$$

Before proving the lemma, let us make sense of this formula. Ignore the term $\frac{\kappa}{\mu-\kappa}$ for the moment and consider the formula $1 - \frac{\varphi(P_t)}{\mu} \cdot \frac{2}{n}$. Note that $\varphi(P_t)/\mu$ is the average distance to the winner in P_t. If the population has spread such that it has reached an average distance of $n/2$ then the expected change would be $1 - \frac{\varphi(P_t)}{\mu} \cdot \frac{2}{n} = 1 - \frac{n}{2} \cdot \frac{2}{n} = 0$. Moreover, a smaller average distance will give a positive drift (expected value in the decrease of the distance after a single function evaluation) and an average distance larger than $n/2$ will give a negative drift. This makes sense as a search point performing an independent random walk will attain an equilibrium state around Hamming distance $n/2$ from x^*.

The term $\frac{\kappa}{\mu-\kappa}$ reflects the fact that losers in the population do not evolve in complete isolation. The population always contains κ copies of x^* that may create offspring and may prevent the population from venturing far away from x^*. In other words, there is a constant influx of search points descending from winners x^*. As the term $\frac{\kappa}{\mu-\kappa}$ indicates, this effect grows with κ, but (as we will see later) it can be mitigated by setting the population size μ sufficiently large.

Proof of Lemma 4.12. We use the notation from Algorithm 26, where x is the parent, y is its offspring, and z is the individual considered for removal from the population. If an individual $x \in P_t$ is selected as parent, the expected distance of its mutant to x^* is

$$\mathrm{E}[\mathrm{H}(y,x^*) \mid x] = \mathrm{H}(x,x^*) + \frac{n-\mathrm{H}(x,x^*)}{n} - \frac{\mathrm{H}(x,x^*)}{n} = 1 + \mathrm{H}(x,x^*)\left(1-\frac{2}{n}\right).$$

Hence after a uniform parent selection and mutation, the expected distance in the offspring is

$$\mathrm{E}[\mathrm{H}(y,x^*) \mid P_t] = \sum_{x\in P_t} \frac{1}{\mu} \cdot \left(1 + \mathrm{H}(x,x^*)\left(1-\frac{2}{n}\right)\right) = 1 + \frac{\varphi(P_t)}{\mu}\left(1-\frac{2}{n}\right). \tag{4.2}$$

After mutation and clearing procedure, there are μ individuals in P_t, including κ winners, which are copies of x^*. Let C denote the multiset of these κ winners. As all $\mu-\kappa$ non-winner individuals in P_t have fitness 0, one of these will be selected uniformly at random for deletion. The expected distance to x^* in the deleted individual is

$$\mathrm{E}[\mathrm{H}(z,x^*) \mid P_t] = \sum_{x\in P_t\setminus C} \frac{1}{\mu-\kappa} \cdot \mathrm{H}(x,x^*) = \sum_{x\in P_t} \frac{1}{\mu-\kappa} \cdot \mathrm{H}(x,x^*) = \frac{\varphi(P_t)}{\mu-\kappa}. \tag{4.3}$$

Recall that the potential is the sum of Hamming distances to x^*, hence adding y and removing z yields $\varphi(P_{t+1}) = \varphi(P_t) + \mathrm{H}(y,x^*) - \mathrm{H}(z,x^*)$. Along with (4.2) and (4.3), the expected change of the potential is

$$\begin{aligned}\mathrm{E}[\varphi(P_{t+1}) - \varphi(P_t) \mid P_t] &= \mathrm{E}[\mathrm{H}(y,x^*) \mid P_t] - \mathrm{E}[\mathrm{H}(z,x^*) \mid P_t] \\ &= 1 + \frac{\varphi(P_t)}{\mu}\left(1-\frac{2}{n}\right) - \frac{\varphi(P_t)}{\mu-\kappa}.\end{aligned}$$

Using that

$$\frac{\varphi(P_t)}{\mu} - \frac{\varphi(P_t)}{\mu-\kappa} = \frac{(\mu-\kappa)\varphi(P_t)}{\mu(\mu-\kappa)} - \frac{\mu\varphi(P_t)}{\mu(\mu-\kappa)} = -\frac{\kappa\varphi(P_t)}{\mu(\mu-\kappa)},$$

the above simplifies to

$$\begin{aligned}\mathrm{E}[\varphi(P_{t+1}) - \varphi(P_t) \mid P_t] &= 1 - \frac{2\varphi(P_t)}{\mu n} - \frac{\kappa\varphi(P_t)}{\mu(\mu-\kappa)} \\ &= 1 - \frac{\varphi(P_t)}{\mu}\left(\frac{2}{n} + \frac{\kappa}{\mu-\kappa}\right).\end{aligned}$$

□

The potential allows us to conclude when the population has reached a search point of distance at least d from x^*. The following lemma gives a sufficient condition.

Lemma 4.13. *If P_t contains κ copies of x^* and $\varphi(P_t) > (\mu-\kappa)(d-1)$ then P_t must contain at least one individual x with $\mathrm{H}(x,x^*) \geq d$.*

Proof. There are at most $\mu - \kappa$ individuals different from x^*. By the pigeon-hole principle, at least one of them must have at least distance d from x^*. □

In order to bound the time for reaching a high potential given in Lemma 4.13, we will use the Theorem 2.33, a straightforward extension of the variable drift theorem (Johannsen, 2010) towards reaching any state smaller than some threshold a. It can be derived with simple adaptations to the proof in Rowe and Sudholt (2014).

The following lemma now gives an upper bound on the first hitting time (the random variable that denotes the first point in time to reach a certain point) of a search point with distance at least d to the winner x^*.

Lemma 4.14. *Let P_t be the current population of the* (μ+1) EA *with genotypic clearing and $\sigma \leq n/2$ on any fitness function such that P_t contains κ copies of a unique winner x^* and $\mathrm{H}(x,x^*) < d$ for all $x \in P_t$. For any $0 \leq d \leq \sigma$, if $\mu \geq \kappa \cdot \frac{dn-2d+2}{n-2d+2}$ then the expected time until a search point x with $\mathrm{H}(x,x^*) \geq d$ is found, or a winner different from x^* is created, is $\mathrm{O}(\mu n \log \mu)$.*

Proof. We pessimistically assume that no other winner is created and estimate the first hitting time of a search point with distance at least d. As φ can only increase by at most n in one step, $h_{\max} := (\mu-\kappa)(d-1)+n$ is an upper bound on the maximum potential that can be achieved in the generation where a distance of d is reached or exceeded for the first time.

In order to apply drift analysis, we define a distance function that describes how close the algorithm is to reaching a population where a distance d was reached. We consider the random walk induced by $X_t := h_{\max} - \varphi(P_t)$, stopped as soon as a Hamming distance of at least d from x^* is reached. Due to our definition of $h_{\max}$, the random walk only attains values in $\{0,\ldots,h_{\max}\}$ as required by the variable drift theorem.

By Lemma 4.12, abbreviating $\alpha := \frac{1}{\mu}\left(\frac{2}{n} + \frac{\kappa}{\mu-\kappa}\right)$, X_t decreases in expectation by at least $h(P_t) := 1 - \alpha\varphi(P_t) = 1 - \alpha h_{\max} + \alpha h(P_t)$, provided $h(P_t) > 0$. By definition of h and Lemma 4.13, the population reaches a distance of at least d once the distance $h_{\max} - \varphi(P_t)$ has dropped below n. Using the generalised variable drift theorem, the expected time till this happens is at most

$$\frac{n}{1-\alpha h_{\max}+\alpha n} + \int_n^{h_{\max}} \frac{1}{1-\alpha h_{\max}+\alpha x}\mathrm{d}x.$$

Using $\int \frac{1}{ax+b}\,\mathrm{d}x = \frac{1}{a}\ln|ax+b|$ (Abramowitz, 1974, Equation 3.3.15), we get

$$\begin{aligned}
&\frac{n}{1-\alpha h_{\max}+\alpha n} + \left[\frac{1}{\alpha}\ln(1-\alpha h_{\max}+\alpha x)\right]_n^{h_{\max}} \\
&= \frac{n}{1-\alpha h_{\max}+\alpha n} + \frac{1}{\alpha}\cdot(\ln(1) - \ln(1-\alpha h_{\max}+\alpha n)) \\
&= \frac{n}{1-\alpha h_{\max}+\alpha n} + \frac{1}{\alpha}\ln((1-\alpha h_{\max}+\alpha n)^{-1}).
\end{aligned}$$

We now bound the term $1-\alpha h_{\max}+\alpha n$ from below as follows.

$$\begin{aligned}
1-\alpha h_{\max}+\alpha n &= 1 - (\mu-\kappa)(d-1)\cdot\frac{1}{\mu}\left(\frac{2}{n} + \frac{\kappa}{\mu-\kappa}\right) \\
&= 1 - \frac{2(\mu-\kappa)(d-1)+\kappa(d-1)n}{\mu n} \\
&= \frac{\mu n - 2\mu d + 2\kappa d - \kappa d n + 2\mu - 2\kappa + \kappa n}{\mu n} \\
&= \frac{\kappa}{\mu} + \frac{n-2d+2}{n} - \frac{\kappa d n - 2\kappa d + 2\kappa}{\mu n} \\
&\geq \frac{\kappa}{\mu} + \frac{n-2d+2}{n} - \frac{n-2d+2}{n} \\
&= \frac{\kappa}{\mu}
\end{aligned}$$

where in the penultimate step we used the assumption $\mu \geq \kappa\cdot\frac{dn-2d+2}{n-2d+2}$. Along with $\alpha \geq 2/(\mu n)$, the expected time bound simplifies to

$$\frac{n}{1-\alpha h_{\max}+\alpha n} + \frac{1}{\alpha}\ln((1-\alpha h_{\max}+\alpha n)^{-1}) \leq \frac{n}{\kappa/\mu} + \frac{\mu n}{2}\ln(\mu/\kappa) = \mathrm{O}(\mu n \log \mu). \quad \square$$

The minimum threshold $\kappa\cdot\frac{dn-2d+2}{n-2d+2}$ for μ contains a factor of κ. The reason is that the fraction of winners in the population needs to be small enough to allow the population to

escape from the vicinity of x^*. The population size hence needs to grow proportionally to the number of winners κ the population is allowed to store.

Note that the restriction $d \leq \sigma \leq n/2$ is necessary in Lemma 4.14. Individuals evolving within the clearing radius, but at a distance larger than $n/2$ to x^* will be driven back towards x^*. If d is significantly larger than $n/2$, we conjecture that the expected time for reaching a distance of at least d from x^* becomes exponential in n.

4.3.2.2 Upper Bound for TwoMax

It is now easy to apply Lemma 4.14 in order to achieve a runtime bound on TwoMax. Putting $d = \sigma = n/2$, the condition on μ simplifies to

$$\mu \geq \kappa \cdot \frac{dn - 2d + 2}{n - 2d + 2} = \kappa \cdot \frac{n^2/2 - n + 2}{2},$$

which is implied by $\mu \geq \kappa n^2/4$. Lemma 4.14 then implies the following. Recall that for $x^* \in \{0^n, 1^n\}$, genotypic distances $\mathrm{H}(x, x^*)$ equal phenotypic distances, hence the result applies to both genotypic and phenotypic clearing.

Corollary 4.15. *Consider the* $(\mu{+}1)$ EA *with genotypic or phenotypic clearing,* $\kappa \in \mathbb{N}, \mu \geq \kappa n^2/4$ *and* $\sigma = n/2$ *on* TwoMax *with a population containing* κ *copies of* 0^n (1^n)*. Then the expected time until a search point with at least (at most)* $n/2$ *ones is found is* $\mathrm{O}(\mu n \log \mu)$.

Theorem 4.16. *The expected time for the* $(\mu{+}1)$ EA *with genotypic or phenotypic clearing,* $\mu \geq \kappa n^2/4$*,* $\mu \leq \mathrm{poly}(n)$ *and* $\sigma = n/2$ *finding both optima on* TwoMax *is* $\mathrm{O}(\mu n \log n)$.

Proof. We first estimate the time to reach one optimum, 0^n or 1^n. The population is elitist as it always contains a winner with the best-so-far fitness. Hence we can apply the level-based argument as follows. If the current best fitness is i, it can be increased by selecting an individual with fitness i (probability at least $1/\mu$) and flipping only one of $n - i$ bits with the minority value (probability at least $(n-i)/(en)$). The expected time for increasing the best fitness i is hence at most $\mu \cdot en/(n-i)$ and the expected time for finding some optimum $x^* \in \{0^n, 1^n\}$ is at most

$$\sum_{i=n/2}^{n-1} \mu \cdot \frac{en}{n-i} = e\mu n \sum_{i=1}^{n/2} \frac{1}{i} \leq e\mu n \ln n.$$

In order to apply Corollary 4.15, we need to have κ copies of x^* in the population. While this isn't the case, a generation picking x^* as parent and not flipping any bits creates another winner x^* that will remain in the population. If there are j copies of x^*, the probability to

create another winner is at least $j/\mu \cdot (1-1/n)^n \geq j/(4\mu)$ (using $n \geq 2$). Hence the time until the population contains κ copies of x^* is at most

$$\sum_{j=1}^{\kappa} \frac{4\mu}{j} = \mathrm{O}(\mu \log \kappa) = \mathrm{O}(\mu \log n)$$

as $\kappa \leq \mu \leq \mathrm{poly}(n)$.

By Corollary 4.15, the expected time till a search point on the opposite branch is created is $\mathrm{O}(\mu n \log \mu) = \mathrm{O}(\mu n \log n)$. Since the best individual on the opposite branch is a winner in its own niche, it will never be removed. This allows the population to climb this branch as well. Repeating the arguments from the first paragraph of this proof, the expected time till the second optimum is found is at most $e\mu n \ln n$. Adding up all expected times proves the claim. □

One limitation of Theorem 4.16 is the steep requirement on the population size: $\mu \geq \kappa n^2/4$. The condition on μ was chosen to ensure a positive drift of the potential for all populations that haven't reached distance d yet, including the most pessimistic scenario of all losers having distance $d-1$ to x^*. Such a scenario is unlikely as we will see in Sections 4.3.4.1 and 4.3.4.2 where experiments suggest that the population tends to spread out, covering a broad range of distances. With such a spread, a distance of d can be reached with a much smaller potential than that indicated by Lemma 4.13. We conjecture that the (μ+1) EA with clearing is still efficient on TwoMax if $\mu = \mathrm{O}(n)$. However, proving this theoretically may require new arguments on the distribution of the κ winners and losers inside the population.

4.3.2.3 On the Choice of the Population Size for TwoMax

To get further insights into what population sizes μ are necessary, we show in the following that the (μ+1) EA with clearing becomes inefficient on TwoMax if μ is too small, that is, smaller than $n/\mathrm{polylog}(n)$. The reason is as follows: assume that the population only contains a single optimum x^*, and further individuals that are well within a niche of size $\sigma = n/2$ surrounding x^*. Due to clearing, the population will always contain a copy of x^*. Hence there is a constant influx of individuals that are offspring, or, more generally, recent descendants of x^*. We refer to these individuals informally as *young*; a rigorous notation will be provided in the proof of Theorem 4.17. Intuitively, young individuals are similar to x^*, and thus are likely to produce further offspring that are also *young*, i. e., similar to x^* when chosen as parents.

We will show in the following that if the population size μ is small, young individuals will frequently takeover the whole population, creating a population where all individuals are similar to x^*. This takeover happens much faster than the time the algorithm needs to evolve a lineage that can reach a Hamming distance $n/2$ to the optimum.

The following theorem shows that if the population size is too small, the (μ+1) EA is unable to escape from one local optimum, assuming that it starts with a population of search points that have recently evolved from said optimum.

Theorem 4.17. *Consider the* (μ+1) EA *with genotypic or phenotypic clearing on* TwoMax *with* $\mu \leq n/(4\log^3 n)$, $\kappa = 1$ *and* $\sigma = n/2$, *starting with a population containing only search points that have evolved from one optimum* x^* *within the last* $\mu n/32$ *generations. Then the probability that both optima are found within time* $n^{(\log n)/2}$ *is* $n^{-\Omega(\log n)}$.

The following lemma describes a stochastic process that we will use in the proof of Theorem 4.17 to model the number of "young" individuals over time. We are interested in the first hitting time of state μ as this is the first point in time where young individuals have taken over the whole population of size μ. The transition probabilities for states $1 < X_t < \mu$ reflect the evolution of a fixed-size population containing two species (young and old in our case): in each step one individual is selected for reproduction, and another individual is selected for replacement. If they stem from the same species, the size of both species remains the same. But if they stem from different species, the size of the first species can increase or decrease by 1, with equal probability.

This is similar to the *Moran process* in population genetics (Ewens, 2004, Section 3.4) which ends when one species has evolved to fixation (i. e. has taken over the whole population) or extinction. Our process differs as state 1 is reflecting, hence extinction of young individuals is impossible. Notably, we will show that, compared to the original Moran process, the expected time for the process to end is larger by a factor of order $\log \mu$. Other variants of the Moran process have also appeared in different related contexts such as the analysis of Genetic Algorithms (Lemma 6 in Dang et al., 2016a) and the analysis of the compact Genetic Algorithm (Lemma 7 in Sudholt and Witt, 2016). The following lemma gives asymptotically tight bounds on the time young individuals need to evolve to fixation.

Lemma 4.18. *Consider a Markov chain* $\{X\}_{t\geq 0}$ *on* $\{1,2,\ldots,\mu\}$ *with transition probabilities for* $1 \leq X_t < \mu$

$$\text{Prob}(X_{t+1} = X_t + 1 \mid X_t) = X_t(\mu - X_t)/\mu^2$$

for $1 < X_t < \mu$,

$$\text{Prob}(X_{t+1} = X_t - 1 \mid X_t) = X_t(\mu - X_t)/\mu^2$$

and $X_{t+1} = X_t$ with the remaining probability. Let T be the first hitting time of state μ, then for all starting states X_0,

$$\frac{1}{2} \cdot (\mu - X_0)\mu \ln(\mu - 1) \leq \mathrm{E}[T \mid X_0] \leq 4(\mu - X_0)\mu \mathrm{H}_n(\mu/2) \leq 4\mu^2 \ln \mu.$$

In addition, if $\mu \leq n$ then $\mathrm{Prob}\big(T \geq 8\mu^2 \log^3 n\big) \leq n^{-\log n}$.

Proof. Let us abbreviate $\mathrm{E}_i = \mathrm{E}[T \mid X_t = i]$, then $\mathrm{E}_\mu = 0$, $\mathrm{E}_1 = \frac{\mu^2}{\mu-1} + \mathrm{E}_2$, and for all $1 < i < \mu$ we have

$$\begin{aligned}
\mathrm{E}_i &= 1 + \frac{i(\mu-i)}{\mu^2} \cdot \mathrm{E}_{i+1} + \frac{i(\mu-i)}{\mu^2} \cdot \mathrm{E}_{i-1} + \left(1 - \frac{2i(\mu-i)}{\mu^2}\right) \cdot \mathrm{E}_i \\
\Leftrightarrow\ \frac{2i(\mu-i)}{\mu^2} \cdot \mathrm{E}_i &= 1 + \frac{i(\mu-i)}{\mu^2} \cdot \mathrm{E}_{i+1} + \frac{i(\mu-i)}{\mu^2} \cdot \mathrm{E}_{i-1} \\
\Leftrightarrow\ 2\mathrm{E}_i &= \frac{\mu^2}{i(\mu-i)} + \mathrm{E}_{i+1} + \mathrm{E}_{i-1} \\
\Leftrightarrow\ \mathrm{E}_i - \mathrm{E}_{i+1} &= \frac{\mu^2}{i(\mu-i)} + \mathrm{E}_{i-1} - \mathrm{E}_i.
\end{aligned}$$

Introducing $D_i := \mathrm{E}_i - \mathrm{E}_{i+1}$, this is

$$D_i = \frac{\mu^2}{i(\mu-i)} + D_{i-1}.$$

For D_1 we get

$$D_1 = \mathrm{E}_1 - \mathrm{E}_2 = \left(\frac{\mu^2}{\mu-1} + \mathrm{E}_2\right) - \mathrm{E}_2 = \frac{\mu^2}{\mu-1}.$$

More generally, we expand D_i to get

$$D_i = \sum_{j=1}^{i} \frac{\mu^2}{j(\mu-j)} = \mu^2 \sum_{j=1}^{i} \frac{1}{j(\mu-j)}$$

Now we can express E_i in terms of D_i variables as follows. For all $1 \leq i < \mu$,

$$\mathrm{E}_i = \underbrace{(\mathrm{E}_i - \mathrm{E}_{i+1})}_{D_i} + \underbrace{(\mathrm{E}_{i+1} - \mathrm{E}_{i+2})}_{D_{i+1}} + \cdots + \underbrace{(\mathrm{E}_{\mu-1} - \mathrm{E}_\mu)}_{D_{\mu-1}} + \underbrace{\mathrm{E}_\mu}_{0}$$

hence

$$\mathrm{E}_i = D_i + \ldots + D_{\mu-1} = \mu^2 \sum_{k=i}^{\mu-1} \sum_{j=1}^{k} \frac{1}{j(\mu-j)}$$

We now bound this double-sum from above and below.

$$\begin{aligned}\mu^2 \sum_{k=i}^{\mu-1} \sum_{j=1}^{k} \frac{1}{j(\mu-j)} &\leq \mu^2 \sum_{k=i}^{\mu-1} \left(\sum_{j=1}^{\lfloor \mu/2 \rfloor} \frac{1}{j(\mu-j)} + \sum_{j=\lfloor \mu/2 \rfloor+1}^{k} \frac{1}{j(\mu-j)} \right) \\ &\leq \mu^2 \sum_{k=i}^{\mu-1} \left(\sum_{j=1}^{\lfloor \mu/2 \rfloor} \frac{1}{j(\mu-j)} + \sum_{j=1}^{\lfloor \mu/2 \rfloor} \frac{1}{j(\mu-j)} \right) \\ &\leq \mu^2 \sum_{k=i}^{\mu-1} \frac{4}{\mu} \cdot \mathrm{H}_n(\lfloor \mu/2 \rfloor) = 4(\mu-i)\mu \mathrm{H}_n(\lfloor \mu/2 \rfloor).\end{aligned}$$

The final inequality follows from $(\mu-i) \cdot \mathrm{H}_n(\lfloor \mu/2 \rfloor) \leq \mu \ln \mu$.

The lower bound follows from

$$\begin{aligned}\mu^2 \sum_{k=i}^{\mu-1} \sum_{j=1}^{k} \frac{1}{j(\mu-j)} \geq \mu \sum_{k=i}^{\mu-1} \sum_{j=1}^{k} \frac{1}{j} \geq \mu \sum_{k=i}^{\mu-1} \ln(k) &= \mu \ln \left(\prod_{k=i}^{\mu-1} k \right) \\ &\geq \mu \ln \left((\mu-1)^{(\mu-i)/2} \right) \\ &= \frac{1}{2} \cdot (\mu-i)\mu \ln(\mu-1).\end{aligned}$$

where in the last inequality we used that $(\mu-1-j)(i+j) \geq \mu-1$ for all $0 \leq j \leq \mu-1$, allowing us to group factors in $\lfloor (\mu-i)/2 \rfloor$ pairs whose product is at least $\mu-1$, leaving a remaining factor of at least $\mu/2 \geq (\mu-1)^{1/2}$ if $(\mu-i)$ is odd.

For the second statement, we use standard arguments on independent phases. By Markov's inequality, the probability that takeover takes longer than $2 \cdot (4\mu^2 \ln \mu)$ is at most $1/2$. Since the upper bound holds for any X_0, we can iterate this argument $\log^2 n$ times. Then the probability that we do not have a takeover in $2 \cdot (4\mu^2 \ln \mu \cdot \log^2 n) \leq 8\mu^2 \log^3 n$ steps (using $\mu \leq n$) is $2^{-\log^2 n} = n^{-\log n}$. □

Now we prove that the time required to reach a new niche with $\sigma = n/2$ is larger than the time required for "young" individuals to takeover the population. In other words, once a winner x^* is found and assigned to an optimum, with a small μ, the time for a takeover is shorter than the required time to find a new niche. This will imply that the algorithm needs superpolynomial time to escape from the influence of the winner x^* and in consequence it needs superpolynomial time to find the opposite optimum.

We analyse the dynamics within the population by means of so-called *family trees* from Section 2.3.7. Here we make use of Lemma 1 in Sudholt (2009) (which is an adaptation from Lemma 2 and proof of Theorem 4 in Witt, 2006) to show that the individuals in $\mathrm{T}(r^*)$ are

still concentrated around r^*. If the distance from r^* to all optima is not too small, then it is unlikely that an optimum has been found after t steps.

Lemma 4.19 (Adapted from Lemma 1 in Sudholt, 2009)**.** *For the* (μ+1) EA *with or without clearing, let r^* be an individual entering the population in some generation t^*. The probability that within the following t generations some $y^* \in T(r^*)$ emerges with $\mathrm{H}(r^*, y^*) \geq 8t/\mu$ is $2^{-\Omega(t/\mu)}$.*

Lemma 1 in Sudholt (2009) applies to (μ+λ) EA without clearing. We recap Witt's basic proof idea to make this section self-contained and also to convince the reader why the result also applies to the (μ+1) EA with clearing.

The analysis is divided in two parts. In the first part it is shown that family trees are unlikely to be very deep. Since every individual is chosen as parent with probability $1/\mu$, the expected length of a path in the family tree after t generations is bounded by t/μ. Large deviations from this expectation are unlikely. Lemma 2 in Witt (2006) shows that the probability that a family tree has depth at least $3t/\mu$ is $2^{-\Omega(t/\mu)}$. This argument only relies on the fact that parents are chosen uniformly at random, which also holds for the (μ+1) EA with clearing.

For family trees whose depth is bounded by $3t/\mu$, all path lengths are bounded by $3t/\mu$. Each path corresponds to a sequence of standard bit mutations, and the Hamming distance between any two search points on the same path can be bounded by the number of bits flipped in all mutations that lead from one search point to the other. By applying Chernoff bounds (see Lemma A.15) with respect to the upper bound $4t/\mu$ on the expectation instead of the expectation itself (cf. page 75 in Witt, 2006), we obtain that the probability of an individual of Hamming distance at least $8t/\mu$ to r^* emerging on a particular path is at most $e^{-4t/(3\mu)}$. Taking the union bound over all possible paths in the family tree still gives a failure probability of $2^{-\Omega(t/\mu)}$. Adding the failure probabilities from both parts proves the claim.

Now, Lemma 4.19 implies the following corollary.

Corollary 4.20. *The probability that, starting from a search point x^*, within $\mu n/16$ generations the* (μ+1) EA *with clearing evolves a lineage that reaches Hamming distance at least $n/2$ to its founder x^* is $2^{-\Omega(n)}$.*

Now we put Lemma 4.18 and Corollary 4.20 together to prove Theorem 4.17.

Proof of Theorem 4.17. By assumption, all individuals in the population are descendants of individuals with genotype x^*, and this property will be maintained over time. This means that every individual x in the population P_t at time t will have an ancestor that has genotype x^* (our notion of *ancestor* and *descendant* includes the individual itself). Tracing back x's

ancestry, let $t^* \leq t$ be the most recent generation where an ancestor of x has genotype x^*. Then we define the *age* of x as $t - t^*$. Informally, the age describes how much time a search point has had to evolve differences from the genotype x^*. Note that the age of x^* itself is always 0 and as the population always contains a winner x^*, it always contains at least one individual of age 0.

Now assume that a new search point x is created with $\mathrm{H}(x, x^*) \geq n/2$. If x has age at most $\mu n/16$ then there exists a lineage from a copy of x^* to x that has emerged in at most $\mu n/16$ generations. This corresponds to the event described in Corollary 4.20, and by said corollary the probability of this event happening is at most $2^{-\Omega(n)}$. Taking the union bound over all family trees (of which there are at most μ in every generation) and the first $n^{\log n}$ generations, the probability that such a lineage does emerge in any family tree and at any point in time within the considered time span is still bounded by $\mu n^{\log n} \cdot 2^{-\Omega(n)} = 2^{-\Omega(n)}$.

We now show using Lemma 4.18 that it is very unlikely that individuals with age larger than $\mu n/16$ emerge. We say that a search point x is *T-young* if it has genotype x^* or if its most recent ancestor with genotype x^* was born during or after generation T. Otherwise, x is called *T-old*. We omit the parameter "T" whenever the context is obvious. A key observation is that youth is inheritable: if a young search point is chosen as parent, then the offspring is young as well. If an old search point is chosen as parent, then the offspring is old as well, unless mutation turns the offspring into a copy of x^*.

Let X_t be the number of young individuals in the population at time t, and pessimistically ignore the fact that old individuals may create young individuals through lucky mutations. Then in order to increase the number of young individuals, it is necessary and sufficient to choose a young individual as parent (probability X_t/μ) and to select an old parent for replacement. The probability of the latter is $(\mu - X_t)/\mu$ as there are $\mu - X_t$ old parents and the individual to be removed is chosen uniformly at random among μ individuals whose fitness is cleared. Hence, for $1 \leq X_t < \mu$, $\mathrm{Prob}(X_{t+1} = X_t + 1 \mid X_t) = X_t(\mu - X_t)/\mu^2$. Similarly, the number of old individuals increases if and only if an old individual is chosen as parent (probability $(\mu - X_t)/\mu$) and a young individual is chosen for replacement (probability X_t/μ), hence for $1 < X_t < \mu$ we have $\mathrm{Prob}(X_{t+1} = X_t - 1 \mid X_t) = X_t(\mu - X_t)/\mu^2$. Otherwise, $X_{t+1} = X_t$. Note that $X_t \geq 1$ since the winner x^* is young and will never be removed. This matches the Markov chain analysed in Lemma 4.18.

Now consider a generation T where all individuals in the population have ages at most $\mu n/32$. By assumption, this property is true for the initial population. At time T, the population contains at least one T-young individual: the winner x^*. By Lemma 4.18, with probability at least $1 - n^{-\log n}$, within the next $8\mu^2 \log^3 n \leq \mu n/32$ generations, using the condition $\mu \leq n/(4\log^3 n)$, the population will reach a state where $X_t = \mu$, that is, all

individuals are T-young. Assuming this does happen, let $T' \leq T + \mu n/32$ denote the first point in time where this happens. Then at time T' all individuals have ages at most $\mu n/32$, and we can iterate the above arguments with T' instead of T.

Each such iteration carries a failure probability of at most $n^{-\log n}$. Taking the union bound over failure probabilities $n^{-\log n}$ over the first $n^{(\log n)/2}$ generations yields that the probability of an individual of age larger than $\mu n/16$ emerging is only $n^{(\log n)/2} \cdot n^{-\log n} = n^{-(\log n)/2}$.

Adding failure probabilities $2^{-\Omega(n)}$ and $n^{-(\log n)/2}$ completes the proof. □

We conjecture that a population size of $\mu = O(n)$ is sufficient to optimise TwoMax in expected time $O(\mu n \log n)$, that is, that the conditions in Theorem 4.16 can be improved.

4.3.3 Generalisation to Other Example Landscapes

Note that, in contrast to previous analyses of fitness sharing (Friedrich et al., 2009; Oliveto et al., 2014), our analysis of the clearing mechanism does not make use of the specific fitness values of TwoMax. The main argument of how to escape from one local optimum only depends on the size of its basin of attraction. Our results therefore easily extend to more general function classes that can be optimised by leaving a basin of attraction of width at most $n/2$. We consider more general classes of examples landscapes introduced by Jansen and Zarges (2016) addressing the need for suitable benchmark functions for the theoretical analysis of evolutionary algorithms on multimodal functions (see Section 2.1.1.2).

4.3.3.1 Nearest Peak Functions

We first argue that our results easily generalise to nearest peak functions with two complementary peaks $p_2 = \overline{p_1}$, arbitrary slopes $a_1, a_2 \in \mathbb{R}^+$, and arbitrary offsets $b_i \in \mathbb{R}_0^+$. The generalisation from peaks $0^n, 1^n$ as for TwoMax to peaks $p_2 = \overline{p_1}$ is straightforward: we can swap the meaning of zeros and ones for any selection of bits without changing the behaviour of the algorithm, hence the (μ+1) EA with clearing will show the same stochastic behaviour on peaks $0^n, 1^n$ as well as on arbitrary peaks $p_2 = \overline{p_1}$. As for TwoMax, if only one peak x^* has been found, the basin of attraction of the other peak is found once a search point with Hamming distance at least $n/2$ to x^* is generated. If the clearing radius is set to $\sigma = n/2$, the (μ+1) EA with clearing will create a new niche, and from there it is easy to reach the complementary optimum $\overline{x^*}$. In fact, our analyses from Section 4.3.2 never exploited the exact fitness values of TwoMax; we only used information about basins of attraction, and that it is easy to locate peaks via hill climbing. We conclude our findings in the following corollary.

Corollary 4.21. *The expected time for the* $(\mu+1)$ EA *with genotypic clearing,* $\kappa \in \mathbb{N}$, $\mu \geq \kappa n^2/4$, $\mu \leq \text{poly}(n)$ *and* $\sigma = n/2$ *finding both peaks on any nearest peak function* JZ_1 *with two complementary peaks* $p_2 = \overline{p_1}$ *is* $\text{O}(\mu n \log n)$.

If $\mu \leq n/(4\log^3 n)$, $\kappa = 1$ *and* $\sigma = n/2$, *and the* $(\mu+1)$ EA *starts with a population containing only search points that have evolved from one optimum* x^* *within the last* $\mu n/32$ *generations, the probability that both optima are found within time* $n^{(\log n)/2}$ *is* $n^{-\Omega(\log n)}$.

4.3.3.2 Weighted Nearest Peak Functions

For JZ_2 things are different: the larger the peak, the larger is its influence area of the search space in comparison to smaller peaks and thus will have a larger basin of attraction. These asymmetric variants with suboptimal peaks with smaller basin of attraction and peaks with larger basin of attraction are similar to the analysis made in Section 4.3.3.1, as long as the parameter σ is set as the maximum distance between the peaks necessary to form as many niches as there are peaks in the solution, and the restriction $0 \leq d \leq n/2$ of Lemma 4.14 is met, the same analysis can be applied for this instance of the family of landscapes benchmark.

According to JZ_2 in Definition 2.14, the bigger the height of the peak, the bigger its influence on the search space in comparison to the smaller peaks. Let $\mathscr{B}_i$ denote the basin of attraction of the highest peak p_i, as long as $0 \leq \mathscr{B}_i \leq n/2$ from Lemma 4.14 it will be possible to escape from the influence of p_i and create a new winner from a new niche with distance $\text{H}(x, p_i) \geq \mathscr{B}_i$. Jansen and Zarges show (Jansen and Zarges, 2016, Theorem 2) that for two complementary peaks $p_2 = \overline{p_1}$ the basin of attraction of p_1 contains all search points x with

$$n - \text{H}(x, p_1) < \frac{a_2}{a_1 + a_2} \cdot n + \frac{b_2 - b_1}{a_1 + a_2}.$$

Using that the peaks are complementary, a symmetric statement holds for $\mathscr{B}_2$. Note that in the special case of $a_1 = a_2$ and $b_1 = b_2$ the right-hand side simplifies to $n/2$.

Along with our previous upper bound on TwoMax from Theorem 4.16 it is easy to show the following result for a large class of weighted nearest peak functions JZ_2.

Theorem 4.22. *For all weighted nearest peak functions* JZ_2 *with two complementary peaks* $p_2 = \overline{p_1}$ *meeting the following conditions on* $a_1, a_2 \in \mathbb{R}^+$ *and* $b_1, b_2 \in \mathbb{R}_0^+$ *and the clearing radius* σ

$$\text{JZ}_2(p_1) \leq \text{JZ}_2(p_2) \;\Rightarrow\; \frac{a_1}{a_1 + a_2} \cdot n + \frac{b_1 - b_2}{a_1 + a_2} \leq \sigma \leq \frac{n}{2}$$
$$\text{JZ}_2(p_2) \leq \text{JZ}_2(p_1) \;\Rightarrow\; \frac{a_2}{a_1 + a_2} \cdot n + \frac{b_2 - b_1}{a_1 + a_2} \leq \sigma \leq \frac{n}{2}$$

the expected time for the (μ+1) EA *with genotypic clearing,* $\kappa \in \mathbb{N}$, $\mu \geq \kappa \cdot \frac{\sigma n - 2\sigma + 2}{n - 2\sigma + 2}$, $\mu \leq \text{poly}(n)$ *and clearing radius* σ *finding all global optima of* JZ_2 *is* $\text{O}(\mu n \log n)$.

Note that in case $\text{JZ}_2(p_1) \neq \text{JZ}_2(p_2)$ there is only one global optimum: the fitter of the two peaks. Then the respective condition (where the left-hand side inequality is true) implies that the basin of attraction of the less fit peak must be bounded by $n/2$. If this condition is not satisfied, the function is deceptive as the majority of the search space leads towards a non-optimal local optimum.

In case $\text{JZ}_2(p_1) = \text{JZ}_2(p_2)$ both peaks are global optima and the conditions require that both basins of attraction have size $n/2$:

$$\sigma = \frac{a_1}{a_1 + a_2} \cdot n + \frac{b_1 - b_2}{a_1 + a_2} = \frac{a_2}{a_1 + a_2} \cdot n + \frac{b_2 - b_1}{a_1 + a_2} = \frac{n}{2}.$$

Proof of Theorem 4.22. The proof is similar to the proof of Theorem 4.16. Assume without loss of generality that $\text{JZ}_2(p_1) \leq \text{JZ}_2(p_2)$. Using the same arguments as in said proof (with straightforward changes to the fitness-level calculations), the (μ+1) EA finds one peak in expected time $\text{O}(\mu n \log n)$. If this is p_1, the (μ+1) EA still needs to find p_2. By the same arguments as in the proof of Theorem 4.16, the (μ+1) EA's population will contain κ copies of p_1 in expected time $\text{O}(\mu \log n)$. Applying Lemma 4.14 with $d = \sigma$ yields that the expected time to find a search point x with Hamming distance at least σ to p_1 is $\text{O}(\mu n \log n)$. Since $\frac{a_1}{a_1 + a_2} \cdot n + \frac{b_1 - b_2}{a_1 + a_2} \leq \sigma$, by Theorem 2 in Jansen and Zarges (2016), x is outside the basin of attraction of p_1. As it is also a winner in a new niche, this new niche will never be removed, and p_2 can be reached by hill climbing on a ONEMAX-like slope from x. By previous arguments, p_2 will then be found in expected time $\text{O}(\mu n \log n)$. □

As a final remark, the analysis has shown that it is possible to escape of the basin of attraction of the higher peak with $\mathscr{B} \leq n/2$, this does not mean that the analysis cannot be applied to $\mathscr{B} \geq n/2$. We need to remember that the current investigation considers a distance $d \leq n/2$ because any distance larger than $n/2$ may lead to a exponential expected time in n for reaching a distance of at least d from x^*. One way to avoid this limitation is by dividing the distance d into several niches by setting the parameter $\sigma \leq n/2$ properly. In this analysis we just considered the population dynamics and its ability of escaping a basin of attraction of at most $n/2$ or escaping from a niche with radius at most $n/2$ but it may be possible to generalise the population dynamics for more than 2 niches with sizes $\leq n/2$ by changing/adapting our definition of the potential function. For the time being we rely on experiments in Section 4.3.4.3 to show that the population can jump from niches with $\sigma \leq n/2$ allowing to find both optimum in different variants of TWOMAX from the

classes of example functions and leave the generalisation of the population dynamics for future theoretical work.

4.3.4 Experimental Analysis

The experimental approach is focused on the analysis of the (μ+1) EA with clearing (Algorithm 26) and is divided in 3 experimental frameworks. Section 4.3.4.1 is focused on an empirical analysis for the general behaviour of the algorithm, the relationship between the parameters σ, κ, and μ, and how these parameters can be set. The main objective is to compare our asymptotic theoretical results with empirical data for concrete parameter values.

For the second empirical analysis (Section 4.3.4.2), we focus our attention on the population size for small ($n = 30$) and large ($n = 100$) problem sizes. The objective is to observe whether smaller population sizes than $\mu = \kappa n^2/4$ are capable of optimising TWOMAX and observe if the quadratic dependence on n is an artefact of our approach. Also we compare two different forms of initialising the population: the standard uniform random initialisation against a biased initialisation where the whole population is initialised with copies of one peak (0^n for TWOMAX). Biased initialisation is used in order to observe how clearing is able to escape from a local optimum and how fast it is compared to a random initialisation.

Finally, for the third analysis (Section 4.3.4.3), we show that it is possible to escape from different basin of attractions for weighted peak functions with two peaks in cases where the two peaks are not complementary, but have different Hamming distances.

4.3.4.1 General Behaviour

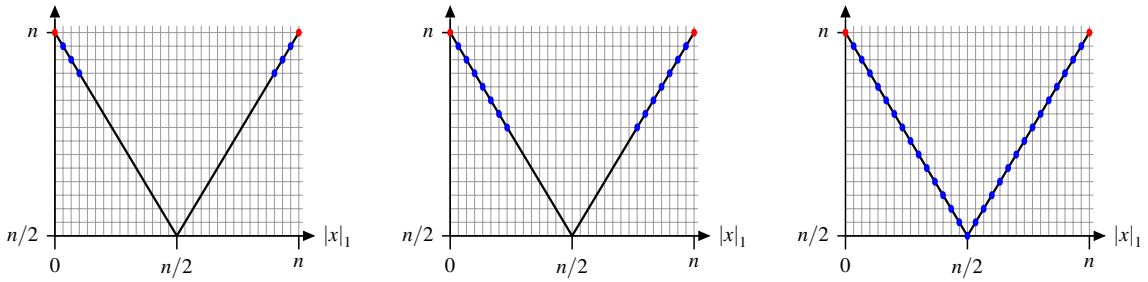
We are interested in observing if the (μ+1) EA with clearing is able to find both optima on TWOMAX, so we consider exponentially increasing population sizes $\mu \in \{2,4,8,\ldots,1024\}$ for just one size of $n = 30$ and perform 100 runs with different settings of parameters σ and κ, so for this experimental framework, we have defined $\sigma \in \{1,2,\sqrt{n},n/2\}$, $\kappa \in \{1,\sqrt{\mu},\mu/2,\mu\}$ with phenotypic distance since it has been proven that this distance metric works for both cases, small and large niches (when the genotypic distance is used it will be explicitly mentioned).

Since we are interested in proving how good/bad clearing is, we define the following outcomes and stopping criteria for each run. *Success*, the run is stopped if the population contains both 0^n and 1^n in the population. *Failure*, once the run has reached 1 million generations and not all optima are not contained in the population. All the results are shown in Table 4.2.

Table 4.2: Success rate measured among 100 runs for the (μ+1) EA with phenotypic clearing on TwoMax for $n = 30$ for the different parameters clearing radius σ, niche capacity κ and *population size* μ.

$\sigma = 1$										
κ	μ									
	2	4	8	16	32	64	128	256	512	1024
1	0.0	0.05	0.96	1.0	1.0	1.0	1.0	1.0	1.0	1.0
$\sqrt{\mu}$	0.0	0.0	0.0	0.38	0.89	1.0	1.0	1.0	1.0	1.0
$\mu/2$	0.0	0.0	0.0	0.0	0.04	0.20	0.42	0.77	0.97	0.98
μ	0.0	0.0	0.0	0.0	0.01	0.13	0.24	0.55	0.75	0.94
$\sigma = 2$										
κ	μ									
	2	4	8	16	32	64	128	256	512	1024
1	0.02	0.88	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
$\sqrt{\mu}$	0.01	0.03	0.55	0.99	1.0	1.0	1.0	1.0	1.0	1.0
$\mu/2$	0.0	0.0	0.0	0.0	0.07	0.18	0.48	0.67	0.93	0.99
μ	0.0	0.0	0.0	0.0	0.00	0.04	0.25	0.60	0.80	0.97
$\sigma = \sqrt{n}$										
κ	μ									
	2	4	8	16	32	64	128	256	512	1024
1	0.33	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
$\sqrt{\mu}$	0.35	0.67	0.97	1.0	1.0	1.0	1.0	1.0	1.0	1.0
$\mu/2$	0.40	0.78	0.95	1.0	1.0	1.0	1.0	1.0	1.0	1.0
μ	0.0	0.0	0.0	0.0	0.0	0.07	0.28	0.50	0.80	0.93
$\sigma = n/2$										
κ	μ									
	2	4	8	16	32	64	128	256	512	1024
1	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
$\sqrt{\mu}$	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
$\mu/2$	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
μ	0.0	0.0	0.0	0.0	0.02	0.12	0.29	0.60	0.81	0.98

With sufficiently many individuals $\mu = (n/2+1)\cdot\kappa$, and for small values of $\sigma \in \{1,2\}$ and $\kappa = 1$, every individual can create its own niche, and since only one individual is allowed to be the winner, the individuals are spread in the search space reaching both optima with 1.0 success rate. In this scenario, since we are allowing sufficiently many individuals in the population, individuals can be initialised in any of both branches, climbing down the branch reaching the opposite branch and reaching the other extreme optima as shown in Figure 4.3 (in this case we only show the behaviour of the population with $\mu \in \{8,16,32\}$, since $\mu \geq 8$ have the same behaviour).



(a) $\mu = 8$ (b) $\mu = 16$ (c) $\mu = 32$

Figure 4.3: Snapshot of a population at the time both optima were reached, showing the spread of individuals in branches of TwoMax for $n = 30$, $\sigma = 1$ and $\kappa = 1$. Where the red (extreme) points represent optimal individuals, blue points represent niche winners. The rows on the grid represents the fitness value of an individual and its position on TwoMax and the vertical lines represent the partitioned search space (niches) created by the parameter σ.

The previous experiment set-up confirms what is mentioned in Section 4.3.1.1: with a small clearing radius, niche capacity and large enough *population size*, the algorithm is able to exhaustively explore the search space without losing the progress reached so far. The population size provides enough pressure to optimise TwoMax. In this scenario the small differences between individuals allow the algorithm to discriminate between the two branches or optima, this enforces having individuals on both branches or occupy all niches supporting the statement of Theorem 4.10. Individuals with the same phenotype may have a large Hamming distance, creating winners with the same fitness (as proved by Corollary 4.11 in Section 4.3.1.2).

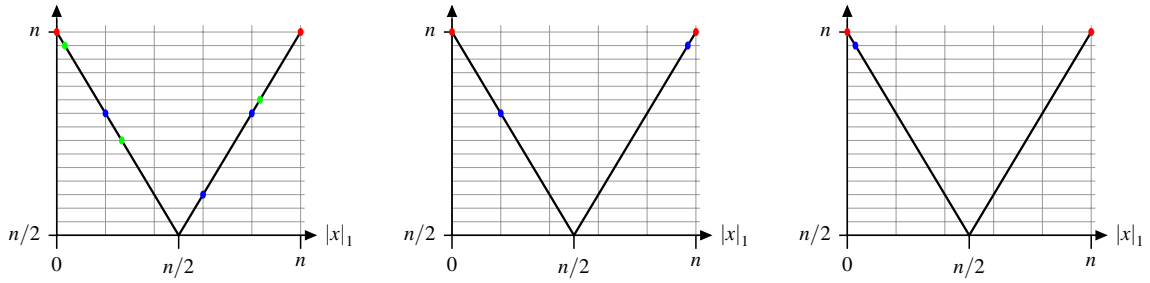
It is mentioned in Pétrowski (1996) that while the value of the niche capacity $\kappa > \mu/2$ approaches the size of the population, the clearing effect vanishes and the search becomes a standard EA. This effect is verified in the present experimental approach. For a large κ and $\sigma \in \{1,2\}$, one branch takes over, removing the individuals on the other branch reducing the performance of the algorithm. In order to avoid this, it is necessary to define $\mu \geq (n+1)\cdot\kappa$ to occupy all the winners slots and create new winners in other niches (Theorem 4.10) or

increase the clearing radius to $\sqrt{n} \leq \sigma \leq n/2$ in order to let more individuals participate in the niche. A reduced niche capacity $1 \leq \kappa \leq \sqrt{\mu}$ seems to have a better effect exploring both branches.

Now that theory and practice have shown that a small clearing radius, niche capacity and large enough population size are able to optimise TwoMax, and in order to avoid the takeover of a certain branch due to a large niche capacity it is necessary to either have: (1) a large enough population, or (2) to increase the clearing radius. For (1) we already have defined and proved that one way to overcome this scenario is to define μ according to Theorem 4.10.

In the case of large niches (2), with $\sigma \in \{\sqrt{n}, n/2\}$ it is possible to divide the search space in fewer niches. Here the individuals have the opportunity to move, change inside the niche, reach other niches allowing the movement between branches, reaching the opposite optimum. Since it is possible to reach other niches, defining the niche capacity $\kappa = \sqrt{\mu}$ will allow to have more winners in each niche but will still allow to move inside the niche.

For example, with $\sigma = \sqrt{n}$, $\kappa = \sqrt{\mu}$ and $\mu \geq 8$, the algorithm is able to reach both optima with at least 0.97 success rate. In Figure 4.4 the effect of κ can be seen with sufficiently many individuals. With a restrictive niche capacity (Figure 4.4a), the population is scattered across the search space but when the niche capacity is increased, the spread is reduced as we allow more individuals to be part of each niche (Figure 4.4b and 4.4c). This behaviour can be generalised and is more evident for larger values of μ.



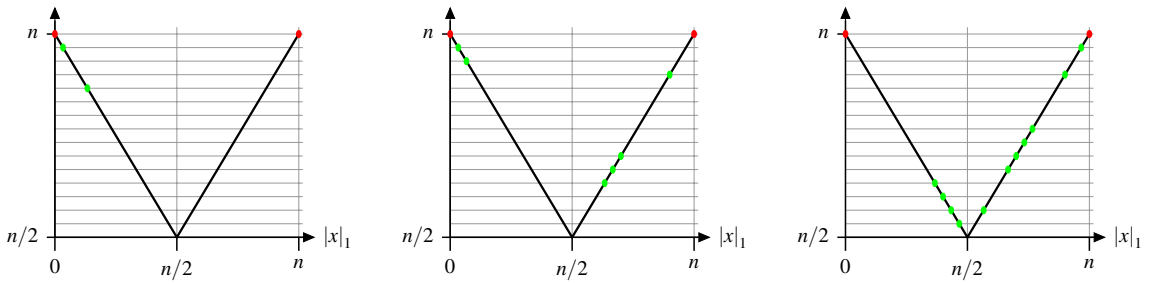
(a) $\kappa = 1$ (b) $\kappa = \sqrt{\mu}$ (c) $\kappa = \mu/2$

Figure 4.4: Snapshot of a population at the time both optima were reached, showing the spread of individuals in branches of TwoMax for $n = 30$, $\sigma = \sqrt{n}$ and $\mu = 8$. Where the red (extreme) points represent optimal individuals, blue points represent niche winners, and the green points represent cleared individuals. The rows on the grid represents the fitness value of an individual and its position on TwoMax and the columns represent the partitioned search space (niches) created by the parameter σ.

The theoretical results described in Section 4.3.2 are confirmed by the previous experimental results; a large enough population is necessary in order to fill the positions of the

winner x^* with κ winners, then force those κ winners with the rest of the population to be subject to a random walk, where it is just necessary for at least one individual to reach the next niche as mentioned in Section 4.3.2.1. In the case of TwoMax, it is after the repetition of moving, climbing down through different niches for a certain period of time when some individual is able to reach both optima as mentioned in Theorem 4.16 and confirmed by the experiments.

Now we have defined the conditions where the algorithm is able to optimise TwoMax, we can set-up the parameters in a more informed/smart way. With $\mu \geq 2$ it is possible to optimise TwoMax if σ and κ are chosen appropriately. For example, with $\sigma = n/2$ (as the minimum distance required to distinguish between one branch and the other), $\kappa \in \{1, \sqrt{\mu}, \mu/2\}$ and $\mu \geq 2$, the algorithm is able to optimise TwoMax because there is always an individual moving around that is able to reach a new niche (Figure 4.5), and finally achieve 1.0 success rate.



(a) $\mu = 8$ (b) $\mu = 16$ (c) $\mu = 32$

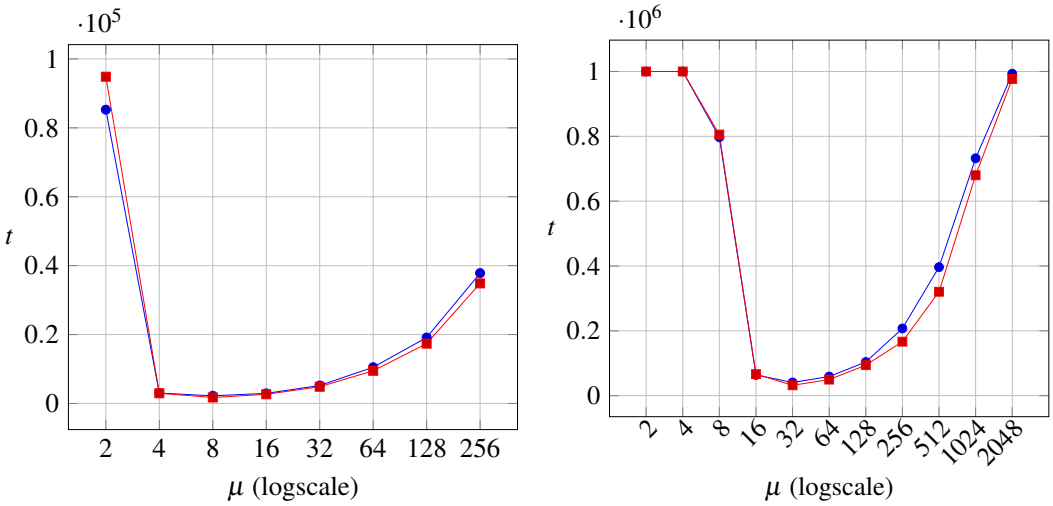
Figure 4.5: Snapshot of a population at the time both optima were reached, showing the spread of individuals in branches of TwoMax for $n = 30$, $\sigma = n/2$ and $\kappa = 1$. Where the red (extreme) points represent optimal individuals, blue points represent niche winners, and the green points represent cleared individuals. The rows on the grid represents the fitness value of an individual and its position on TwoMax and the columns represent the partitioned search space (niches) created by the parameter σ.

4.3.4.2 Population Size

In this section we address the limitation of Theorem 4.16 related to the steep requirement of the population size: $\mu \geq \kappa n^2/4$. As observed from Section 4.3.4.1, experiments suggest that a smaller population size is able to optimise TwoMax. So for the analysis of the population size we have considered the population size $2 \leq \mu \leq \kappa n^2/4$ in order to observe what is the minimum population size below the threshold $\kappa n^2/4$ able to optimise TwoMax. With $\sigma = n/2$, and $\kappa = 1$ for $n \in \{30, 100\}$ with phenotypic clearing, we report the average of generations of 100 runs. The run is stopped if both optima have been found or the algorithm

has reached a maximum of 1 million generations, this is enough time for the algorithms to converge on one or both optima.

Figure 4.6a shows the average number of generations among 100 runs with $n = 30$. Even for $\mu = 2$ the average runtime is below the 1 million threshold, hence some of the runs were able to find both optima on TwoMax in fewer than 1 million generations. The reason for this high average runtime is because once both individuals have reached one optimum, it will be one winner, and one loser subjected to a random walk until it gets replaced by an offspring of the winner. This process will continue until the loser reaches a Hamming distance of $n/2$ from the optimum to escape of the basin of attraction. Once this is achieved, it is just necessary for the individual to climb the other branch.



(a) $n = 30$ (b) $n = 100$

Figure 4.6: The average number of generations measured among 100 runs at the time both optima were found on TwoMax or $t = 1$ million generations have been reached for $n = 30$ and $n = 100$, $\sigma = n/2$, $\kappa = 1$ and $2 \leq \mu \leq \kappa n^2/4$ for the populations with randomised and biased initialisation (blue and red line, respectively).

Most importantly, all population sizes in the interval $2 \leq \mu \leq \kappa n^2/4$ are able to optimise TwoMax; this experimental setting shows that with a smaller population size for a relatively small $n = 30$ the algorithm is able to optimise TwoMax. Another interesting characteristic of the algorithm is its capacity for escaping from a local optimum.

For $n = 100$, in Figure 4.6b it is more evident that with a small population size, it is not possible to escape from the basin of attraction of a peak, and the takeover happens before the population has the chance to evolve a distance of at least $n/2$ confirming the theoretical arguments described in Section 4.3.2.3. Once the population size is increased, the population is able to escape of the basin of attraction. The most interesting result shown in Figure 4.6

is that even with a population size $\mu \leq \kappa n^2/4$ the algorithm is able to find both optima on TwoMax (even if runs require more than 1 million generations), indicating that the quadratic dependence on n in $\kappa n^2/4$, is an artefact of our approach.

Finally, for larger μ sizes and $n = 100$ it can be seen that biased initialisation is noticeably faster than random initialisation, and as the population grows the difference between the means grows. One reason could be simply because one peak has already been found, and the algorithm only needs to find the remaining peak.

4.3.4.3 Escaping from Different Basins of Attraction

Finally, in this section we show that the runtime analysis used in Section 4.3.2.1, and used to prove the theoretical analysis on the general classes of example landscapes functions in Section 4.3.3 can be used for the Jansen-Zarges weighted peak functions (see Definition 2.14) with two peaks of different Hamming distances. For simplicity we restrict our attention to equal slopes and heights: $a_1 = a_2 = 1$ and $b_1 = b_2 = 0$.

We can simplify this class of JZ_2 functions by using that the (μ+1) EA is *unbiased* as defined by Lehre and Witt (2012): simply speaking, the algorithm treats all bit values and all bit positions in the same way. Hence we can assume without loss of generality that $p_1 = 0^n$. We can further imagine shuffling all bits such that $p_2 = 0^{n-\mathrm{H}(p_1,p_2)}1^{\mathrm{H}(p_1,p_2)}$, which again does not change the stochastic behaviour of the (μ+1) EA. Then all JZ_2 functions with $a_1 = a_2 = 1$ and $b_1 = b_2 = 0$ are covered by choosing $p_1 = 0^n$ and p_2 from the set $\{0^n, 0^{n-1}1, 0^{n-2}1^2, \ldots, 1^n\}$. As can be seen the peaks p_1 and p_2 can be as close as 0^n and $0^{n-1}1$, or as far as 0^n and 1^n.

It contrast to the simple setting of TwoMax where $\sigma = n/2$ makes most sense, in this more general setting it is necessary to define the clearing radius σ according to the Hamming distance between peaks. In particular, the following conditions should be satisfied.

1. $\sigma \leq \mathrm{H}(p_1, p_2)$ as otherwise one peak is contained in the clearing radius around the other peak,
2. $\sigma \leq n/2$ as otherwise a niche can contain the majority of search points in the search space, leading to potentially exponential times to escape from the basin of attraction of a local optimum if $\sigma \geq (1 + \Omega(1)) \cdot n/2$, and
3. $\sigma \geq \mathrm{H}(p_1, p_2)/2$ as this is the minimum distance that distinguishes the two peaks.

In the following we study two different choices of σ: the maximum value that satisfied the above constraints, $\sigma = \min\{\mathrm{H}(p_1, p_2), n/2\}$, and the minimum feasible value, $\sigma = \mathrm{H}(p_1, p_2)/2$. These two choices allow us to investigate the effect of choosing large or small

niches in this setting. We use genotypic clearing with $\kappa = 1$ and we make use of the results from Section 4.3.4.2 to define $\mu = 32$ as a population size able to optimise TwoMax for large $n = 100$. We report the average number of generations of 100 runs, with the same stopping criterion: both optima have been found or the algorithm has reached a maximum of 1 million generations.

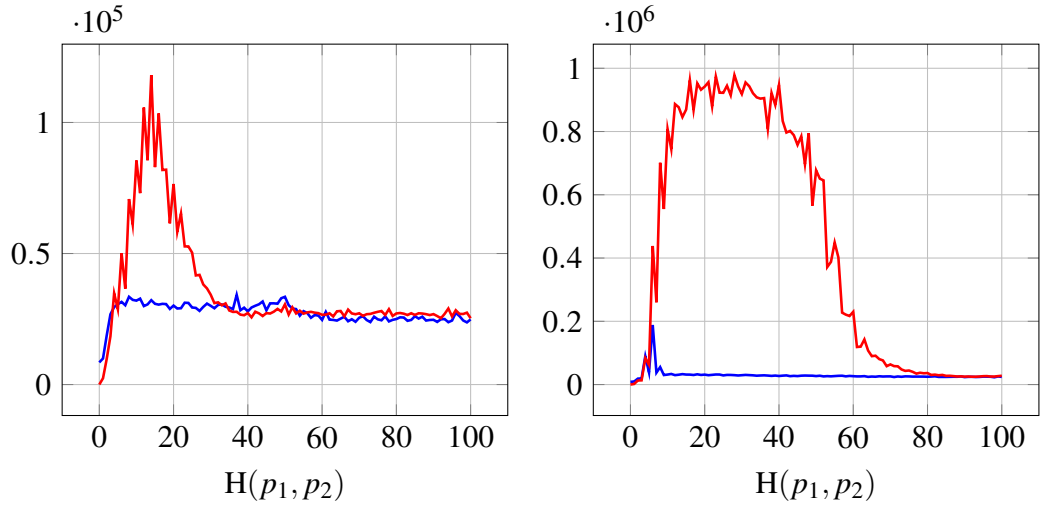
For the case of large clearing radius, $\sigma = \min\{\mathrm{H}(p_1, p_2), n/2\}$, Figure 4.7a shows that it is possible to find both optima efficiently across the whole range of $\mathrm{H}(p_1, p_2)$. For random initialisation there are hardly any performance differences, except for a drop in the runtime when the two peaks get very close. For the case of biased populations we see differences by a small constant factor: the closer the peaks, the more difficult it is to escape (or find the new niche) since it requires to flip a specific number of bits to find the other optima. But as the two peaks move away both initialisation methods seem to behave the same indicating that the arguments used in Sections 4.3.2.1 and 4.3.3 reflect correctly how the algorithm behaves.

Figure 4.7b shows that with the smallest feasible clearing radius $\sigma = \mathrm{H}(p_1, p_2)/2$ the algorithm is still able to find both optima for all $\mathrm{H}(p_1, p_2)$, but the average runtime for biased initialisation is much higher compared to $\sigma = \min\{\mathrm{H}(p_1, p_2), n/2\}$. From observing the actual population dynamics during a run, it seems that the reason for this high number of generations is because several niches are created around both peaks, i. e., once a peak has been found (and a niche is formed around the peak), the population spreads out by forming many niches between p_1 and p_2. In the case of biased initialisation it is necessary to jump between specific niches to reach the opposite peak, or make several jumps between different niches in order to escape from its basin of attraction, which leads to this high number of generations.

4.3.5 Conclusions

The presented theoretical and empirical investigation has shown that clearing possesses desirable and powerful characteristics. We have used rigorous theoretical analysis related to its ability to explore the landscape in two cases, small and large niches, and provide an insight into the behaviour of this diversity-preserving mechanism.

In the case of small niches, we have proved that clearing can exhaustively explore the landscape when the proper distance and parameters like clearing radius, niche capacity and population size μ are set. Also, we have proved that clearing is powerful enough to optimise all functions of unitation. In the case of large niches, clearing has been proved to be as strong as other niching mechanisms like deterministic crowding and fitness sharing since it is able to find both optima of the test function TwoMax.



(a) $\sigma = \min\{\mathrm{H}(p_1,p_2), n/2\}$ (large niches) (b) $\sigma = \mathrm{H}(p_1,p_2)/2$ (small niches)

Figure 4.7: The average number of generations measured among 100 runs at the time both peaks $p_1 = 0^n$ and $p_2 \in \{0^n, 0^{n-1}1, 0^{n-2}1^2, \ldots, 1^n\}$ were found with $a_1 = a_2 = 1$ and $b_1 = b_2 = 0$ on the fitness landscape defined by JZ_2 or have reached $t = 1$ million generations for $n = 100$, with genotypic clearing with $\sigma = \min\{\mathrm{H}(p_1,p_2), n/2\}$ and $\sigma = \mathrm{H}(p_1,p_2)/2$, $\kappa = 1$ and $\mu = 32$ for populations with randomised (blue line) and biased (red line) initialisation.

The analysis made has shown that our results can be easily extended to more general classes of examples landscapes. The analysis done for TwoMax can easily be applied to different classes of bimodal problems using arguments based on how to escape the basin of attraction of one local optimum. We demonstrated this for functions with two complementary peaks and asymmetric variants of TwoMax, consisting of a suboptimal peak with a smaller basin of attraction and an optimal peak with a larger basin of attraction.

Our experimental results suggest that the same efficient performance also applies to bimodal functions where the two peaks have varying Hamming distances. Here clearing is able to escape from local optima with different basins of attractions by moving/jumping between niches formed by the clearing radius. Defining σ as the smallest possible value that allows to distinguish between peaks creates several small niches, forcing the individuals in the population to make several jumps between niches until an individual can reach the basin of attraction of the other peak. This means that the algorithm requires more generations to find both peaks. But if σ is defined as the maximum feasible value, $\sigma = \min\{\mathrm{H}(p_1,p_2), n/2\}$, the $(\mu{+}1)$ EA is faster and remarkably robust with respect to the Hamming distances between the two peaks. Nevertheless, both approaches allow the population to escape from different basin of attractions.

It remains an open problem to theoretically analyse the population dynamics of clearing with more than 2 niches and to prove rigorously that clearing is effective across a much broader range of problems, including problems with more than 2 peaks. This involves obtaining more detailed insights into the dynamics of the population, including the distribution and evolution of the losers across multiple niches.

Chapter 5

Empirical Analysis of Diversity Mechanisms for Multimodal Optimisation

This chapter is based on the following publication:

1. Covantes Osuna, E. and Sudholt, D. (2018a). Empirical Analysis of Diversity-Preserving Mechanisms on Example Landscapes for Multimodal Optimisation. In *Parallel Problem Solving from Nature – PPSN XV*, pages 207–219. Springer International Publishing.

As mentioned in the previous chapter, finding global optima or high-quality local optima can become a challenge for any optimisation algorithm in multimodal problems (Sareni and Krahenbuhl, 1998; Singh and Deb, 2006). So the use of a diverse population on an EA can be useful to explore several hills in the fitness landscape simultaneously and offer several good solutions to the user, a feature desirable for decision making, in multi-objective optimisation and in dynamic optimisation. In this chapter we turn to an empirical analysis.

Previous empirical analyses have considered real-parameter multimodal optimisation problems (Das et al., 2011) like the 4 one-dimensional, five-peaked, sinusoidal, multimodal functions called M_{1-4} defined in (Mahfoud, 1995, Section 5.3). The single variable x is restricted to the real-value range $[0,1]$ encoded using binary representation and decoded by interpreting the bitstring as unsigned binary integer and dividing it by $2^n - 1$, where n is the length of the bit string. Other studies used Gray codes (Sareni and Krahenbuhl, 1998). The drawback of real-valued encodings is that it is not obvious how phenotypic features such as local optima appear in genotype space; for example what Hamming distance local optima have and how likely it is that mutation jumps from one basin of attraction to another.

This makes the analysis of the population dynamics a very challenging task for theoretical analysis.

Previous theoretical studies on the TWOMAX function led to insights into the capabilities and weaknesses of various diversity mechanisms, however a question left open is how diversity mechanisms deal with many local optima. The main goal in this chapter is to provide insights into the working principles of these mechanisms by testing their ability to find and maintain many local optima in the population as well as their ability to escape from local optima with different basins of attraction. We use previous theoretical results to inform the choice of algorithm parameters and to discuss in how far our empirical results agree or disagree with theoretical results obtained for TWOMAX.

5.1 Jansen-Zarges Multimodal Function Classes

In the same way we use the Jansen-Zarges multimodal function classes in Section 4.3.3 for the case of runtime analysis of clearing, we use these function classes for our empirical analysis. Recall that for these functions it is necessary to define the following parameters: number of peaks (each i-th peak is defined by its position in the search space $p_i \in \{0,1\}^n$), its offset ($b_i \in \mathbb{R}_0^+$) and its slope ($a_i \in \mathbb{R}^+$). The general idea is that the fitness of an individual depends on the peaks in its vicinity. The function JZ_1 called *nearest peak function*, has the fitness of a search point x determined by the closest peak, and the function JZ_2 called *weighted nearest peak* uses the height of the peaks to set the fitness of an individuals, the higher the peak, higher its influence on the search space in comparison to smaller peaks.

Note that, in case of equal slopes $a_1 = \cdots = a_k$ and equal heights $b_1 = \cdots = b_k$, both functions JZ_1 and JZ_2 using parameters $a_1, \ldots, a_k, b_1, \ldots, b_k$ are identical as for JZ_2 the maximum over all terms $a_i \cdot \mathrm{G}(x, p_i) + b_i$ for all $1 \leq i \leq k$ is attained for the closest peak $i = \mathrm{cp}(x)$.

Theorem 5.1. *For* JZ_1 *and* JZ_2 *using the same parameters* $a_1 = \cdots = a_k$ *and* $b_1 = \cdots = b_k$ *we have* $\mathrm{JZ}_1 = \mathrm{JZ}_2$.

In the case of two peaks p_1 and p_2, if these peaks are complementary, that is, $p_2 = \overline{p_1}$, then JZ_1 and JZ_2 generalise the TWOMAX function, with TWOMAX being the special case of $p_1 = 0^n, p_2 = 1^n, a_1 = a_2 = 1$ and $b_1 = b_2 = 0$ (Jansen and Zarges, 2016). Theoretical and empirical results for the (μ+1) EA with clearing on two complementary peaks are given in Section 4.3.

We consider peaks being placed independently and uniformly at random, as this strategy is simple, fair, and it scales towards an arbitrary number of peaks. The slopes are chosen

equal to 1 for all peaks for the sake of simplicity. Even though the peaks are placed randomly, if the peaks have moderately similar heights, the resulting fitness landscape has a clear structure: with high probability all peaks are local optima, and all search points within a Hamming ball of radius $\Omega(n)$ belong to a peak's basin of attraction. This holds for both functions JZ_1 and JZ_2 as they have equal fitness values within the mentioned Hamming balls (but may have different values on other search points).

Theorem 5.2. *Assume k peaks $p_1, \dots, p_k$ chosen independently and uniformly at random from $\{0,1\}^n$. If $a_1 = \cdots = a_k = 1$ and $\max_{1 \le i \le k} b_i - \min_{1 \le i \le k} b_i \le cn$ for a constant $c < 1/2$ then with probability $1 - k^2 e^{-\Omega(n)}$ for radius $r := (1/2 - c)/3 \cdot n$ we have:*

1. *all k peaks $p_1, \dots, p_k$ are local optima in both JZ_1 and JZ_2,*
2. *for all $1 \le i \le k$, all search points in $\mathscr{B}_i := \{x \mid \mathrm{H}(x, p_i) \le r\}$, a Hamming ball of radius r around p_i, belong to the basin of attraction of p_i with respect to both JZ_1 and JZ_2, that is, there is a Hamming path from x to p_i on which the values of JZ_1 and JZ_2 are strictly increasing, and*
3. *for all search points $x \in \bigcup_{i=1}^{k} \mathscr{B}_i$, $\mathrm{JZ}_1(x) = \mathrm{JZ}_2(x)$.*

Proof. Assume without loss of generality $\min_{1 \le i \le k} b_i = 0$ (as adding a fixed value does not affect the problem structure), hence $b_i \le cn$ for all $1 \le i \le k$.

By Chernoff bounds (see Lemma A.15), the probability that two different peaks will have Hamming distance at most $n/2 - r$ is $e^{-\Omega(n)}$. By the union bound, the probability that this holds for any pair of peaks is at most $k^2 \cdot e^{-\Omega(n)}$. We assume in the following that every two peaks have a Hamming distance larger than $n/2 - r$.

Now consider a search point $x \in \mathscr{B}_i$, that is, $\mathrm{H}(x, p_i) \le r$. Since $r \le n/6$ we have $n/2 \ge 3r$, and thus for all $j \ne i$ we have $\mathrm{H}\big(x, p_j\big) \ge \mathrm{H}\big(p_i, p_j\big) - \mathrm{H}(x, p_i) > n/2 - r - r \ge r \ge \mathrm{H}(x, p_i)$. So p_i is a unique closest peak, $\mathrm{cp}(x) = i$. By definition of JZ_1, the second statement follows for JZ_1 as on every shortest Hamming path from x to p_i, subsequently decreasing the Hamming distance to p_i increases the fitness by $a_i = 1$. Since $r \ge 1$ if n is large enough, p_i is a local optimum for JZ_1.

It only remains to show the third statement as then the first two statements also apply to JZ_2. To prove that $\mathrm{JZ}_2(x) = \mathrm{JZ}_1(x)$ for $x \in \mathscr{B}_i$, we need to show that the maximum over terms $a_j \cdot \mathrm{G}\big(x, p_j\big) + b_j = n - \mathrm{H}\big(x, p_j\big) + b_j$ from the definition of JZ_2 is attained for $j = i$. We have $n - \mathrm{H}(x, p_i) + b_i \ge n - r$ as $\mathrm{H}(x, p_i) \le r$ and $b_i \ge 0$. For $j \ne i$ we have $n - \mathrm{H}\big(x, p_j\big) + b_j < n/2 + 2r + cn$ as $b_j \le cn$ and $\mathrm{H}\big(x, p_j\big) \ge \mathrm{H}\big(p_i, p_j\big) - \mathrm{H}(x, p_i) > n/2 - r - r = n/2 - 2r$. Noting that $n/2 + 2r + cn = n/2 + 3r + cn - r = n/2 + (n/2 - cn) + cn - r = n - r$ establishes

$n - \mathrm{H}(x, p_j) + b_j \leq n - \mathrm{H}(x, p_i) + b_i$ and hence $\mathrm{JZ}_2(x) = \max_{1 \leq j \leq k}(n - \mathrm{H}(x, p_j) + b_j) = n - \mathrm{H}(x, p_i) + b_i = \mathrm{JZ}_1(x)$. □

5.2 Experimental Analysis

For the experimental analysis we test each of the algorithms from Table 4.1 on Jansen-Zarges multimodal function classes. We consider a problem size $n = 100$, genotypic distance for all algorithms that require a dissimilarity measure and stop runs after $10\mu n \ln n$ generations as done previously in Sections 4.1.1 and 4.2.3. This time limit is motivated by Lemma 4.1 stating that, loosely speaking, $2e\mu n \ln n \approx 5.44\mu n \ln n$ generations are sufficient to perform hill climbing on two peaks with high probability (Lemma 4.1).

The experimental framework is divided in 3 experimental set-ups. In Section 5.2.1 we assess the ability of each mechanism to find many peaks with equal height, and in Section 5.2.2, we assess the ability of each mechanism to maintain the population diversity when considering peaks with different heights to yield global and local optima. For both sections, the number of peaks was increased exponentially as $k = \{2, 4, 8, \ldots, 64\}$. For each k, we generated 100 different instances choosing k peaks uniformly at random from $\{0,1\}^n$. In each experiment, all algorithms are tested on the same set of 100 instances to ensure a fair comparison. The challenge for each mechanism is to find and maintain as many peaks as possible before reaching the $10\mu n \ln n$ generations; we record the fraction of the peaks found. The population size is chosen large enough ($\mu = 100$) to be able to accommodate all peaks.

The analysis in Section 5.2.3 is inspired by Section 4.3.4.3 and focusses on landscapes with two peaks. In this section we take a closer look at the ability of the diversity mechanisms to deal with different basins of attraction, including a wider range of two-peaked landscapes than the ones likely to be generated by placing peaks uniformly at random. The goal is to observe which mechanisms are able to escape from local optima by tunnelling through the fitness valley that separates two peaks. We choose $\mu = 32$ as in Section 4.3.4.3 and also consider the same two initialisations: the standard uniform random initialisation and biased initialisation where the whole population is initialised with copies of one peak (0^n for TwoMax). Biased initialisation is used in order to observe how the mechanisms are able to escape from a local optimum and how fast it is compared to a random initialisation.

Based on the theoretical analysis in Section 4.2.1, we define the window size $w = 2.5\mu \ln n$ for RTS. We know from Friedrich et al. (2009) and Oliveto et al. (2014) that both fitness sharing approaches with phenotypic sharing and $\sigma = n/2$ are always efficient on TwoMax but no theory for genotypic sharing is available. Preliminary experiments for genotypic sharing and $\sigma = n/2$ on TwoMax yielded poor results; however with $\sigma = n$ (which implies

that all individuals always share fitness) both peaks were found in most runs. This makes sense on other landscapes as well as if σ is set smaller than the radius or basin of attraction around a local optimum, then fitness sharing is unable to push individuals away from said local optimum. Thus it seems best to err on the side of choosing σ too large rather than too small.

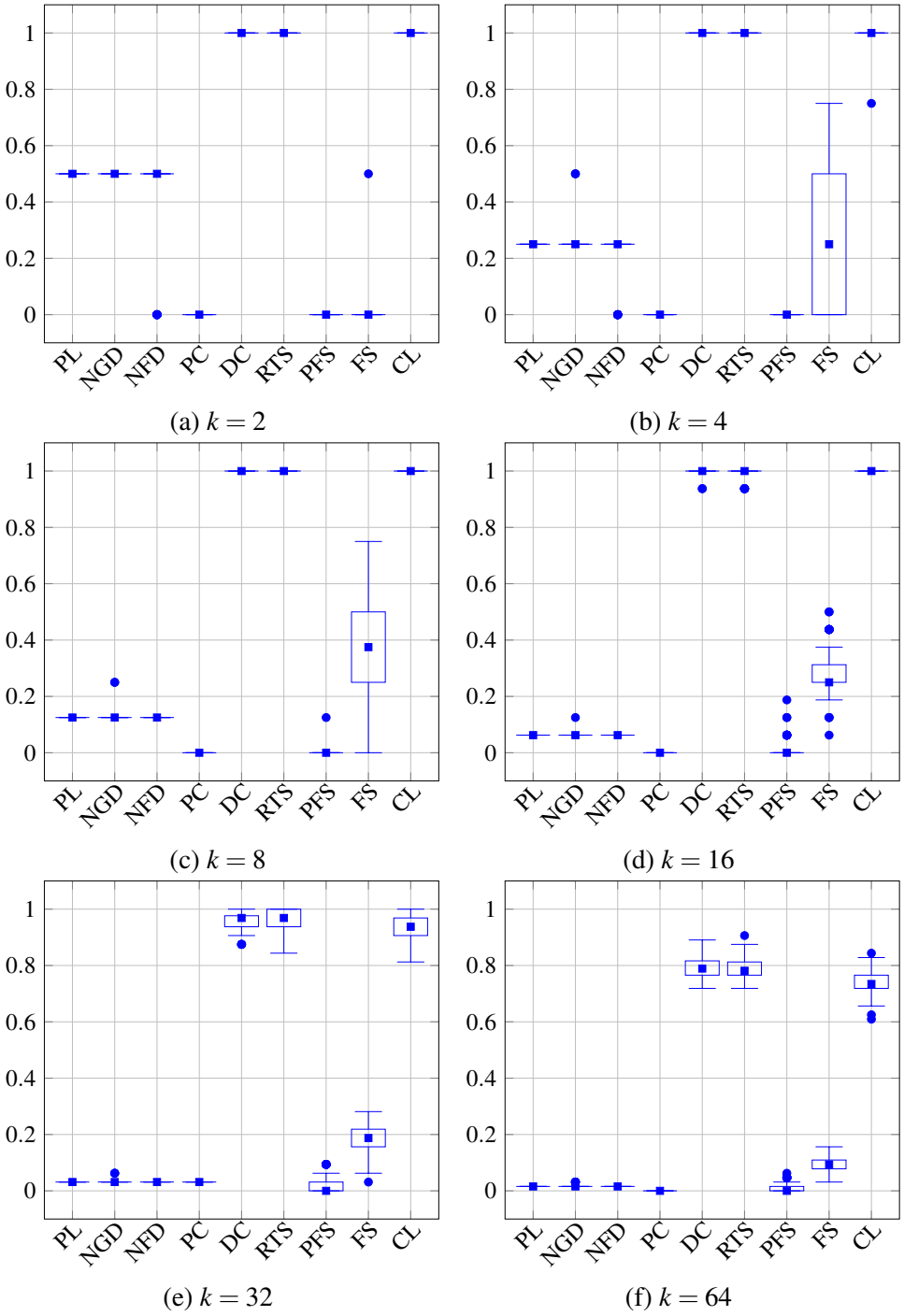
For clearing the situation is different. If σ is chosen too large, such that there are several optima within a distance of σ, then global optima may be cleared, making it impossible to maintain many optima in the population. So for clearing it seems best to err on the side of choosing σ too small rather than too large. We choose $\sigma = n/3$ for Sections 5.2.1 and 5.2.2 as with high probability every two different peaks will have a Hamming distance larger than $n/3$ (cf. Theorem 5.2). For Section 5.2.3, we use the recommendation $\sigma = \min\{\mathrm{H}(p_1, p_2), n/2\}$ from Section 4.3.4.3.

5.2.1 Finding Peaks of Equal Height

We consider the JZ_1 function with equal slopes $a_1 = \cdots = a_k = 1$ and offsets $b_1 = \cdots = b_k = 0$. We know from Theorem 5.1 that with equal parameters $\mathrm{JZ}_1 = \mathrm{JZ}_2$. In Figure 5.1, we show the fraction of peaks obtained in each of the 100 instances and its variance for each choice of k.

As can be seen, the the plain (μ+1) EA, no genotype duplicates and no fitness duplicates perform poorly; these have already been proven to perform poorly on TwoMax (Friedrich et al., 2009). Probabilistic crowding as predicted in Section 4.1 is not able to find even one peak. Fitness sharing performs best for an intermediate number of peaks, $k \in \{4, 8, 16\}$, but still far worse than the best mechanisms. This is in contrast to theoretical results (Friedrich et al., 2009; Oliveto et al., 2014) where fitness sharing in both variants was shown to be very effective on TwoMax. These differences may be down to the differences between TwoMax and JZ_1 with random peaks and/or they may be caused by the differences between phenotypic and genotypic sharing. Interestingly, population-based fitness sharing performs far worse than the conventional fitness sharing. This is surprising as population-based fitness sharing uses a significant amount of computation time to search for the best possible population (in terms of shared fitness) it can create out of all parents and offspring, hence we would have expected it to perform better than fitness sharing. A possible explanation for the poor performance of fitness sharing is that even when the population is able to locate basins of attraction of several peaks, we found several individuals scattered around each peak, apparently repelling each other and preventing each other from reaching the peak.

Finally, deterministic crowding, RTS and clearing perform surprisingly well: they find all optima most of the time for $k \leq 16$, and find most optima for $k = 32$. Only for a large number



(a) $k = 2$ (b) $k = 4$

(c) $k = 8$ (d) $k = 16$

(e) $k = 32$ (f) $k = 64$

Figure 5.1: Fraction of peaks found on JZ_1 with peaks of equal slopes $a_1 = \cdots = a_k = 1$ and offsets $b_1 = \cdots = b_k = 0$ for all (μ+1) EA variants from Table 4.1 among 100 instances generated uniformly at random for each number of peaks $k = \{2, 4, 8, \ldots, 64\}$, $\mu = 100$ and $n = 100$, stopping runs after $10\mu n \ln n$ generations. Squares indicate median values.

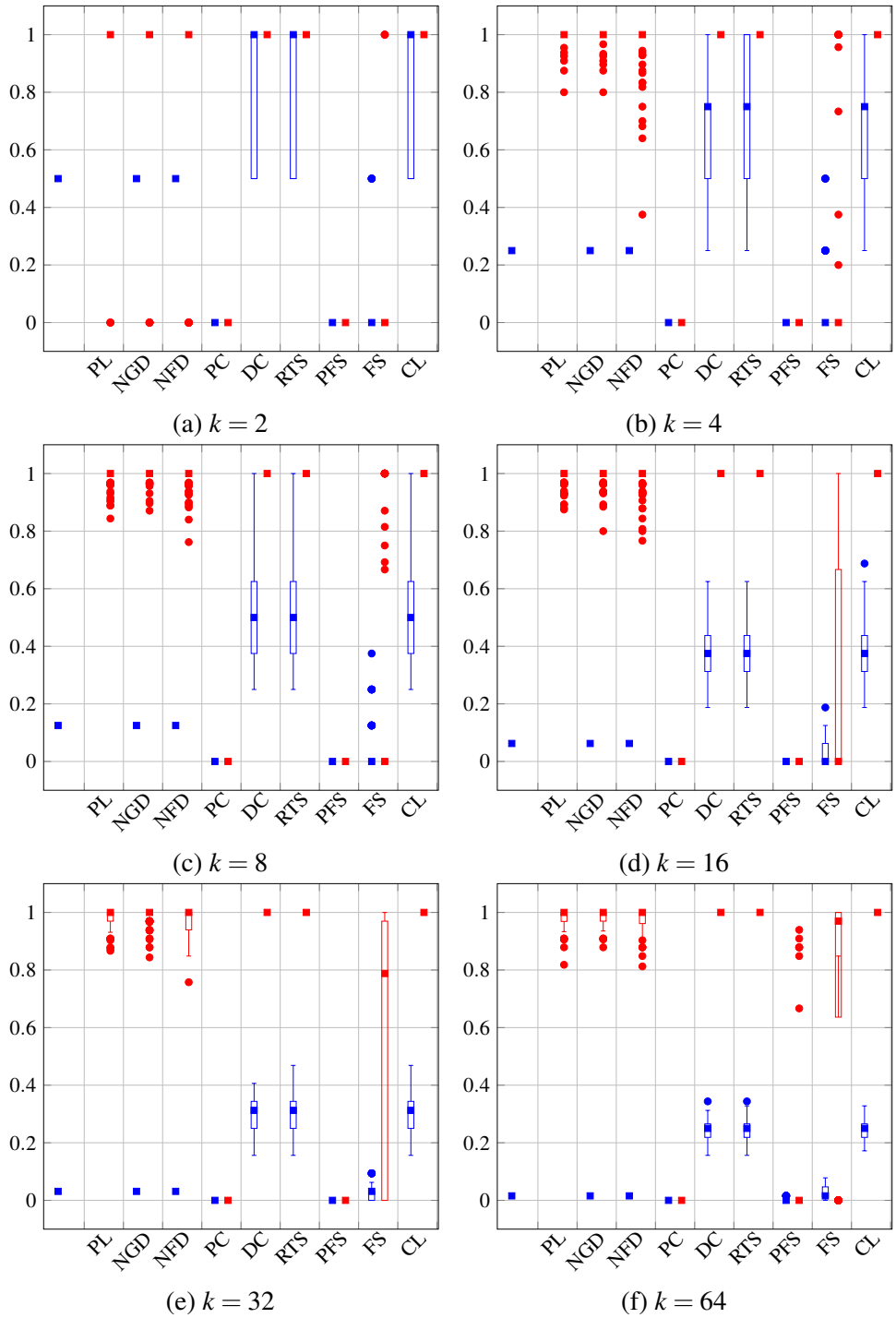
of $k = 64$ peaks, performance deteriorated to around 80% of peaks found. This deterioration is not surprising as the population size was fixed to $\mu = 100$. RTS with $w = 2.5\mu \ln n$ seems to behave similarly to deterministic crowding as predicted in Section 4.2.1.

5.2.2 Finding Peaks with Different Height

For this case we make use of the JZ_2 function with $a_1 = \cdots = a_k = 1$ and $b_1 \cdots b_k$ chosen independently and uniformly at random from $[0, 1, \ldots, n/3]$. This range is motivated by Theorem 5.2, as here two peaks differ in their heights by at most $n/3$, choosing the leading constant $c := 1/3$ as the simplest constant smaller than $1/2$. Theorem 5.2 then yields that all search points within Hamming balls of radius $n/18$ centred at a peak are located in the peak's basin of attraction. Figure 5.2 (blue/left box plots) shows the fraction of peaks obtained in each of the 100 instances and its variance for each $k = \{2, 4, 8, \ldots, 64\}$ peaks. To gauge the quality of the peaks found, we also plot the normalised best fitness found (red/right box plots), formally f_i^*/opt_i where f_i^* is the fitness of the best peak found on instance i and opt_i is the optimal value of instance i.

In this setting the plain (μ+1) EA, no genotype duplicates and no fitness duplicates manage to find the global optimum in up to 80% of instances. This suggests that on this function class it is fairly easy to find a global optimum. However, they rarely find more than one peak, hence they seem to suffer from premature convergence. Probabilistic crowding continues to show the worst performance of all mechanisms. Population-based fitness sharing and fitness sharing find fewer peaks on JZ_2 compared to JZ_1. This makes sense since the former setting is more difficult than the latter; both mechanisms seem to suffer from the issues mentioned in Section 5.2.1.

Finally, deterministic crowding, RTS and clearing also find fewer peaks due to the difficulty of this setting, but still show the best performance of all mechanisms analysed in this paper and they manage to find the global optimum in all instances. For $k \leq 8$ is not possible to always find all peaks any more, but they still manage to find at least 50% of the peaks. Then, for $k \geq 16$ the performance deteriorates in such a way that it is not possible to reach any more 50% of the peaks but still the mechanisms manage to find some of the peaks. The general cause of the drop in the performance seems to be that all mechanisms struggle to escape from the optimum found, also that low-quality optimums are being dropped when better peaks have been found.



(a) $k = 2$ (b) $k = 4$

(c) $k = 8$ (d) $k = 16$

(e) $k = 32$ (f) $k = 64$

Figure 5.2: Experimental results for all (μ+1) EA variants from Table 4.1 among 100 instances generated uniformly at random for each number of peaks $k = \{2, 4, 8, \ldots, 64\}$, $\mu = 100$ and $n = 100$, stopping runs after $10\mu n \ln n$ generations. Blue/left: fraction of peaks found on JZ_2 with peaks with different heights, $b_1 \ldots b_k$ chosen uniformly at random from $\{0, 1, \ldots, n/3\}$. Red/right: normalised best fitness found on JZ_2 experiments. Squares indicate median values.

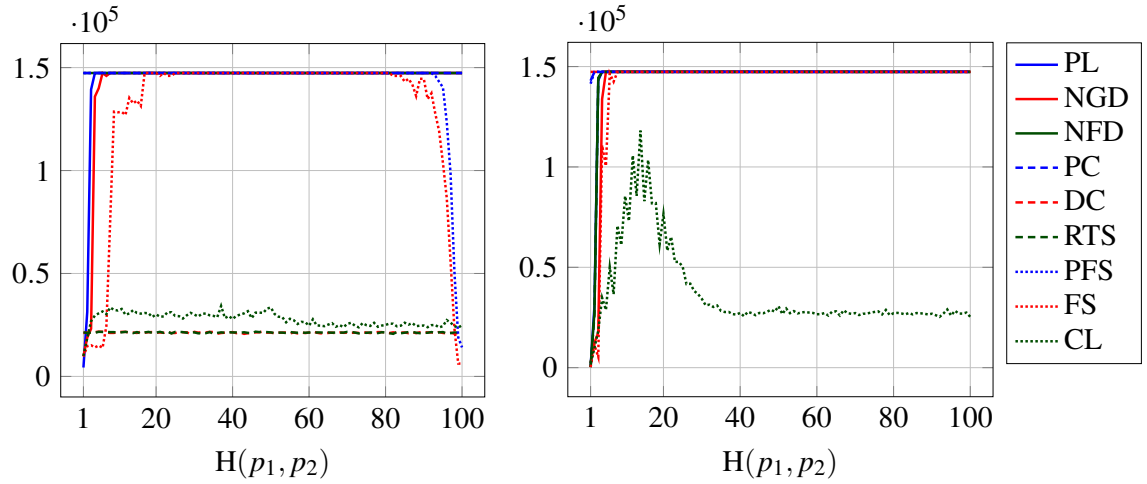
5.2.3 Escaping from Local Optima

Theorem 5.2 and its proof suggest that when peaks are chosen uniformly at random, they will have a Hamming distance close to $n/2$. We would like to investigate how the diversity mechanisms behave if peaks have different Hamming distances. Following Section 4.3.4.3, we focus on two peaks and vary their Hamming distance between 1 and n by choosing $p_1 = 0^n$ and $p_2 \in \{0^{n-1}1, 0^{n-2}1^2, \ldots, 1^n\}$, along with $a_1 = a_2 = 1$ and $b_1 = b_2 = 0$. As argued in Section 4.3.4.3, this captures the performance across all possible JZ_1 functions with two complementary peaks and the given slopes and offsets. In particular, it includes many functions that only have an exponentially small probability to be generated when choosing peaks independently and uniformly at random with biased initialisation, the algorithms have to find the other optimum by tunnelling through the fitness valley that separates these two peaks. This is a much harder task compared to hill climbing on various hills, where the aim is for the population to maintain a good spread over the search space.

We use the set-up and empirical data for clearing from Section 4.3.4.3 and report the average number of generations of 100 runs, with two stopping criteria: both optima have been found or $t = 10\mu n \ln n$ generations were reached. We also apply the two-sided Mann-Whitney U tests with a significance level of 0.05 on the data obtained from the 100 generations for each $\mathrm{H}(p_1, p_2)$.

From Figure 5.3a all mechanisms are effective when the Hamming distance is so small that the peaks are very close together such that the second peak can be found by a mutation of the first peak found (except for probabilistic crowding, that is not able to reach a single peak). But as the distance increases, the time for some mechanisms increases rapidly; they are inefficient on all non-trivial settings. Deterministic crowding and RTS seem to be agnostic of Hamming distances as they show a very stable and equal performance across the whole range of Hamming distances. This make sense as deterministic crowding climbs up both peaks with equal probability (cf. the analysis on TwoMax in Friedrich et al., 2009) and RTS behaves similarly to deterministic crowding. Clearing is very effective and only mildly worse than deterministic crowding and RTS. Remember that the data for clearing corresponds to the one shown previously in Section 4.3.4.3, in that section, 1 million generations were used, greater than the one defined in this section ($10\mu n \ln n$). Let us note that in any run of the clearing algorithm, the 1 million and the $10\mu n \ln n$ generations threshold are never reached. We see that for fitness sharing with genotypic sharing is only effective if the peaks have a Hamming distance that is very close to n or trivially small. For intermediate values, fitness sharing fails badly.

With biased initialisation (Figure 5.3b), clearing is the only mechanism able to escape from local optima with different basins of attraction. As shown theoretically in Section 4.3,



(a) Random Initialisation

(b) Biased Initialisation

Figure 5.3: The average number of generations among 100 runs for finding both peaks $p_1 = 0^n$ and $p_2 = \{0^{n-1}1, 0^{n-2}1^2, \ldots, 1^n\}$ on the fitness landscape defined by JZ_2 with $a_1 = a_2 = 1$ and $b_1 = b_2 = 0$ or $t = 10\mu n \ln n$ generations were reached, for all (μ+1) EA variants mentioned in Table 4.1, using $n = 100$ and $\mu = 32$. Results for both random and biased initialisation are shown.

this is because cleared individuals are able to explore the fitness landscape by performing a random walk. In this case, clearing in few occasions uses more generations than the $10\mu n \ln n$ allowed, we choose to leave the data untouched since we know that the other algorithms are not able to escape of the basing of attraction and in a certain point clearing is able to escape of the basin of attraction of the peaks, so it is not an unfair comparison and still can be used to show the better performance of clearing. We know from Friedrich et al. (2009) and Oliveto et al. (2014) that both fitness sharing approaches with phenotypic sharing and sharing radius $\sigma = n/2$ are able to escape from local optima as well, if the two peaks are complementary. With genotypic sharing both fitness sharing approaches perform very poorly and seem unable to escape from local optima. Also the other mechanisms fail as they are unable to accept worse search points.

The analysis of the algorithms with the Mann-Whitney U test are divided into 2 categories, "good" and "bad" algorithms. From Figure 5.3a, it is clear that the good algorithms are deterministic crowding, RTS and clearing since the three have a better performance than the rest of the algorithms, so it makes sense to compare them aside from the other algorithms that behave poorly. In the case of the bad algorithms we have the plain (μ+1) EA, no genotype and phenotype duplicates, probabilistic crowding and both fitness sharing approaches. Let us concentrate on the bad algorithms without both fitness sharing methods. The Mann-Whitney U tests confirm that when the $6 \leq \mathrm{H}(p_1, p_2) \leq 100$, the performance of the plain

(μ+1) EA, no genotype and fitness duplicates are significantly worst than the good ones. Now, if we introduce the population-based fitness sharing into the bad algorithms, the range gets reduced to $6 \leq \mathrm{H}(p_1, p_2) \leq 94$. And by introducing individual-based fitness sharing we get the following range: $17 \leq \mathrm{H}(p_1, p_2) \leq 86$.

As can be seen, the fitness sharing methods have better performance than the rest of the bad individuals when the peaks can be reached by mutation or the peaks have a distance closer to n. For example, individual-based fitness sharing has significantly better performance than the good algorithms when $1 \leq \mathrm{H}(p_1, p_2) \leq 7$ and $97 \leq \mathrm{H}(p_1, p_2) \leq 100$, and the population-based fitness sharing starts working and showing significantly better performance than the good algorithms when $97 \leq \mathrm{H}(p_1, p_2) \leq 100$. Now for the case of the good algorithms, there is a significant differences in the performance of clearing and the crowding methods: deterministic crowding and RTS while no significant difference between deterministic crowding and RTS.

For biased initialisation, and as mentioned previously, some of the algorithms are able to reach the second peak using mutation, but once the peaks begin to move away, the performance of most of the algorithms start to deteriorate. Only clearing shows a significant difference in its performance compared to the other algorithms with $3 \leq \mathrm{H}(p_1, p_2) \leq 100$.

5.3 Conclusions

We have performed an extensive empirical study involving 9 common diversity mechanisms on Jansen-Zarges multimodal function classes, covering various degrees of multimodality from 2 to 64 peaks and peaks having equal or different heights, reflected in their basins of attraction.

Our results show that the plain (μ+1) EA, the simple mechanisms: avoiding genotype and fitness duplicates cannot maintain subpopulations on several peaks; once a peak has been found it seems impossible to escape from such a peak. Probabilistic crowding shows a terrible performance as it is unable to locate even a single peak. These findings are in line with theoretical results on TwoMax (Section 4.1 and Friedrich et al., 2009).

Previous theoretical results have shown that both fitness sharing approaches are always efficient on TwoMax if phenotypic distances are being used and parameters are set appropriately (Friedrich et al., 2009; Oliveto et al., 2014). This includes the ability to climb down a peak and to tunnel through fitness valleys to reach other niches. Unfortunately this is not the case for fitness sharing with genotypic distance. Only when the peaks have a Hamming distance that is trivially small or very close to n they seem to be effective; for any other intermediate case they show a poor performance.

Deterministic crowding, restricted tournament selection and clearing perform well for peaks with the same slope and height, much better than all other diversity mechanisms. Only for large numbers of peaks ($k = 64$) and different heights the performance starts to deteriorate. Finally, only clearing has shown the ability to escape from local optima since all other mechanisms seem unable to accept worse search points or unable to tunnel through fitness valleys.

Part III

Runtime Analysis of Diversity Mechanisms on Multi-Objective Optimisation

Chapter 6

Diversity-based Parent Selection for Evolutionary Multi-objective Optimisation

This chapter is based on the following publications:

1. Covantes Osuna, E., Gao, W., Neumann, F., and Sudholt, D. (2017). Speeding Up Evolutionary Multi-objective Optimisation Through Diversity-based Parent Selection. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '17, pages 553–560. ACM.

2. Covantes Osuna, E., Gao, W., Neumann, F., and Sudholt, D. (2018). Design and analysis of diversity-based parent selection schemes for speeding up evolutionary multi-objective optimisation. *Theoretical Computer Science*. To appear. Preprint available from http://arxiv.org/abs/1805.01221.

In Chapters 4 and 5 we focussed on diversity through refining environmental selection (or survival selection). Based on the diversity mechanism, the EA chooses individuals to survive to the next generation. In this chapter we use diversity through refining parent selection, parents are chosen to generate offspring.

Selection plays a crucial role in the use of EAs as it sets the direction of the evolutionary process. Let us remember that an EA consists of two parts where selection of individuals is carried out. Parent selection decides on which individuals of the current population produce offspring, whereas survival selection selects the population for the next generation from the current set of parents and offspring after the offspring population has been produced.

The area of EMO designs population-based EAs where the population is used to approximate the so-called Pareto front. Given that EAs use a population which is a set of solutions

to a given problem, EAs are suited in a natural way for computing trade-offs with respect to two (or more) conflicting objective functions.

Well established MOEAs such as NSGA-II (Deb et al., 2002), SPEA2 (Bleuler et al., 2001), IBEA (Zitzler and Künzli, 2004) have two basic principles driven by selection. First of all, the goal is to push the current population close to the "true" Pareto front. The second goal is to "spread" the population along the front such that it is well covered. The first goal is usually achieved by dominance mechanisms between the search points or indicator functions that prefer non-dominated points. The second goal involves the use of diversity mechanisms. Alternatively, indicators such as the hypervolume indicator play a crucial role to obtain a good spread of the different solutions of the population along the Pareto front.

In the context of EMO, parent selection is often uniform whereas survival selection is based on dominance and the contribution of an individual to the diversity of the population. In this paper, we explore the use of different parent selection mechanisms in EMO. The goal is to speed up the optimisation process of an EMO algorithm by selecting individuals that have a high chance of producing beneficial offspring. To our knowledge the use of different parent selection schemes has not been widely studied and there are only a few algorithms placing emphasis on selecting good parents for reproduction.

NSGA-II (Deb et al., 2002) and SPEA2 (Bleuler et al., 2001) focus on survival selection. However, both use tournament selection based on Pareto ranking and their incorporated diversity measure to select the parents. We establish a similar ranking of the individuals in the parent population and examine a wide range of parent selection distributions and their impact on the performance of our studied algorithms. In Phan et al. (2011) a MOEA with parent selection using a so-called prospect indicator is used to improve SMS-EMOA. The prospect indicator evaluates the potential (or prospect) of an individual to reproduce offspring that dominates itself. Their experimental results show improvement over classical MOEAs.

The parent selection mechanisms studied in this paper use the diversity contribution of an individual in the parent population to select promising individuals for reproduction. The main assumption is that individuals with a high diversity score are located in poorly explored or less dense areas of the search space, so the chances of creating new non-dominated individuals are better than in areas where there are several individuals. In this sense we have designed parent selection schemes for MOEAs that let the MOEA focus on individuals where the neighbourhood is not fully covered and in consequence, force the reproduction in those areas and to the spread of the population along the search space.

In our investigations, we focus on parent selection mechanisms that favour individuals having a high hypervolume contribution (HVC) or high crowding distance contribution (CDC). HVC plays a crucial role in the survival selection of hypervolume-based EMO

algorithms whereas the crowding distance measure is used in popular algorithms such as NSGA-II. We propose several different parent selection mechanisms that take one of these two measures and then select individuals according to their diversity contribution. The different selection mechanisms differ in their selection strength, from mild preferences for more appealing parents to more aggressive schemes that yield a quite drastic change of behaviour. Specifically, we propose schemes based on the ranks of the individuals according to their diversity contribution, selecting according to an exponential, power law, or harmonic distribution. Furthermore, we consider tournament selection, selecting the individuals with the highest diversity contribution (HDC) as well as a ranking scheme called Non-Minimum Uniform at Random (NMUAR) which ignores the individuals with the minimum diversity contribution.

We show by means of rigorous runtime analysis that the use of diversity-based parent selection mechanisms can significantly improve the performance of MOEAs. The area of runtime analysis has contributed significantly to the theoretical understanding of EMO algorithms (Friedrich et al., 2011; Giel and Lehre, 2010; Horoba and Neumann, 2010; Qian et al., 2016) and allows to study different components of EMO methods from a rigorous perspective. In order to gain insights into the potential benefits of the diversity-based parent selection mechanisms, we study the functions ONEMINMAX and LOTZ (Definition 2.16 and 2.17, respectively). ONEMINMAX generalizes the well-known ONEMAX function and LOTZ generalizes the well-known LEADINGONES problem to the multi-objective case. Both functions have been examined in a wide range of theoretical studies for variants of the SEMO algorithm. Other studies in the area of runtime analysis of MOEAs consider hypervolume-based algorithms (Doerr et al., 2016; Nguyen et al., 2015), namely a variant of IBEA, and MOEAs incorporating other diversity mechanisms for survival selection (see Section 3.5.1).

We show that the use of various diversity-based parent selection mechanisms speeds up SEMO by factors of order n or $n/\log n$ for ONEMINMAX and LOTZ with regards to the expected time for finding the whole Pareto front. For LOTZ the use of rank-based parent selection can reduce the expected time to compute the whole Pareto front from $\Theta(n^3)$ to $O(n^2)$ (see Definition 2.2 for the asymptotic notation). Studying ONEMINMAX, we show a similar effect, i. e., that the expected time reduces from $\Theta(n^2 \log n)$ to $O(n \log n)$ for our best performing rank-based parent selection methods. The results for ONEMINMAX also hold for GSEMO which uses standard bit mutations where every bit in the mutation step is flipped with probability $1/n$.

We provide an analysis for SEMO and a modified version of GSEMO on LOTZ as the analysis of GSEMO was too challenging. This modified GSEMO uses a feature we call

L-dominant attribute, which ensures that individuals closest to the front are selected in the parent selection step. Furthermore, we provide additional experimental results. This includes a detailed experimental investigation on the stagnation probabilities for parent selection methods that are in some cases not able to obtain the whole Pareto front. These experimental results motivate new additional theoretical analyses of the stagnation probability for very greedy schemes for GSEMO with the L-dominant attribute on LOTZ as well as SEMO and GSEMO on ONEMINMAX provided in Section 6.7.

We point out situations for LOTZ where using parent selection to focus on the highest diversity contribution can lead to stagnation if global mutations are being used. However, the same parent selection mechanism is effective for SEMO where only local mutations are being used. Investigating ONEMINMAX and NMUAR in the parent selection step, we show that the choice of the reference point for hypervolume-based selection can make the difference between stagnation and an expected polynomial time. Namely, we show that choosing the reference point as $(-n-1,-1)$ for NMUAR has a positive probability of reaching stagnation whereas any symmetric reference point $(-r,-r)$, $r \geq 1$, leads to an expected time of $\mathrm{O}(n^2)$.

Finally, we discuss our findings and conclude that the use of a power-law distribution within the parent selection provides the best trade-off between speed and the risk of stagnation.

The outline of the chapter is as follows. In Section 6.1 establishes the algorithmic framework used in the theoretical and experimental analysis. Section 6.3 establishes some general properties that enable speed-ups through diversity-based parent selection. Our rigorous runtime results for ONEMINMAX and LOTZ are presented in Section 6.4 and 6.5, respectively. An experimental study complementing the theoretical results is presented in Section 6.6 and additional experimentally motivated theoretical studies on the effectiveness of greediness in parent selection are presented in Section 6.7. Finally, we finish with some discussion and concluding remarks.

6.1 Preliminaries

We consider ONEMINMAX and LOTZ (see Definition 2.16 and 2.17) which are benchmark functions that facilitate the theoretical analysis. These functions have previously been used in the theoretical analysis of evolutionary algorithms and our choice therefore allows for comparisons with previous approaches such as the ones investigated in Giel (2003); Giel and Lehre (2010); Laumanns et al. (2004).

We focus our analysis on two simple MOEAs, SEMO and its variant GSEMO because of their simplicity and suitability for a rigorous theoretical analysis. SEMO starts with an initial

solution $s \in \{0,1\}^n$ chosen uniformly at random. All non-dominated solutions are stored in the population P. Then, it selects a solution s uniformly at random from P, and a new search point s' is produced by the mutation step which flips one bit of s chosen uniformly at random. The new population contains for each non-dominated fitness vector $f(s)$, $s \in P \cup \{s'\}$, one corresponding search point (dominated individuals are removed from the population), and in the case where $f(s')$ is not dominated, s' is added to P (see Algorithm 6).

For SEMO, we know that the expected runtime on ONEMINMAX is at most $O(n^2 \log n)$ (Giel and Lehre, 2010). We prove that this upper bound is asymptotically tight.

Theorem 6.1. *The expected time for SEMO to cover the whole Pareto front on* ONEMINMAX *is* $\Theta(n^2 \log n)$.

Proof. The upper bound was shown in Giel and Lehre (2010). For the lower bound, let $|x|_1$ denote the number of 1-bits and $|x|_0$ denote the number of 0-bits in x. Define $X_t := \min_{x \in P_t}\{|x|_1\}$ if for the initial search point x_0 we have $|x_0|_1 \geq n/2$, and $X_t := \min_{x \in P_t}\{|x|_0\}$ otherwise. Note that, by definition, $X_0 \geq n/2$. Now, $X_t = 0$ is a necessary requirement for covering the whole Pareto front at time t. Hence we lower-bound the sought time by the expected time for X_t to reach value 0.

Since only local mutations are used, X_t can only decrease by 1. In order to decrease X_t we have to select a parent with Hamming distance X_t to 0^n or 1^n, respectively, which happens with probability $1/|P_t|$. Note that $|P_t| \geq n/2 - X_t$ as the population contains individuals with $X_t, X_t + 1, \ldots, \lceil n/2 \rceil$ ones. Moreover, mutation needs to flip one of the X_t bits differing to 0^n or 1^n, respectively. Hence

$$\text{Prob}(X_{t+1} = X_t - 1 \mid X_t) \leq \frac{1}{n/2 - X_t} \cdot \frac{X_t}{n}.$$

The total expected time to decrease X_t to 0 is thus at least

$$\sum_{j=1}^{n/2} \left(\frac{n}{2} - j\right) \frac{n}{j} = \sum_{j=1}^{n/2} \frac{n^2}{2j} - \sum_{j=1}^{n/2} n \geq \frac{n^2 \ln n}{2} - O(n^2)$$

as $\sum_{j=1}^{n/2} 1/j \geq \ln n/2 = \ln n - \ln 2$. □

The reason for the relatively high runtime is that the growing population slows down exploration. The population can only expand on the Pareto front in case search points with the current highest or lowest number of ones are chosen (corresponding to a minimum X_t-value in the proof of Theorem 6.1). Once the population has grown to a size of $\mu = \Theta(n)$, the probability that this happens has decreased to $\Theta(1/n)$. This means that only a $\sim 1/n$-th

fraction of the time the algorithm has a chance to expand on the Pareto front! Uniform parent selection means that most steps are spent idling. The same effect occurs for SEMO on LOTZ as proved in Laumanns et al. (2004).

Theorem 6.2 (Lemma 2 in Laumanns et al., 2004)**.** *The expected time for SEMO to cover the whole Pareto front on* LOTZ *is* $\Theta(n^3)$.

In the case of GSEMO, a new solution s' is created by flipping each bit from a solution s independently with probability $1/n$, then it proceeds in the same way as SEMO (see Algorithm 7). For GSEMO we have upper bounds of the same order, $O(n^2 \log n)$ for ONEMINMAX (Giel and Lehre, 2010) and $O(n^3)$ for LOTZ (Giel, 2003), though no lower matching bound is available in the literature for the case of GSEMO on LOTZ.

We remark that LOTZ can also be optimised more efficiently, in time $O(n^2)$, by a tailored algorithm that uses local search along individual objectives during initialisation to locate both extreme points of the Pareto front, 0^n and 1^n, and then uses crossover to produce the whole Pareto front from these points (Qian et al., 2013). Incorporating a fairness mechanism which makes sure that each individual produces roughly the same number of offspring into SEMO leads to the algorithm FEMO. For FEMO a runtime bound of $\Theta(n^2 \log n)$ has been given in Laumanns et al. (2004). The runtime analysis provided for IBEA in Nguyen et al. (2015) gives an upper bound of $O(n^2 \log n)$ and $O(n^3)$ for ONEMINMAX and LOTZ, respectively, if the population size is set to $n+1$ and therefore does not improve on the results for SEMO given in Laumanns et al. (2004).

Our aim is to develop rigorous runtime bounds of SEMO and GSEMO introducing different diversity-based parent selection. We want to study how these mechanisms help to improve the performance of the MOEAs.

6.2 Diversity-Based Parent Selection

Hypervolume-based EAs have become very popular in recent years for multi-objective optimisation where the hypervolume indicator is used as a measurement of the coverage of the population (Auger et al., 2012; Zitzler and Künzli, 2004). The hypervolume indicator measures a set of elements corresponding to images of the individuals with the volume of the dominated portion of the objective space. It is calculated based on the selection of a reference point. In particular, given a reference point $r \in \mathbb{R}^m$, the hypervolume indicator is defined on a set $P \subset S$ as

$$I_H(P) = \lambda \left(\bigcup_{x \in P} [f_1(x), r_1] \times [f_2(x), r_2] \times \cdots \times [f_m(x), r_m] \right)$$

where $\lambda(S)$ denotes the Lebesgue measure of a set S and $[f_1(a), r_1] \times [f_2(a), r_2] \times \cdots \times [f_m(a), r_m]$ is the orthotope with $f(a)$ and r in opposite corners. We define the contribution of an element $x \in P$ to the hypervolume of a set of elements P as

$$\mathrm{HVC}(x, P) = \mathrm{I_H}(P) - \mathrm{I_H}(P \setminus \{x\}).$$

The calculation of hypervolume indicator and the calculation of the contribution are both NP-hard when the number of objectives m is a parameter (Bringmann and Friedrich, 2010, 2012). However, both can be computed in polynomial time if m is fixed. In the following, for bi-objective problems like ONEMINMAX and LOTZ, we can directly calculate the contribution of an element by taking into account the two direct neighbours in the objective space as follows.

Definition 6.3 (Hypervolume contribution)**.** *For a given reference point $r = (r_1, r_2)$, we set $f_1(x_0) = r_1$ and $f_2(x_{\mu+1}) = r_2$ where x_0 and $x_{\mu+1}$ are individuals used to estimate the hypervolume contribution, and hereinafter μ denotes the size of the current population in SEMO/GSEMO. Furthermore, we assume that $r_1 = f_1(x_0) < f_1(x_1)$, $r_2 = f_2(x_{\mu+1}) < f_2(x_\mu)$. Let the population be sorted according to the value of $f_1(x_i)$ such that*

$$f_1(x_0) < f_1(x_1) < f_1(x_2) < \cdots < f_1(x_\mu).$$

The contribution of an individual x_i to the hypervolume of a population P is then given by

$$\mathrm{HVC}(x_i, P) = (f_1(x_i) - f_1(x_{i-1})) \cdot (f_2(x_i) - f_2(x_{i+1})).$$

Another diversity metric applied to our framework is the *crowding distance* used in NSGA-II (Deb et al., 2002). The crowding distance operator measures the density of solutions surrounding a particular solution in the population. A solution with a lower crowding distance value implies that the region occupied by this solution is crowded by other solutions. The solutions with a higher crowding distance value are chosen/preferred for reproduction.

Since both SEMO and GSEMO use a population of non-dominated individuals, i. e., all individual have the minimum non-domination rank possible, we can directly apply the crowding distance as our diversity metric (Algorithm 28). First, each i individual in the population P is able to store the distance from its neighbours denoted by $P[i]_{.\mathrm{distance}}$. The population is sorted for each m-th objective function in ascending order. Thereafter, for each objective function value, the boundary solutions (solutions with smallest and largest function values) are assigned an infinite distance value ($P[1]_{.\mathrm{distance}} := P[l]_{.\mathrm{distance}} := \infty$). All

other intermediate solutions are assigned a distance value equal to the absolute normalised difference of the function values of two adjacent solutions (see Line 9 of Algorithm 28, here $P[i+1]_{.m}$ and $P[i-1]_{.m}$ denote the function value of individuals $i+1$ and $i-1$ from population P respectively and $f_m^{\max}$ and $f_m^{\min}$ are the maximum and minimum values of the m-th objective function).

Algorithm 28 Crowding Distance Operator

Require: Set of search points P.
Ensure: Crowding distance contribution for each individual in P.
1: Let $l := |P|$.
2: **for all** i individuals $\in P$ **do**
3: Set $P[i]_{.\text{distance}} := 0$
4: **end for**
5: **for all** m objectives **do**
6: Sort P according to m objective function value in ascending order.
7: $P[1]_{.\text{distance}} := P[l]_{.\text{distance}} := \infty$.
8: **for** $i = 2$ **to** $l-1$ **do**
9: $P[i]_{.\text{distance}} := P[i]_{.\text{distance}} + \frac{P[i+1]_{.m} - P[i-1]_{.m}}{f_m^{\max} - f_m^{\min}}$
10: **end for**
11: **end for**
12: **return** P.

As in previous theoretical studies, we measure the runtime as the number of function evaluations needed to fully cover the Pareto front. This common practice is motivated by the fact that function evaluations are often the most time-consuming operations. Note that for SEMO and GSEMO the number of function evaluations coincides with the number of generations needed as each generation only creates one new offspring whose fitness is evaluated.

For the hypervolume contribution (HVC), according to Definition 6.3, the reference point can be defined so that the current extreme individuals in the population and individuals in intermediate empty areas have a high diversity score, and a strong influence for the algorithm. In the case of the crowding distance contribution (CDC) the same behaviour applies, extreme points in the search space receive a high distance while intermediate individuals surrounded by empty areas receive a higher distance than the ones where the area is more crowded.

With this information we can define selection mechanisms capable of selecting those extreme points and pushing the spread of the population toward the outer areas of the search space. However, as our theoretical analysis will show, in case the population already contains the extreme points of the Pareto front (0^n and 1^n for ONEMINMAX and LOTZ), we need to

be flexible enough to ignore those points and select intermediate individuals surrounded by empty areas in the search space to fully cover the Pareto front.

The selection mechanisms defined in this paper use the previous diversity contribution metrics but any other metric can be easily applied that follows the behaviour mentioned before. Firstly, we define 3 different rank-based selection schemes in which the probability of selecting individuals with a high diversity score is higher than for individuals with a lower diversity score (see Definition 6.4). The first is called *exponential*; it is a rather aggressive scheme that strongly favours the best-ranked individuals and has a very small tail. The second is called *power law* as it follows a power law distribution; it is much less aggressive with a fat tail and yet a constant probability of selecting the first constant ranks. And finally, the third ranking scheme is called *harmonic*; it is the least aggressive scheme with a fat tail and only a probability of $O(1/(\log \mu))$ for selecting the best few individuals.

Definition 6.4 (Rank-based selection schemes)**.** *The probability of selecting the i-th ranked individual is*

$$\frac{2^{-i}}{\sum_{j=1}^{\mu} 2^{-j}}, \quad \frac{1/i^2}{\sum_{j=1}^{\mu} \frac{1}{j^2}}, \quad \frac{1/i}{\sum_{j=1}^{\mu} \frac{1}{j}}$$

for the exponential, power law, and harmonic ranking scheme (see Figure 6.1), respectively.

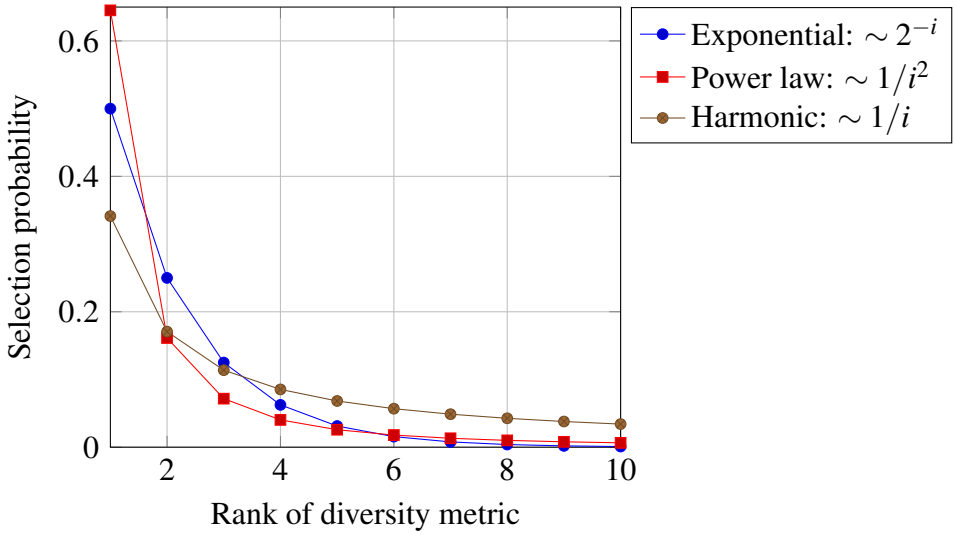


Figure 6.1: Rank-based selection schemes and its selection probabilities.

Secondly, we use the classical tournament selection, but with a specific tournament size of μ, the current size of the population. This means we choose μ individuals uniformly at

random with replacement from the population and then select the individual with the highest diversity contribution from this multi-set. Selection with replacement implies that there is a chance of not selecting particular individuals, while other individuals might be picked multiple times.

Now we introduce the diversity-based parent selection into SEMO (see Algorithm 29) and GSEMO (see Algorithm 30). Instead of using uniform parent selection, we estimate the diversity contribution for all the individuals in the population, and a parent is selected according to the diversity-based parent selection method. Then we continue as in the original algorithms. Our parent selection mechanisms are not limited to these algorithms and may prove useful on a much broader class of MOEAs.

Algorithm 29 SEMO with diversity-based parent selection

Choose an initial solution $s \in \{0,1\}^n$ uniformly at random.
Determine $f(s)$ and initialize $P := \{s\}$.
while stopping criterion **not** met **do**
    Estimate diversity contribution $\forall s \in P$.
    Choose $s \in P$ according to the parent selection mechanism.
    Choose $i \in \{1,\ldots,n\}$ uniformly at random.
    Define s' by flipping the i-th bit of s.
    if s' is **not** dominated by any individual in P **then**
        Add s' to P, and remove all individuals weakly dominated by s' from P.
    end if
end while

Algorithm 30 GSEMO with diversity-based parent selection

Choose an initial solution $s \in \{0,1\}^n$ uniformly at random.
Determine $f(s)$ and initialize $P := \{s\}$.
while stopping criterion **not** met **do**
    Estimate diversity contribution $\forall s \in P$.
    Choose $s \in P$ according to the parent selection mechanism.
    Create s' by flipping each bit in s independently with probability $1/n$.
    if s' is **not** dominated by any individual in P **then**
        Add s' to P, and remove all individuals weakly dominated by s' from P.
    end if
end while

6.3 On Diversity-Based Progress

We show that diversity-based parent selection mechanisms can achieve a fast spread on the Pareto front. The following arguments and analyses consider the situation where the population is located on the Pareto front. This is trivially the case for ONEMINMAX as all search points are Pareto-optimal. For LOTZ we later supply a separate analysis that covers the process of reaching the Pareto front.

For ONEMINMAX and LOTZ the most promising parents are those that have a Hamming neighbour that is on the Pareto set, but not yet contained in the population. We call these search points *good*:

Definition 6.5 (good individuals)**.** *With reference to a population P and a fitness function with Pareto front F^* and corresponding Pareto set X^*, we call a search point $x \in P \cap X^*$* good *if there is a Hamming neighbour y of x such that $y \in X^*$ but $f(y) \notin f(P)$ where $f(P)$ denotes the set of objective vectors of population P. Otherwise, x is called* bad*.*

A diversity measure should encourage the selection of such good individuals.

Definition 6.6 (diversity-favouring)**.** *We call a measure $\mathrm{C}(x,P)$* diversity-favouring *on $S \subseteq \{0,1\}^n$ with respect to a fitness function with Pareto front F^* if for all populations P and all $x,y \in P \cap X^* \cap S$ we have the following: if x is bad and y is good then $\mathrm{C}(x,P) < \mathrm{C}(y,P)$.*

Note that the definition is restricted to a subset S of the search space. The reason is to allow the exclusion of certain search points for which the property is not true. For ONEMINMAX and LOTZ, the property does not hold for the extreme points on the Pareto front, 0^n and 1^n. We show that both HVC and CDC are both diversity-favouring on all other search points. For HVC we assume that the reference point is dominated by $(-1,-1)$. In other words, the reference point can be any point (r_1,r_2) with $r_1 \leq -1$ and $r_2 \leq -1$.

Lemma 6.7. *The hypervolume contribution $\mathrm{HVC}(x,P)$ is diversity-favouring on $\{0,1\}^n \setminus \{0^n,1^n\}$ for both* ONEMINMAX *and* LOTZ *if the reference point is dominated by $(-1,-1)$.*

Proof. Let us consider an individual $x_i \notin \{0^n,1^n\}$ of the sorted population according to f_1, using the notation from Definition 6.3. If x_i is bad, then there are Hamming neighbours x_{i-1} and x_{i+1} of x_i in P, the $\mathrm{HVC}(x_i,P)$ is the minimum possible, since $f_1(x_i) - f_1(x_{i-1}) = 1$ and $f_2(x_i) - f_2(x_{i+1}) = 1$ yielding $\mathrm{HVC}(x_i,P) = (f_1(x_i) - f_1(x_{i-1})) \cdot (f_2(x_i) - f_2(x_{i+1})) = 1$.

Now, let us consider a good search point y_i, that is, y_{i-1} or y_{i+1} is not a Hamming neighbour of y_i. Then we have $f_1(y_i) - f_1(y_{i-1}) > 1$ or $f_2(y_i) - f_2(y_{i+1}) > 1$ and in any case $\mathrm{HVC}(y_i,P) = (f_1(y_i) - f_1(y_{i-1})) \cdot (f_2(y_i) - f_2(y_{i+1})) > 1$. Thus $\mathrm{HVC}(y_i,P) > \mathrm{HVC}(x_i,P)$, which completes the proof. □

Lemma 6.8. *The crowding distance contribution* $\mathrm{CDC}(x,P)$ *is diversity-favouring on* $\{0,1\}^n \setminus \{0^n,1^n\}$ *for both* ONEMINMAX *and* LOTZ.

Proof. By Algorithm 28 the search points with the minimum and maximum f_1 score in the population are going to have infinite diversity score, regardless of the objective chosen to sort the population.

Let us say that there is a bad individual x_i with Hamming neighbours x_{i-1} and x_{i+1} contained in P. According to the numerator of Line 9 of Algorithm 28, the difference between the $f_1(x_{i-1})$ (or $f_2(x_{i-1})$) and $f_1(x_{i+1})$ is the minimum possible, which means the minimum $\mathrm{CDC}(x_i,P)$ is assigned to the individual x_i.

In the case of a good search point y_i, that is, y_{i-1} or y_{i+1} are not Hamming neighbours of y_i, the difference between the next contained search points in P is higher. If the difference between $f_1(y_i)$ (or $f_2(y_i)$) is higher than the minimum possible, this means $\mathrm{CDC}(x_i,P) < \mathrm{CDC}(y_i,P)$ which completes the proof. □

Note that in both above measures 0^n and 1^n, if contained in the population, will always receive a high score, regardless of whether they are good or bad. If they are bad, there is a high chance that a bad individual will be selected as parent in a diversity-based parent selection mechanism. With this in mind, the probability of selecting a good individual can be bounded from below as follows.

Lemma 6.9. *Let* $\mathrm{C}(x,P)$ *be a diversity-favouring measure on* $\{0,1\}^n \setminus \{0^n,1^n\}$. *Consider either* ONEMINMAX *or* LOTZ *and assume the population P is a subset of the Pareto set,* $P \subseteq X^*$. *Imagine P being sorted according to non-increasing* $\mathrm{C}(x,P)$ *values. Consider a parent selection mechanism based on* $\mathrm{C}(x,P)$ *such that* r_i *is the probability of selecting the i-th element of P in the sorted sequence. Then the probability of selecting a good individual is at least* $\min\{r_1,r_2,r_3\}$ *unless P already covers the Pareto front.*

Proof. Before the whole Pareto front is covered by the population P, there exists at least one good individual x in population P with no corresponding Hamming neighbour s in the Pareto set X^*. Then the individuals which correspond to the Hamming neighbours of the missing point s are good search points.

Since $\mathrm{C}(x,P)$ is defined as a diversity-favouring measure on $\{0,1\}^n \setminus \{0^n,1^n\}$, the good search points have higher contribution than bad search points that are neither 0^n nor 1^n. Therefore, among the top three ranked elements in P, there exists at least one good individual. The probability of selecting this good individual is at least $\min\{r_1,r_2,r_3\}$. □

The parent selection mechanisms thus have the following probability of selecting good individuals.

Lemma 6.10. *In the setting described in Lemma 6.9, the probability p_{good} of selecting a good individual is*

1. $\Omega(1)$ *for the exponential and power law ranking schemes,*
2. $\Omega(1/\log\mu)$ *for the harmonic ranking scheme,*
3. $\Omega(1)$ *for tournament selection with tournament size μ.*

Proof. For the parent selection with the exponential ranking scheme, the probability follows from Lemma 6.9, which fulfils

$$r_1 \geq r_2 \geq r_3 = \frac{2^{-3}}{\sum_{j=1}^{\mu} 2^{-j}} \geq 2^{-3} = \Omega(1).$$

For the power law ranking scheme, since $\sum_{j=1}^{\mu} \frac{1}{j^2} \leq \sum_{j=1}^{\infty} \frac{1}{j^2} = \pi^2/6$, the probability fulfils

$$r_1 \geq r_2 \geq r_3 = \frac{1/3^2}{\sum_{j=1}^{\mu} \frac{1}{j^2}} \geq \frac{2}{3 \cdot \pi^2} = \Omega(1).$$

In the case of the harmonic ranking scheme, since $\sum_{j=1}^{\mu} \frac{1}{j} \leq \ln\mu + 1$, the probability fulfils

$$r_1 \geq r_2 \geq r_3 = \frac{1/3}{\sum_{j=1}^{\mu} \frac{1}{j}} \geq \frac{1}{3 \cdot (\ln\mu + 1)} = \Omega(1/\log\mu).$$

For tournament selection, the probability of selecting a good individual is at least $\min\{r_1, r_2, r_3\}$ and $r_1 \geq r_2 \geq r_3$. In order for the individual with the 3rd maximum contribution to be selected in the tournament selection, the individuals with the 1st and 2nd maximum contribution should never be selected in the μ times (probability of $(1-2/\mu)^{\mu}$). And, conditional on this happening, the individual with the 3rd maximum contribution has to be chosen at least once amongst the other $\mu - 2$ individuals in the μ times with probability $1 - \left(1 - \frac{1}{\mu-2}\right)^{\mu}$. Hence, the probability of selecting a good individual is at least

$$p_{\text{good}} \geq \left(1 - \left(1 - \frac{1}{\mu-2}\right)^{\mu}\right) \cdot \left(1 - \frac{2}{\mu}\right)^{\mu} \geq \left(1 - \frac{1}{e}\right) \cdot \left(1 - \frac{2}{\mu}\right)^{\mu}$$

using $\left(1 - \frac{1}{x}\right)^{x} \leq 1/e$ for $x > 1$. Since $f(x) = \left(1 - \frac{1}{x}\right)^{x}$ is non-decreasing when $x \geq 1$, with $\mu \geq 3$, $\left(1 - \frac{2}{\mu}\right)^{\frac{\mu}{2}} \geq \left(1 - \frac{2}{3}\right)^{\frac{3}{2}} \geq 0.19$. Therefore, $p_{\text{good}} \geq \left(1 - \frac{1}{e}\right) \cdot 0.19^2 = \Omega(1)$. □

6.4 Speedups on OneMinMax

For any parent selection mechanism defined before, the parent selection is focused on selecting an individual with a high diversity score. In the case of HVC or CDC, having a high diversity contribution means that, apart from the possible exceptions of 0^n and 1^n, the parent will be good, i. e., located in a less populated area of the Pareto front. We show that by preferring good individuals in the parent selection, SEMO and GSEMO can quickly find the whole Pareto front for OneMinMax.

Lemma 6.11. *Suppose that the probability of selecting a good individual as a parent is at least p_{good}. Then the expected runtime for SEMO or GSEMO to find all solutions in the Pareto front on* OneMinMax *is bounded above by $\mathrm{O}\big((n \log n)/p_{\text{good}}\big)$.*

Proof. We call a step a *relevant step* if the algorithm selects a good parent on the Pareto front. We show in the following that $\mathrm{O}(n \log n)$ relevant steps are sufficient for covering the whole Pareto front of OneMinMax, regardless of irrelevant steps performed. This shows the claim as the expected time for a relevant step is $1/p_{\text{good}}$.

We use the *accounting method* (see, Section 2.3.2 or Section 17.2 in Cormen et al., 2009) to bound the number of relevant steps. As mentioned in Section 2.3.2 we use this method to bound the time required to create desired offspring, i. e., count the expected number of mutations needed to create an individual from a parent selected for mating (this is our definition of relevant steps). Specifically, we count the number of relevant steps spent in selecting a good parent with i ones. Summing up (upper bounds on) all these times across all $0 \leq i \leq n$ will imply the claim.

Note that, once potential gaps at $i-1$ and $i+1$ are filled, there can be no more relevant steps at i ones, due to the definition of a relevant step. Hence the expected number of relevant steps at i ones is bounded by the expected number of mutations from i needed to fill both these gaps. If an individual with i ones, $0 < i < n$, is selected as parent, the probability of mutation creating an individual with $i-1$ ones is at least $i/n \cdot (1-1/n)^{n-1} \geq i/(en)$, and the probability of mutation creating an individual with $i+1$ ones is at least $(n-i)/n \cdot (1-1/n)^{n-1} \geq (n-i)/(en)$ (this holds both for SEMO and GSEMO; for SEMO the factor $1/e$ can be removed). The time for filling both gaps is at most $en/i + en/(n-i)$. Hence there are at most $en/i + en/(n-i)$ relevant steps selecting a parent with i ones. In the special cases of $i = 0$ or $i = n$ the time to fill the neighbouring gaps simplifies to $en/n = e$.

Summing over all i, the expected total number of relevant steps is hence at most

$$2e + \sum_{i=1}^{n-1} \left(\frac{en}{i} + \frac{en}{n-i} \right) = 2e + 2 \sum_{i=1}^{n-1} \frac{en}{i} = 2 \sum_{i=1}^{n} \frac{en}{i} \leq 2en(\log n + 1).$$

Where the summation $H_n = \sum_{i=1}^{n} 1/i$ is known as the *harmonic number* and satisfies $H_n = \ln n + \Theta(1)$ this completes the proof. □

Combining Lemma 6.10 and Lemma 6.11, we have proved the following results. Note that the population size μ is always at most $n+1$ on ONEMINMAX and LOTZ, hence for the harmonic ranking scheme, $p_{\text{good}} = \Omega(1/\log \mu) = \Omega(1/\log n)$.

Theorem 6.12. *Consider SEMO and GSEMO with diversity-based parent selection using any diversity measure that is diversity-favouring on* $\{0,1\}^n \setminus \{0^n, 1^n\}$ *(e. g.,* HVC *or* CDC*). Then the expected time to find the whole Pareto front on* ONEMINMAX *is bounded by* $O(n \log n)$ *for the exponential and power law ranking schemes, and for tournament selection with tournament size* μ*. It is bounded by* $O(n \log^2 n)$ *for the harmonic ranking scheme.*

As both SEMO and GSEMO with the classical uniform parent selection need time $\Theta(n^2 \log n)$ on ONEMINMAX, our parent selection schemes lead to speedups of order $\Theta(n)$ and $\Theta(n/\log n)$, respectively.

6.5 Speedups on LOTZ

We now turn to the function LOTZ. In contrast to ONEMINMAX, where all individuals are Pareto optimal, for LOTZ we have to estimate the time for the population to reach the Pareto front. For SEMO the approach to the Pareto front can be estimated easily since SEMO keeps only one individual in the population. For local mutations as used in SEMO, whenever an offspring is created, either the offspring dominates the parent, or the parent dominates the offspring (or both, if they have the same function values). The population size remains unchanged before there is a solution on the Pareto front. For any parent on the Pareto front, SEMO only accepts its offspring if it is also on the Pareto front, otherwise the offspring is dominated by the parent.

Lemma 6.13. *The expected time for SEMO to reach the Pareto front is* $O(n^2)$*. Assume that afterwards the probability of selecting a good individual in the population is at least* p_{good}*. The expected runtime for SEMO to reach a population covering the whole Pareto front on* LOTZ *is bounded above by* $O(n^2/p_{\text{good}})$.

Proof. The time for the population to find the first Pareto-optimal point is $O(n^2)$ and has already been proved in Lemma 1 in Laumanns et al. (2004). So we can focus on the time required to find the whole Pareto front. By the *accounting method* used to prove Lemma 6.11 and the definition of relevant step: the algorithm selects a good parent on the Pareto front, we

count the number of relevant steps spent selecting a good parent with i leading ones, 1^i0^{n-i}, and sum up all these times across all $0 \leq i \leq n$ to prove the claim.

The potential gaps consist of non-existing non-dominated individuals at $i-1$ and $i+1$ ($1^{i-1}0^{n-i+1}$ and $1^{i+1}0^{n-i-1}$, respectively). It is necessary to fill those gaps by including these search points in the population. Once this has happened, there can be no more relevant steps at i leading ones. So the expected number of mutations at i leading ones is bounded by the expected number of mutations from i needed to fill $i-1$ and $i+1$. If 1^i0^{n-i} is selected as parent, the probability of mutation creating $1^{i-1}0^{n-i+1}$ or $1^{i+1}0^{n-i-1}$ is $1/n$, respectively. The time for filling both gaps (if existent) is at most $n+n$. Hence there are in expectation at most $2n$ relevant steps selecting a parent with i leading ones.

Summing over all i, the expected total number of relevant steps is hence at most

$$\sum_{i=0}^{n} 2n = 2n(n+1) = \mathrm{O}\big(n^2\big).$$

Noting that the expected waiting time for a relevant step is $1/p_{\text{good}}$. Thus the overall expected runtime for SEMO to achieve a population covering the whole Pareto front on LOTZ is upper bounded by $\mathrm{O}\big(n^2\big) + \mathrm{O}\big(n^2/p_{\text{good}}\big) = \mathrm{O}\big(n^2/p_{\text{good}}\big)$. □

Combining Lemma 6.10 and Lemma 6.13, we now have proved the following results.

Theorem 6.14. *Consider SEMO with diversity-based parent selection using any diversity measure that is diversity-favouring on* $\{0,1\}^n \setminus \{0^n, 1^n\}$ *(e. g.,* HVC *or* CDC*). Then the expected time to find the whole Pareto front on* LOTZ *is bounded by* $\mathrm{O}\big(n^2\big)$ *for the exponential and power law ranking schemes, and for tournament selection with tournament size* μ*. It is bounded by* $\mathrm{O}\big(n^2 \log n\big)$ *for the harmonic ranking scheme.*

The analysis of GSEMO turns out to be more difficult than the analysis of SEMO. The reason is that the approach to the Pareto front becomes harder to analyse. With global mutations, GSEMO can create incomparable search points while approaching the Pareto front. This means that the population can expand in size while approaching the Pareto front, and even after the whole population has reached the Pareto front, it is possible to create search points off the Pareto front that are accepted in the population.

Experiments in Section 6.6 indicate that this behaviour does not slow down the algorithm by more than a constant factor. However, proving that the bound $O(n^2)$ for SEMO also holds for GSEMO turns out to be very challenging. We therefore take a different approach and analyse a modified variant of GSEMO that is easier to analyse. Experiments presented in Section 6.6 confirm that this modification does not significantly change the average runtime (inspecting Tables 6.3 and 6.4, the quotients of average times for the modified GSEMO

Algorithm 31 Modified Global SEMO with diversity-based parent selection

```
Choose an initial solution s ∈ {0,1}^n uniformly at random.
Determine f(s) and initialize P := {s}.
while stopping criterion not met do
    Let P' ⊆ P be the set of all search points with a maximum L-dominant attribute in P.
    Estimate diversity contribution ∀s ∈ P' w. r. t. the population P'.
    Choose s ∈ P' according to parent selection mechanism.
    Create s' by flipping each bit of s independently with probability 1/n.
    if s' is not dominated by any individual in P then
        Add s' to P, and remove all individuals weakly dominated by s' from P.
    end if
end while
```

and those for the original GSEMO across all parent selection mechanisms are 0.88 for $\mathrm{HVC}(-1,-1)$, 1.19 for $\mathrm{HVC}(-n,-n)$, and 1.27 for CDC, averaging to 1.1 are close to 1 in many settings and always in the interval $[0.48, 2.03]$).

The idea behind this modification is to simplify the approach to the Pareto front by restricting parent selection to search points that are maximal with regards to a linear combination of both objectives.

Definition 6.15 (L-dominant attribute)**.** *Let* $\mathrm{L}(x) = \mathrm{LO}(x) + \mathrm{TZ}(x)$*, where* $\mathrm{LO}(x)$ *and* $\mathrm{TZ}(x)$ *denotes the total number of leading ones and the total number of trailing zeros of a certain individual x, respectively.*

We modify GSEMO in such a way that it only picks parents with maximal L-dominant attribute in the population (see Algorithm 31), and also the computation of the diversity contribution is restricted to these search points. This has two effects: it simplifies and facilitates the analysis of the individuals while they are approaching the Pareto front. While the original GSEMO can store incomparable search points with different L-values in the population, the modified GSEMO only considers incomparable search points with maximum L-value. In addition, since all x individuals on the Pareto front have the largest possible value of $\mathrm{L}(x) = n$, once the Pareto front is reached, the algorithm only selects individuals on the Pareto front as parents according to their diversity contribution.

Lemma 6.16. *The expected time for the modified GSEMO to reach the Pareto front is bounded above by* $\mathrm{O}(n^2)$.

Proof. According to Definition 6.15, before reaching the Pareto front, the solution with $\max_{x \in P}(\mathrm{L}(x))$ is selected to generate an offspring. Consider the event of only flipping the first 0-bit or the last 1-bit of the selected individual. Since the offspring from this event

has a higher value of one of the objectives than its parent which is of the maximum $\mathrm{L}(x)$ in the population, the offspring is non-dominated by any individuals in the population and is accepted by the algorithm. Hence, the probability of increasing $\max_{x\in P}(\mathrm{L}(x))$ is at least

$$2\cdot\frac{1}{n}\cdot\left(1-\frac{1}{n}\right)^{n-1}\geq\frac{2}{en}.$$

Throughout the process, the value of $\max_{x\in P}(\mathrm{L}(x))$ in the population never goes down. Therefore, the overall expected runtime for GSEMO with this selection scheme to reach the Pareto front is at most

$$\sum_{\mathrm{L_{max}}=0}^{n-2}\frac{en}{2}=\mathrm{O}\big(n^2\big).$$ □

Lemma 6.17. *Assume that the probability of selecting a good individual in the population is at least p_{good}. The expected time for the modified GSEMO to reach a population covering the whole Pareto front on* LOTZ *is bounded above by $\mathrm{O}\big(n^2/p_{\mathrm{good}}\big)$.*

Proof. As for SEMO, before the population covers the whole Pareto front, the optimisation process of the modified GSEMO can be divided into two stages. The first stage focusses on obtaining the first individual on the Pareto front and the second one focusses on covering the Pareto front. As proved in Lemma 6.16, the expected time for the modified GSEMO to reach the Pareto front is at most $\mathrm{O}\big(n^2\big)$.

In the second stage, by following the definition of relevant step, the parent to be selected is a good search point on the Pareto front with the maximum $\mathrm{L}(x)$ dominant attribute. The algorithm will select individuals on the Pareto front with the maximum $\mathrm{L}(x)$ dominant attribute according to their diversity contribution. So now we can apply the accounting method used to prove previous lemmas to bound the number of relevant steps spent selecting the good parent.

As in Lemma 6.13, we define a good parent with i leading ones with possible gaps on $i-1$ and/or $i+1$ across all $0\leq i\leq n$. And by introducing the factor $1/e$ to the analysis in Lemma 6.13, we now have the time for filling both gaps is at most $1/(en)+1/(en)$. Hence there are at most $en+en=2en$ relevant steps selecting a good parent with i leading ones. Summing over all i, the expected total number of relevant steps is hence at most

$$\sum_{i=0}^{n}2en=2e\sum_{i=0}^{n}n=\mathrm{O}\big(n^2\big)$$

The overall runtime for the modified GSEMO on LOTZ to reach a population covering the whole Pareto front is bounded above by $\mathrm{O}\big(n^2/p_{\mathrm{good}}\big)$. □

As mentioned on the proof of the previous lemma, once the individual with the maximum $\mathrm{L}(x)$ dominant attribute has reached the Pareto front, the algorithm will always select good individuals on the Pareto front (with the maximum $\mathrm{L}(x)$ dominant attribute) according to their diversity contribution. This characteristic allows us to apply Lemma 6.10, and by Lemma 6.17, we now have proved the following results.

Theorem 6.18. *Consider the modified GSEMO with diversity-based parent selection using any diversity measure that is diversity-favouring on* $\{0,1\}^n \setminus \{0^n, 1^n\}$ *(e. g.,* HVC *or* CDC*). Then the expected time to find the whole Pareto front on* LOTZ *is bounded by* $\mathrm{O}\left(n^2\right)$ *for the exponential and power law ranking schemes, and for tournament selection with tournament size* μ*. It is bounded by* $\mathrm{O}\left(n^2 \log n\right)$ *for the harmonic ranking scheme.*

6.6 Experimental Analysis

The experimental approach is focused on the analysis of SEMO, GSEMO and the modified GSEMO and their performance with and without the diversity-based parent selection mechanisms. We are interested in observing if we can speed up the performance from the classical approaches. For the case of the modified GSEMO, we measure its performance only on LOTZ and we compare its performance to GSEMO in order to observe the impact of the L-dominant attribute on the performance of the algorithm.

Experiments also allow for a more detailed comparison of the HVC, CDC, and the parent selection methods. In the case of the HVC, we have defined two settings for the reference points, $(-1,-1)$ and $(-n,-n)$. For the first reference point, a slight preference to the extreme points is provided while with the second, the influence of the extreme points becomes very strong. This particular characteristic became an interesting feature to observe in the case of the ranking-based selection schemes, and exposes a potential flaw for the case of HVC with low (or high in the case of minimisation) reference point or CDC (since it assigns infinite value to the extreme points) and the parent selection mechanisms that focus very aggressively toward the extreme points, as we shall see below.

Since we are interested in the time required to find the Pareto front, we report the following outcomes and stopping criteria for each run. *Success*, the whole Pareto front has been covered, i. e., the run is stopped if the population contains all individuals on the Pareto front. *Failure/Stagnation*, once the run has reached 1 million generations and the Pareto front has not been fully covered, this is enough time for the algorithms to create new individuals and fill the gaps on the Pareto front. We repeat the experimental framework for 100 runs with problem size $n = 100$ for all algorithmic approaches and report the mean and standard deviation (STD) as our metrics of interest.

Table 6.1: Mean (first rows) and STD (second rows) of generations required to find the Pareto front for SEMO and GSEMO on ONEMINMAX and LOTZ with $n = 100$.

Algorithms	ONEMINMAX	LOTZ
SEMO	4.16E+04	3.17E+05
	1.15E+04	5.34E+04
GSEMO	1.06E+05	6.58E+05
	3.47E+04	1.12E+05

Table 6.1 shows the mean and STD of generations required to find the Pareto front for the classic SEMO and GSEMO that use uniform parent selection for both test functions. Table 6.2 and 6.3 refer to the mean and standard deviation of generations required to find the Pareto front for SEMO and GSEMO with the different diversity-based parent selection schemes for ONEMINMAX and LOTZ, respectively. Finally, Table 6.4 shows the mean and standard deviation of generations required to find the Pareto front for the modified GSEMO on LOTZ.

As we mentioned before, a parent selection mechanisms that is extremely focused on the extreme points can be potentially dangerous, and to exemplify this, we have introduced a deterministic selection mechanism which we have named *Highest Diversity Contribution* (HDC): always select an individual with the highest diversity contribution (break ties uniformly at random if there are several such points). We also have defined a modified version of the uniform random selection used by SEMO and GSEMO, that we call *Non-Minimum Uniform at Random* (NMUAR), where the individuals with the minimum diversity contribution in the population are ignored (provided that the population does contain multiple diversity contribution values) and one individual is selected uniformly at random from all remaining individuals. In this sense individuals with high diversity contributions have better probabilities to be selected and the approach is flexible enough to choose between extreme and intermediate individuals.

As it can be observed in Table 6.2 and 6.3, HDC fails to find the Pareto front for ONEMINMAX and LOTZ in the case of GSEMO for both diversity-based metrics. For the case of GSEMO with HDC selection mechanism with HVC and CDC on ONEMINMAX the failure rate was 0.94 and 0.93, respectively. On LOTZ, the failure rate was 1.0 for both diversity metrics.

The reason for these bad results for GSEMO (and the modified GSEMO) on ONEMINMAX is due to the mutation operator. Both algorithms can create gaps by creating an offspring that may differ from its parent with more than one bit. In the case of GSEMO on LOTZ, the algorithm can create incomparable search points and the population expands

Table 6.2: Mean (first rows) and STD (second rows) of generations required to find the Pareto front for SEMO and GSEMO with diversity-based parent selection methods on ONEMINMAX with $n = 100$. "Stagnation" indicates a failure rate larger than 0.

Algorithms	**HVC**$(-\mathbf{1}, -\mathbf{1})$	**HVC**$(-\mathbf{n}, -\mathbf{n})$	**CDC**
SEMO & HDC	9.14E+02 1.76E+02	8.90E+02 1.65E+02	1.05E+03 2.40E+02
GSEMO & HDC	2.12E+03 4.28E+02	Stagnation Stagnation	Stagnation Stagnation
SEMO & NMUAR	8.92E+02 1.81E+02	1.05E+03 2.72E+02	1.03E+03 2.59E+02
GSEMO & NMUAR	2.14E+03 4.97E+02	2.54E+03 6.57E+02	2.58E+03 7.86E+02
SEMO & exponential	1.28E+03 2.72E+02	1.27E+03 2.71E+02	1.36E+03 3.44E+02
GSEMO & exponential	3.21E+03 9.35E+02	3.18E+03 9.12E+02	3.24E+03 7.72E+02
SEMO & harmonic	3.05E+03 6.97E+02	3.24E+03 8.63E+02	3.28E+03 8.03E+02
GSEMO & harmonic	7.89E+03 1.90E+03	7.26E+03 1.69E+03	8.03E+03 2.09E+03
SEMO & power law	1.15E+03 2.48E+02	1.24E+03 2.89E+02	1.34E+03 3.00E+02
GSEMO & power law	2.87E+03 6.35E+02	2.85E+03 6.22E+02	3.32E+03 1.07E+03
SEMO & tournament(μ)	1.05E+03 2.24E+02	1.08E+03 2.18E+02	1.21E+03 3.09E+02
GSEMO & tournament(μ)	2.58E+03 5.48E+02	2.60E+03 7.91E+02	2.81E+03 7.34E+02

Table 6.3: Mean (first rows) and STD (second rows) of generations required to find the Pareto front for SEMO and GSEMO with diversity-based parent selection methods on LOTZ with $n = 100$. "Stagnation" indicates a failure rate larger than 0.

Algorithms	$\mathbf{HVC}(-\mathbf{1},-\mathbf{1})$	$\mathbf{HVC}(-\mathbf{n},-\mathbf{n})$	**CDC**
SEMO & HDC	1.24E+04 9.79E+02	1.25E+04 1.22E+03	1.41E+04 1.80E+03
GSEMO & HDC	3.06E+04 2.62E+03	Stagnation Stagnation	Stagnation Stagnation
SEMO & NMUAR	1.25E+04 1.10E+03	1.38E+04 1.49E+03	1.41E+04 1.52E+03
GSEMO & NMUAR	3.17E+04 3.13E+03	3.50E+04 3.85E+03	3.58E+04 3.75E+03
SEMO & exponential	1.57E+04 1.31E+03	1.58E+04 1.33E+03	1.78E+04 2.47E+03
GSEMO & exponential	3.45E+04 2.87E+03	4.00E+04 8.60E+03	5.87E+04 1.63E+04
SEMO & harmonic	3.14E+04 3.60E+03	3.08E+04 3.24E+03	3.53E+04 5.68E+03
GSEMO & harmonic	6.69E+04 7.23E+03	6.33E+04 7.40E+03	6.73E+04 1.02E+04
SEMO & power law	1.54E+04 1.26E+03	1.51E+04 1.36E+03	1.69E+04 2.13E+03
GSEMO & power law	3.40E+04 3.30E+03	5.03E+04 1.24E+04	5.73E+04 1.43E+04
SEMO & tournament(μ)	1.38E+04 1.25E+03	1.41E+04 1.12E+03	1.55E+04 1.94E+03
GSEMO & tournament(μ)	3.16E+04 2.88E+03	6.53E+04 2.15E+04	7.87E+04 2.57E+04

Table 6.4: Mean (first rows) and STD (second rows) of generations required to find the Pareto front for the modified GSEMO and diversity-based parent selection methods on LOTZ with $n = 100$. "Stagnation" indicates a failure rate larger than 0.

Algorithms	**HVC**$(-\mathbf{1},-\mathbf{1})$	**HVC**$(-\mathbf{n},-\mathbf{n})$	**CDC**
HDC	3.06E+04 2.78E+03	Stagnation Stagnation	Stagnation Stagnation
NMUAR	3.19E+04 2.92E+03	3.60E+04 4.50E+03	3.55E+04 4.92E+03
Exponential	3.95E+04 3.65E+03	3.99E+04 3.62E+03	4.55E+04 6.00E+03
Harmonic	8.13E+04 8.37E+03	8.11E+04 8.53E+03	9.49E+04 1.50E+04
Power law	3.81E+04 3.62E+03	3.81E+04 3.66E+03	4.32E+04 5.75E+03
Tournament(μ)	3.46E+04 2.95E+03	3.49E+04 3.42E+03	3.88E+04 5.50E+03

in size while approaching the Pareto front. This implies that the Pareto front is reached in different areas at different times during the run, leaving intermediate unexplored regions. Once the Pareto front has been reached, the algorithm can create gaps by creating an offspring by flipping more than one leading one or trailing zero. Then it will continue selecting those individuals ignoring the intermediate ones, leaving the population in a *stagnation* state. This observation also justifies why we introduced parent selection schemes of varying degree of aggressiveness. We analyse this process rigorously in Section 6.7.1.

For all other parent selection schemes defined in this paper, we have achieved a significant speed up in the performance of SEMO and GSEMO of around one order of magnitude. As it can be observed in Table 6.2 and 6.3, SEMO and GSEMO with diversity-based parent selection mechanisms are able to find the Pareto front faster than its classical counterparts, i. e., fewer generations are required for both test functions. Note that the problem size $n = 100$ is relatively moderate; as our theoretical results prove, speedups over the original algorithms will grow further when the problem size is increased.

In the case of the modified GSEMO, the same stagnation state was reached (see Table 6.4). For the modified GSEMO with HDC selection mechanism with HVC and CDC on ONEMINMAX the failure rate was 0.97 and 1.0, respectively. On LOTZ the failure rate for the modified GSEMO with HVC and CDC decreases considerably, reaching 0.37 and 0.33, respectively.

The modified GSEMO on LOTZ achieved a considerably lower failure rate compared to the original GSEMO, where it was 1.0. We believe that there are two reasons for this. Firstly, for the modified GSEMO it is not possible to reach the Pareto front in different areas, avoiding the creation of gaps while approaching the Pareto front; the individual with the largest L-dominant attribute will always reach the Pareto front. Secondly, after the Pareto front has been reached, the algorithm will select individuals on the Pareto front as parents according to their diversity contribution. Here, from the i individual, the mutation operator needs to flip $1^{i-1}0^{n-i+1}$ or $1^{i+1}0^{n-i-1}$ to create a new individual. In this sense it is more difficult to leave an empty space between points leading to this better performance but it is always possible for the algorithm to flip multiple consecutive bits to create a gap, resulting in the mentioned failure rates.

The modified GSEMO can also achieve a significant speed up in performance on LOTZ. With this preliminary analysis we can see that the introduction of the L-dominant attribute does not drastically change the average runtime and it can be used as an approximation or first step towards the definition of a bound for GSEMO with diversity-based parent selection on LOTZ.

6.7 Comparing Selection Schemes: How Much Greed is Good?

In this section we focus our attention on the Highest Diversity Contribution (HDC) and the Non-Minimum Uniformly at Random (NMUAR) methods. In Section 6.7.1 we discuss in detail how HDC seems to be the fastest selection mechanism for SEMO, but the worst for GSEMO as it leads to stagnation. We show by means of rigorous runtime analysis how this is a rare and natural example where multi-bit flips do a lot of harm by leading the population into a stagnation state.

Finally in Section 6.7.2 we discuss the results obtained regarding the NMUAR mechanism. As shown in Section 6.6, NMUAR performs experimentally well for SEMO and GSEMO with no stagnation outcome. We show that for a particular choice of the reference point NMUAR can lead the population into a stagnation state. On the positive side, we show that NMUAR is able to efficiently optimise both LOTZ and ONEMINMAX for common choices of the reference point.

6.7.1 Why Highest Diversity Contribution Stagnates

In this section we theoretically examine the stagnation results of Section 6.6 related to GSEMO with the HDC selection strongly favouring the extreme points. As it can be observed from Tables 6.2 and 6.3, a greedy approach seems to be the best for SEMO. SEMO can find all individuals on the Pareto front but also is the fastest in doing so. This is because for SEMO on ONEMINMAX, all individuals are part of the Pareto front and the algorithm starts with one individual on the Pareto front. In the case of LOTZ the algorithm always reaches the Pareto front with just one individual. Once on the Pareto front, the spread of the population to outer areas can only be achieved by individuals that differ from its parent in just one bit, i. e., no gaps or empty spaces are left between points.

In the following we show by means of rigorous runtime analysis why the previous experimental results occur for the modified GSEMO on LOTZ. Let the reference point be dominated by $(-n^2,-n^2)$ for the HVC in order to simplify the analysis for proving that focusing on extreme points can lead to undesired results. Our main result of this section is the following.

Theorem 6.19. *Consider the modified GSEMO with Highest Diversity Contribution, choosing as diversity metric either* CDC *or* HVC *with a reference point dominated by* $(-n^2,-n^2)$ *on the function* LOTZ. *Then at the first point in time the population* P_t *contains both* 0^n *and* 1^n, P_t *equals the whole front with probability* $\Omega(1)$ *and* $1-\Omega(1)$. *The expected time to find the whole Pareto front is* $n^{\Omega(n)}$.

The remainder of this subsection is devoted to the proof of Theorem 6.19. First, we define what a gap means and transition probabilities for mutations on the Pareto front that will be used in the remainder of this section.

Definition 6.20 (Gap)**.** *We say that a population* P_t *has a gap at position* i *if* $1^i0^{n-i} \notin P_t$, *but* $1^j0^{n-j} \in P_t$ *and* $1^k0^{n-k} \in P_t$ *for* $j < i < k$.

Definition 6.21 (Transition probabilities)**.** *We define*

$$p_k = n^{-k} \cdot \left(1-\frac{1}{n}\right)^{n-k} = \left(1-\frac{1}{n}\right)^{n} \cdot (n-1)^{-k}.$$

as the probability of jumping from any search point 1^i0^{n-i} *to* $1^{i+k}0^{n-i-k}$ *and* $1^{i-k}0^{n-i+k}$ *(if existent).*

Next, we show that, once the Pareto front has been reached, the Highest Diversity Contribution selection will always choose a parent x with an extreme number of ones. The

following lemma applies to a population P containing only search points on the Pareto front. This setting applies for the modified GSEMO once the Pareto front has been reached as then parent selection is only based on search points with a maximum L-dominant attribute, corresponding to points on the Pareto front.

Lemma 6.22. *Consider the Highest Diversity Contribution (HDC) selection mechanism, choosing as diversity metric either* $\mathrm{HVC}(x,P)$ *with reference point dominated by the point* $(-n^2,-n^2)$ *or* $\mathrm{CDC}(x,P)$ *on the function* LOTZ*, for a population P containing only search points on the Pareto front. Then the parent chosen by HDC will always either have a minimum or a maximum number of ones among all search points in P.*

Proof. Let us consider an individual x_i of the sorted population according to f_1, using the notation from Definition 6.3, and let us define $f_1(x_0) \leq -n^2$ and $f_2(x_{\mu+1}) \leq -n^2$ as reference point. For any point $x_i = 1^j 0^{n-j}$ where $1 < i < \mu$, the highest possible contribution that the point x_i can achieve is if it has as neighbours the points $x_1 = 0^n$ and $x_\mu = 1^n$, so we have $f_1(x_i) = j$, $f_1(x_{i-1}) = f_1(x_1) = 0$ and $f_2(x_i) = n-j$, $f_2(x_{i+1}) = f_2(x_\mu) = 0$. In this sense, by Definition 6.3, the highest possible contribution for x_i is $\mathrm{HVC}(x_i,P) \leq (j-0)\cdot(n-j-0) \leq j\cdot(n-j)$ and since j is restricted to $0 < j < n$, the maximum contribution possible for x_i is when $j = n/2$ and $n-j = n/2$ achieving $\mathrm{HVC}(x_i,P) \leq n^2/4 < n^2$.

For the case of points $x_1 = 0^n$ or $x_\mu = 1^n$, the hypervolume contribution of any of the these two points is at least n^2, since the lowest possible contribution for these points is obtained when the individuals x_2 or $x_{\mu-1}$ are contained in the population, then $\mathrm{HVC}(x_1,P) \leq n^2 \cdot 1$ (the same for x_μ). So we have that $\mathrm{HVC}(x_1,P) > \mathrm{HVC}(x_i,P)$ and $\mathrm{HVC}\big(x_\mu,P\big) > \mathrm{HVC}(x_i,P)$ for all $1 < i < \mu$.

For the case of CDC, both extreme points are always assigned an infinite diversity score, the highest possible score given by the CDC metric. So all intermediate individuals are ignored by the selection mechanism and HDC only selects the individual with the highest number of zeroes or ones in the population. □

We further show that gaps emerge and remain with constant probability.

Lemma 6.23. *In the setting of Theorem 6.19, with probability* $\Omega(1)$ *the modified GSEMO will evolve a population with a gap at position* $n/4 \leq i \leq 3n/4$.

The probability that this gap will remain at the first generation where the population contains both 0^n *and* 1^n *is* $\Omega(1)$.

Proof. In the following, we identify a search point $1^i 0^{n-i}$ with its index i. Note that, as long as no gap at index $n/4 \leq i \leq 3n/4$ is being created, the population spreads on this subset of the Pareto front as one Hamming path. This Hamming path is likely to start at some index

$n/4 \leq i \leq 3n/4$ and then spread towards lower and higher indices, but it could also start at an index $i < n/4$ and spread towards higher indices, or start at $i > 3n/4$ and spread towards lower indices. This means that, for every index $n/4+1 \leq j \leq 3n/4-1$ there will eventually be a search point 1^j0^{n-j} that will be chosen as parent, and (depending on the direction of the spread) at least one Hamming neighbour from $\{1^{j-1}0^{n-j+1}, 1^{j+1}0^{n-j-1}\}$ will not be contained in the population. Without loss of generality let this be $1^{j-1}0^{n-j+1}$ (the other case is symmetric). Then with probability at least p_2 a mutation of 1^j0^{n-j} will create a search point with smaller index than $j-1$, creating a gap at position $j-1$. With probability p_1 the modified GSEMO will create $1^{j-1}0^{n-j+1}$, and there will never be a gap at position $j-1$. Considering these two events, the conditional event of creating a gap, given that another search point on the front with smaller index is created, is at least

$$\frac{p_2}{p_1+p_2} \geq \frac{p_2}{p_1} = \frac{1}{n-1}.$$

The probability that at least one index $n/4+1 \leq j \leq 3n/4-1$ (of which there are $n/2-\mathrm{O}(1)$ many) will lead to the creation of a gap is at least

$$\begin{aligned} &1-\left(1-\frac{1}{n-1}\right)^{n/2-O(1)} \\ &= 1-\left(1-\frac{1}{n-1}\right)^{(n-1)/2} \cdot \left(1-\frac{1}{n-1}\right)^{O(1)} \geq 1-e^{-1/2}-\mathrm{O}(1/n) = \Omega(1) \end{aligned}$$

where the inequality used $\left(1-\frac{1}{n-1}\right)^{n-1} \leq 1/e$ and Bernoulli's inequality.

Now assume that a gap has been created at position g with $n/4 \leq g \leq 3n/4$. From here on, every index $1 \leq i \leq n-1$ has a chance to fill the gap if the population contains 1^i0^{n-i}, this search point is being chosen as parent, and mutation flips $|g-i|$ bits to create 1^g0^{n-g}, hence filling the gap. Note that, if 1^i0^{n-i} is picked as parent, and without loss of generality $i < g$, if mutation creates an offspring 1^j0^{n-j} with $j < i$ then 1^i0^{n-i} will never be selected as parent again, and the gap at g will never be filled from index i. Considering these two events, the conditional probability of *not* filling the gap from index $i < g$ is at least

$$\frac{p_1}{p_1+p_{g-i}} = \frac{(n-1)^{-1}}{(n-1)^{-1}+(n-1)^{i-g}} = \frac{1}{1+(n-1)^{1+(i-g)}}.$$

The above is $1/2$ if $i = g-1$ and at least $1-(n-1)^{1-|i-g|}$ for $i < g-1$. The same probability bounds hold for $i = g+1$ and $i > g+1$, respectively. Note that mutations from index i are independent from mutations on other indices, hence we can multiply probability bounds for

all indices $i \neq g$. Hence, the probability that the gap is *not* filled from any index $i \neq g$ is at least

$$\begin{aligned}
&\frac{1}{2} \cdot \prod_{1 \leq i < g-1} \left(1-(n-1)^{1-|i-g|}\right) \cdot \frac{1}{2} \cdot \prod_{g+1 < i \leq n-1} \left(1-(n-1)^{1-|i-g|}\right) \\
&\geq \frac{1}{4} \cdot \left(\prod_{d=2}^{\infty} \left(1-(n-1)^{1-d}\right)\right)^2 \\
&\geq \frac{1}{4} \cdot \left(1-\sum_{d=2}^{\infty} (n-1)^{1-d}\right)^2 \\
&= \frac{1}{4} \cdot \left(1-\sum_{d=1}^{\infty} (n-1)^{-d}\right)^2 \\
&= \frac{1}{4} \cdot \left(1-\frac{1}{n-2}\right)^2 = \Omega(1).
\end{aligned}$$

□

Now we can make use of Lemma 6.23 to prove Theorem 6.19.

Proof of Theorem 6.19. A sufficient condition for finding all points on the Pareto front in the setting of Theorem 6.19 is to always create a new point on the Pareto front via 1-bit mutations. Because global mutations are used, it is possible to create a new search point on the Pareto front by making a k-bit jump, for $k \geq 2$, with probability p_k.

Let E be the event that a new point is created on the Pareto front via 1-bit flip, and let B be the event of creating a new point on the Pareto front. We have $\text{Prob}(E) \geq p_1$, where the inequality becomes an equality if there is only one possible 1-bit flip applicable. The probability of event B is at most $\text{Prob}(B) \leq \text{Prob}(E) + 2p_2 + 2p_3 + \cdots + 2p_n$, taking into account all possible jump lengths, and the fact that it may be possible to make jumps in both directions. The conditional probability of event E is at least

$$\begin{aligned}
\text{Prob}(E \mid B) &\geq \frac{p_1}{p_1 + 2p_2 + 2p_3 + \ldots + 2p_n} \\
&= \frac{\left(1-\frac{1}{n}\right)^n \cdot (n-1)^{-1}}{\left(1-\frac{1}{n}\right)^n \cdot ((n-1)^{-1} + 2(n-1)^{-2} + \ldots + 2(n-1)^{-n})} \\
&= \frac{1}{1 + 2(n-1)^{-1} + \ldots + 2(n-1)^{n-1}} \\
&\geq \frac{1}{1 + 2\sum_{i=1}^{\infty}(n-1)^{-i}} \\
&= \frac{1}{1+\frac{2}{n-2}} = 1 - \frac{\frac{2}{n-2}}{1+\frac{2}{n-2}} = 1 - \frac{2}{n}.
\end{aligned}$$

Now, the same probability bounds hold for all i on the Pareto front. Mutations from point i are independent from mutations on other indices, hence we can multiply the probability for all indices i. Hence, the probability of creating a new point due to 1-bit mutation is at least

$$\prod_{i=1}^{n}\left(1-\frac{2}{n}\right) = \left(1-\frac{2}{n}\right)^n = \Omega(1).$$

Now we have proved that the modified GSEMO is able to find all points on the Pareto front via 1-bit mutation, and by Lemma 6.23, the modified GSEMO will create a gap at position $n/4 \leq i \leq 3n/4$ via more than 1-bit flip and this gap will remain after the points 0^n and 1^n have been found with probability $\Omega(1)$. At this point it will be necessary to flip at least $n/4$ specific number of bits from one of the extreme points in order to "fill" a gap. By Definition 6.21, the probability of making a $n/4$ jump from any extreme point is at most

$$p_{n/4} = \left(1-\frac{1}{n}\right)^n \cdot (n-1)^{-n/4} = n^{-\Omega(n)}.$$

Since the above probability bound holds for all current $n/4 \leq i \leq 3n/4$ gaps, we get that the algorithm requires at least exponential runtime $n^{\Omega(n)}$ to fill all the remaining i gaps. □

Note that the poor performance of the modified GSEMO is down to the choice of mutation operator, and the possibility of flipping multiple bits in one mutation. In contrast, SEMO using local mutations finds the Pareto front efficiently when HDC is used.

Theorem 6.24. *Consider SEMO with Highest Diversity Contribution, choosing as diversity metric either* CDC *or* HVC *with a reference point* $(-r,-r)$ *for* $r \geq 1$ *on the function* LOTZ. *Then the expected time for finding the whole Pareto front is* $\mathrm{O}(n^2)$.

Proof. We already know that SEMO reaches the Pareto front in expected time $\mathrm{O}(n^2)$. Afterwards, the population spreads on the Pareto front as one Hamming path. Let $P = \{1^i0^{n-i}, 1^{i+1}0^{n-i-1}, \ldots, 1^{j-1}0^{n-j+1}1^j0^{n-j}\}$ be the current population sorted according to the number of ones, with i, j being the minimum and maximum number of ones, respectively.

Then for any k with $i < k < j$ we have $\mathrm{HVC}(1^k0^{n-k}, P) = 1$ in addition to the contribution of the points $\mathrm{HVC}(1^i0^{n-i}, P) = i + r$ and $\mathrm{HVC}(1^j0^{n-j}, P) = n - j + r$. The latter two values simplify to r if $i = 0$ or $j = n$, respectively, that is, for 0^n and 1^n. For all values $i > 0$ we have $\mathrm{HVC}(1^i0^{n-i}, P) \geq r + 1$ and the same holds for $j < n$ implying $\mathrm{HVC}(1^i0^{n-j}, P) \geq r + 1$. This implies that the highest diversity contribution is always attained for a good search point, as long as the whole Pareto front has not been found yet. In other words, $p_{\text{good}} = 1$ and we obtain an upper bound of $\mathrm{O}(n^2)$ by following the arguments from Section 6.5.

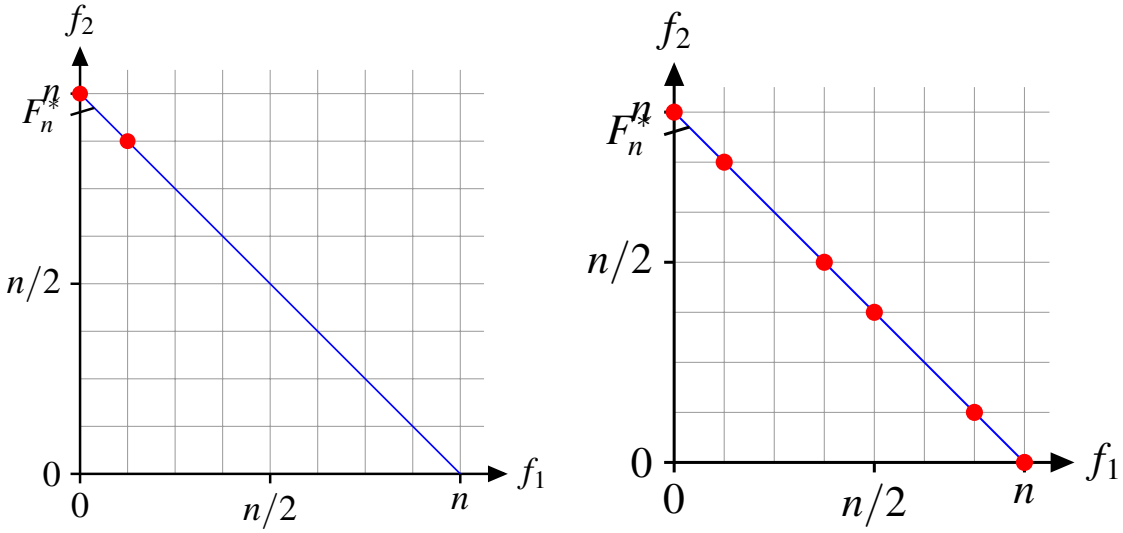
For CDC, we have $p_{\text{good}} \geq 1/2$ as both 1^i0^{n-i} and 1^j0^{n-j} have a crowding distance contribution of ∞, and at least one of them must be different from 0^n and 1^n. The upper bound of $\mathrm{O}(n^2)$ follows as before. □

6.7.2 NMUAR is Fast but Brittle

As mentioned previously and based on the results of Table 6.2, 6.3 and 6.4, NMUAR empirically performs well for SEMO, GSEMO and the modified GSEMO with any diversity metric in its different variants. No stagnation was detected during the experimental analysis made in Section 6.6. It seems that the selection mechanism performs better compared with the other selection approaches. Nevertheless, as an observant reviewer for Covantes Osuna et al. (2017) pointed out, it is possible to find populations where the probability of selecting a good parent is 0, and the analytical framework used in Sections 6.4 and 6.5 breaks down.

Two such populations are shown in Figure 6.2. In Figure 6.2a, 1^n is bad as it can not produce yet unseen points with local mutations on the Pareto front. However, depending on the choice of reference point, it may have the highest hypervolume contribution. The remaining point, while being good has the minimum hypervolume contribution. So, it is never picked as a parent by the NMUAR scheme. This means that the algorithm will never select a good search point, which leads to a stagnation state. Furthermore, Figure 6.2b shows that for the case of certain problem sizes, more points can be added on the Pareto front, such that all non-boundary points feature the same, minimum hypervolume contribution, and all

of these points are ignored by NMUAR, leaving only bad search points 0^n and 1^n that may be selected as parents. This also shows that the example from Figure 6.2a is not unique.



(a) $n = 8$ (b) $n = 7$

Figure 6.2: Examples of populations where NMUAR with CDC or HVC may only select bad individuals from $\{0^n, 1^n\}$ on ONEMINMAX and/or LOTZ, depending on the choice of reference point (all non-extreme points have the same score, NMUAR only selects extreme points).

In the following we show that, despite these risks, NMUAR is able to efficiently optimise both ONEMINMAX and LOTZ. First we define the following probability of selecting good individuals by providing some additional arguments on how to deal with different situations for p_{good}.

Lemma 6.25. *Let P denote the current population and $P' \subseteq P$ denote the population from which NMUAR selects uniformly at random. Consider* ONEMINMAX *or* LOTZ *and assume that the Pareto front has been reached, but P does not cover the whole front. The probability p_{good} of selecting a good individual using* CDC *or* HVC *with any reference point dominated by $(-1,-1)$ and NMUAR selection is $p_{\text{good}} = \Omega(1)$ if one of the following conditions is met:*

1. *P contains neither 0^n nor 1^n,*
2. *P contains a search point $x \in \{0^n, 1^n\}$ and x is good, or*
3. *P contains individuals with f_1-values i, $i+1$, and $i+2$, for some value $0 \leq i \leq n-2$.*

Proof. For HVC, note that any potential bad search points from $\{0,1\}^n \setminus \{0^n, 1^n\}$ will have the same diversity score of 1, which is minimal amongst all possible HVC values. All good

search points have a larger diversity score. This means that NMUAR will never choose a parent with minimum HVC score. The same applies to CDC where the minimum value depends on values $f_m^{\min}$ and $f_m^{\max}$.

The only risk is that NMUAR may choose a bad search point from $\{0^n, 1^n\}$. While the population does not contain any such points, $p_{\text{good}} = 1$. As long as the population contains a search point $x \in \{0^n, 1^n\}$ and x is good, we have $p_{\text{good}} \geq 1/2$ as there can only be one potential bad search point, namely $\bar{x}$, that has a chance to be selected.

The third condition implies that the individual with f_1-value $i+1$, which is a bad individual, has a minimum diversity contribution. Hence NMUAR will only remove bad individuals and all good individuals will remain. There will be at least one good search point $x \in P$ as long as the population does not cover the whole front; it can be found by scanning the Pareto front, starting at i and moving towards smaller f_1-values and starting at $i+2$ in the direction of larger f_1 values. In both directions either a search point from $\{0^n, 1^n\}$ or a good search point will be found. As the whole front has not been covered yet, at least one direction will result in a good search point. As there can be at most two bad search points in P' (0^n and 1^n), $p_{\text{good}} \geq 1/3$. □

Now we can prove the following theorem.

Theorem 6.26. *The expected time for SEMO and GSEMO to find the whole Pareto front on* OneMinMax *is bounded by* $O(n \log n)$ *for the NMUAR selection scheme with either* CDC *or* HVC *with a reference point defined as* $(-r, -r)$ *for* $r \geq 1$.

Proof. Let P and P' be as in Lemma 6.25. Whenever $p_{\text{good}} = \Omega(1)$ we can apply the arguments from Section 6.4, but we need to provide additional arguments to deal with possible settings where p_{good} is not guaranteed to be $\Omega(1)$. In order for $p_{\text{good}} \notin \Omega(1)$ to hold, we must have $P' \subseteq \{0^n, 1^n\}$ with all members of P' being bad. This implies that, if $1^n \in P'$, the population must contain a search point with $n-1$ ones (as otherwise 1^n would be good) and it cannot contain any search point with $n-2$ ones (as otherwise the third condition of Lemma 6.25 would be true). The same logic applies to 0^n and its neighbours.

We show that such a pathological case where $p_{\text{good}} \notin \Omega(1)$ is impossible for SEMO, due to our assumptions on the choice of reference point $(-r, -r)$. For OneMinMax all points are in the Pareto front, and because local mutations are being used, the population always contains all possible f_1 values in some integer range. Hence the population can only be $P = \{1^n, x\}$ where x has $n-1$ ones, or $P = \{0^n, x'\}$ where x' has a single one. W. l. o. g. the former is the case. Then CDC assigns value ∞ to both search points, hence $p_{\text{good}} = 1/2$. For HVC we have $\text{HVC}(1^n) = r$ and $\text{HVC}(x) = n - 1 + r$, hence $P' = \{x\}$ and $p_{\text{good}} = 1$.

For GSEMO, if $P' \subseteq \{0^n, 1^n\}$ and w. l. o. g. $1^n \in P'$, 1^n is selected as parent with probability at least $1/2$. Any mutation of 1^n flipping two arbitrary bits will create a search point with $n-2$ ones, which then fulfils the third condition from Lemma 6.25 for the next and all future populations. The expected waiting time for making this mutation is $O(1)$. Afterwards, $p_{\text{good}} = \Omega(1)$ by Lemma 6.25 and we obtain an upper bound for both SEMO and GSEMO of $O(n \log n)$ following the previous analyses from Section 6.4. □

Similar arguments can be used to prove that SEMO and the modified GSEMO can optimise LOTZ efficiently.

Theorem 6.27. *The expected time for SEMO and the modified GSEMO to find the whole Pareto front on* LOTZ *is bounded by* $O(n^2)$ *for the NMUAR selection scheme with either* CDC *or* HVC *with a reference point defined as* $(-r,-r)$ *for* $r \geq 1$.

Proof. By the same arguments as in the proof of Theorem 6.26, in order to have $p_{\text{good}} \notin \Omega(1)$ the population must contain 1^n and $1^{n-1}0$, but not $1^{n-2}00$, or the symmetric constellation involving 0^n, 10^{n-1}, and 110^{n-2}. For SEMO, arguing as in the proof of Theorem 6.26 the choice of reference point then implies that $\text{HVC}\left(1^{n-1}0\right) > \text{HVC}\left(1^n\right)$, hence we must always have $p_{\text{good}} = \Omega(1)$.

For the modified GSEMO, if $P' \subseteq \{0^n, 1^n\}$ and w. l. o. g. $1^n \in P'$, 1^n is selected as parent with probability at least $1/2$. The probability of a mutation turning 1^n into $1^{n-2}00$ is at least $1/(en^2)$, and once it occurs, it fulfils the third condition from Lemma 6.25 for the next and all future populations. The expected waiting time for making this mutation is $O(n^2)$. Afterwards, $p_{\text{good}} = \Omega(1)$ by Lemma 6.25 and we obtain an upper bound for both SEMO and the modified GSEMO of $O(n^2)$ following the previous analyses from Section 6.5. □

Note that NMUAR is not robust to the choice of the reference point. The proof of Lemma 6.25 has revealed a scenario where, with an asymmetric choice of the reference point, SEMO can get stuck.

Theorem 6.28. *There is a choice of reference point in the area dominated by* $(-1,-1)$ *such that SEMO with* HVC *and NMUAR selection has a positive probability of stagnating on* ONEMINMAX *and* LOTZ.

Proof. Choose the reference point as $(-n-1,-1)$. With positive probability, SEMO is initialised with 1^n. Then only offspring with an f_1 value of $n-1$ are accepted. Once the population equals $P = \{1^n, x\}$, where $f_1(x) = n-1$, we have $\text{HVC}\left(1^n\right) = n+1$ and $\text{HVC}\left(x\right) = n$, hence NMUAR will always choose 1^n as parent, leading to stagnation. □

6.8 Conclusions

Diversity plays a crucial role in the area of EMO. So far, diversity-based parent selection has not been the main focus on algorithm design. We have proposed a range of diversity-based parent selection schemes, aiming to speed up the spread on the Pareto front. We have demonstrated for two example functions, ONEMINMAX and LOTZ, that our new selection schemes can significantly speed up EMO algorithms. Our theoretical results show that a linear factor can be saved for the investigated settings and this is confirmed by our experimental results showing a speedup of one magnitude for problems of size $n = 100$.

We have analysed different selection schemes with different preference toward the individual's diversity contribution, from aggressive schemes that put a strong emphasis on individuals with the highest diversity contribution to more relaxed schemes that introduce a bias for more diversity, but still give all individuals a chance to be selected as parents.

The analysis has shown that very extreme schemes can lead to undesired results. For selection mechanisms that entail a rather extreme change of behaviour, such as Highest Diversity Contribution (HDC) and Non-Minimum Uniformly at Random (NMUAR), search may stagnate. On the other hand, our rank-based approaches as well as tournament selection are successful for ONEMINMAX and LOTZ, for both SEMO and GSEMO. Among these, the power law selection scheme is the fastest, hence we recommend this scheme as having the best trade-off between speed and risk. We believe the power law selection to also be beneficial for other problems as it has a high probability of selecting parents with the highest diversity contribution, but it also has a fat tail, allowing any individual to still be selected as parent with a reasonable probability.

Our theoretical analysis of stagnation behaviour has further revealed an interesting and quite natural setting where standard bit mutations are detrimental in MOEAs, compared to local mutations flipping only one bit. The performance difference is very drastic as the choice of the mutation operator decides between an expected polynomial time and exponential time for finding the whole Pareto front.

For future work, it would be interesting to study the benefit of diversity-based parent selection on more complex problems. From a theoretical perspective, combinatorial optimisation problems such as the travelling thief problem where our ranked-based approaches have already been used in Wu et al. (2018) or minimum spanning tress and covering problems for which SEMO has already been studied would be natural candidates. On the experimental side, it would be interesting to integrate the presented diversity-based parent selection methods into state-of-the-art EMO algorithms and to evaluate their performance on well-established benchmark sets.

Part IV

Conclusions and Outlook

Chapter 7

Conclusion

Looking back at the topics addressed in this thesis, its main goal was devoted to narrowing the gap between theory and practice by providing insights into the working principles of diversity-preserving mechanisms by testing their ability to find and maintain many local optima in the population, as well as their ability to escape from local optima with different basins of attraction by means of rigorous runtime analysis and empirical investigations.

In Chapter 1 we introduced the reader to the field of EAs and we motivated not only the importance of studying diversity-preserving mechanisms using theory and practice independently but also using both approaches together. Then, in Chapter 2, we introduce the mathematical techniques for the runtime analysis of randomised search heuristics (Section 2.3). In Chapter 3 we have surveyed rigorous runtime analyses of EAs with explicit or implicit diversity mechanisms. Here, we have seen that by comparing results for different example functions, the performance of the algorithms may lead to different results, mechanisms may work efficiently for one problem and may be ineffective for other problems, and vice versa. In any case, this review has shown that diversity can be beneficial for enhancing the global exploration capabilities of an EA. It is possible to improve the performance of implicit operators like crossover, improve the performance and robustness in dynamic optimisation, and improving the performance of MOEAs by speeding up the expected optimisation time by constant factors, polynomial factors, or even exponential factors. Finally, explicit diversity mechanisms can help to escape from local optimum as in the case of ageing and clearing. All these results increase our understanding of the effects certain mechanisms have on the performance of EAs, and the fact that exist many mechanisms that have been analysed show the interest for trying to understand how these mechanisms work. Proof of this is that our work has been published in high rated conferences and top journals (Section 1.1)

Chapter 4 covered three classical niching mechanisms, *probabilistic crowding*, *restricted tournament selection* and *clearing*. There, we presented rigorous theoretical runtime analyses

for these niching mechanisms in the context of the (μ+1) EA on TwoMax, we assessed their performance in comparison to other diversity mechanisms. In addition, we provided insights into the working principles of these mechanisms. In Section 4.1 we rigorously proved that probabilistic crowding is a complete disaster; fitness-proportional selection for survival selection resembled uniform selection, consequently, it is not even able to evolve search points that are significantly better than those found by random search. This result highlights the importance of scaling the fitness to enable probabilistic crowding to find both optima on TwoMax. We also think that the proof arguments used for probabilistic crowding can also be used to analyse more advanced versions of crowding (Galan and Mengshoel, 2010; Mengshoel et al., 2014). We choose to leave this analysis for future work.

Section 4.2 presented the results for restricted tournament selection. The results have shown that the behaviour of this niching technique varies a lot depending on how the population size and the window size are set. If μ and w are set too small, one subpopulation may takeover the individuals in other subpopulations. But if w is large enough, restricted tournament selection behaves similarly to deterministic crowding, in which case the probability of finding both optima converges to 1 very quickly as μ grows ($1-2^{-\mu+1}$). We leave as future work the theoretical analysis of the population dynamics of restricted tournament selection for intermediate values for w. Experimental results have shown that restricted tournament selection can optimise TwoMax for smaller w than the one required in our positive result.

The performance of the clearing mechanism was analysed in Section 4.3. For the results obtained in this section alone, we can conclude that the mechanism possesses desirable and powerful characteristics. For the case of small niches, clearing can exhaustively explore the landscape when the proper distance and parameters like clearing radius σ, niche capacity κ and population size μ are set and optimise all functions of unitation. In the case of large niches, clearing has been proved to be as strong as other niching mechanisms like deterministic crowding, restricted tournament selection and fitness sharing since it is able to find both optima of the test function TwoMax. Another important attribute from clearing is that is able to escape from local optima with different basins of attractions by moving/jumping between niches formed by the clearing radius.

It remains an open problem to theoretically analyse the population dynamics of clearing with more than 2 niches and to prove rigorously that clearing is effective across a much broader range of problems, including problems with more than 2 peaks. This involves obtaining more detailed insights into the dynamics of the population, including the distribution and evolution of the losers across multiple niches.

Chapter 5 covered an extensive empirical study involving 9 common diversity mechanisms on the Jansen-Zarges multimodal function classes. We have covered various degrees of

multimodality from 2 to 64 peaks, with peaks having equal or different heights, reflected in their basins of attraction. The results have shown that the plain (μ+1) EA, avoiding genotype and fitness duplicates cannot maintain several individuals distributed across the search space. And as mentioned in Section 4.1, probabilistic crowding showed the worst performance at it is unable to locate even a single peak.

Both fitness sharing approaches with genotypic distance showed a poor performance unlike its versions with phenotypic distance that showed an efficient performance on TWOMAX; this included the ability to climb down a peak and to tunnel through fitness valleys to reach other niches. Deterministic crowding, restricted tournament selection and clearing perform well for peaks with the same slope and height, much better than all other diversity mechanisms. Only for large numbers of peaks ($k = 64$) and different heights the performance starts to deteriorate. Finally, only clearing has shown the ability to escape from local optima since all other mechanisms are unable to accept worse search points or unable to tunnel through fitness valleys.

Finally, in Chapter 6 we have demonstrated for two example functions, ONEMINMAX and LOTZ, that our ranking-based diversity selection mechanisms can significantly speed up EMO algorithms. Our theoretical results show that a linear factor can be saved for the investigated settings and this is confirmed by our experimental results showing a speedup of one magnitude for problems of size $n = 100$. Our analysis has shown that very extreme schemes can lead to undesired results. For selection mechanisms that entail a rather extreme change of behaviour, such as highest diversity contribution and non-minimum uniformly at random, the search may stagnate. This analysis has revealed an interesting and quite natural setting where standard bit mutations are detrimental in MOEAs, compared to local mutations. The performance difference is very drastic as the choice of the mutation operator decides between an expected polynomial time and exponential time for finding the whole Pareto front. On the other hand, our rank-based approaches as well as tournament selection are successful for ONEMINMAX and LOTZ, for both SEMO and GSEMO.

For future work, it would be interesting to study the benefit of diversity-based parent selection on more complex problems. From a theoretical perspective, combinatorial optimisation problems such as minimum spanning tress and covering problems for which SEMO has already been studied would be natural candidates. On the experimental side, it would be interesting to integrate the presented diversity-based parent selection methods into state-of-the-art EMO algorithms like the NSGA-II or a steady-state variant of the NSGA-II and to evaluate their performance on well-established benchmark sets.

As general overview, this thesis has yielded many contributions for theory and practice. The main contribution of this thesis was the plethora of results for diversity-preserving

mechanisms. We show that maintaining and promoting diversity in an EA is a very important task. Diversity is crucial in EAs to enable global exploration and to avoid poor performance due to premature convergence. This research has also revealed that some of the mechanisms may be helpful for some cases and not for others by providing a detailed description of the population dynamics and parameters settings by rigorous theoretical runtime analysis and experimental supplements. Aside from all the answers and insides provided in this thesis where we define when/why/how diversity is key in the evolutionary process, we recognise that there is still an open question: which mechanisms perform well for certain problem and more important why (if we take into consideration that this thesis focusses in a fraction of them as mentioned in Section 3.1 from Chapter 3). This open question lead to the conclusion that the analysis of diversity mechanisms and the population dynamics remains an interesting, challenging, and fruitful research area.

References

Abramowitz, M. (1974). *Handbook of Mathematical Functions, With Formulas, Graphs, and Mathematical Tables*. Dover Publications, Incorporated.

Alba, E. and Dorronsoro, B. (2005). The exploration/exploitation tradeoff in dynamic cellular genetic algorithms. *IEEE Transactions on Evolutionary Computation*, 9(2):126–142.

Arora, S., Hazan, E., and Kale, S. (2012). The Multiplicative Weights Update Method: a Meta-Algorithm and Applications. *Theory of Computing*, 8(6):121–164.

Auger, A., Bader, J., Brockhoff, D., and Zitzler, E. (2012). Hypervolume-based multiobjective optimization: Theoretical foundations and practical implications. *Theoretical Computer Science*, 425:75–103.

Auger, A. and Doerr, B., editors (2011). *Theory of Randomized Search Heuristics — Foundations and Recent Developments*, volume 1 of *Theoretical Computer Science*. World Scientific Publishing Co., Inc.

Bäck, T., Fogel, D. B., and Michalewicz, Z., editors (1997). *Handbook of Evolutionary Computation*. IOP Publishing Ltd., 1st edition.

Ballester, P. J. and Carter, J. N. (2004). An Effective Real-Parameter Genetic Algorithm with Parent Centric Normal Crossover for Multimodal Optimisation. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '04, pages 901–913. Springer Berlin Heidelberg.

Battiti, R. (1996). Reactive Search: Toward Self-Tuning Heuristics. In *Modern Heuristic Search Methods*, chapter 4, pages 61–83. John Wiley & Sons, Inc.

Beyer, H.-G. and Schwefel, H.-P. (2002). Evolution strategies – A comprehensive introduction. *Natural Computing*, 1(1):3–52.

Birattari, M., Paquete, L., Stützle, T., and Varrentrapp, K. (2001). Classification of Metaheuristics and Design of Experiments for the Analysis of Components. Technical Report AIDA-01-05, Intellektik Darmstadt University of Technology Darmstadt, Germany.

Bleuler, S., Brack, M., Thiele, L., and Zitzler, E. (2001). Multiobjective genetic programming: reducing bloat using SPEA2. In *IEEE Congress on Evolutionary Computation*, volume 1, pages 536–543.

Blum, C. and Roli, A. (2003). Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. *ACM Comput. Surv.*, 35(3):268–308.

Boussaïd, I., Lepagnot, J., and Siarry, P. (2013). A survey on optimization metaheuristics. *Information Sciences*, 237:82 – 117.

Bremermann, H. J. (1962). Optimization Through Evolution and Recombination. In *Self-organizing Systems*, pages 93–106. Pergamon Press, Oxford.

Briest, P., Brockhoff, D., Degener, S., Englert, M., Gunia, C., Heering, O., Jansen, T., Leifhelm, M., Plociennik, K., Röglin, H., Schweer, A., Sudholt, D., Tannenbaum, S., and Wegener, I. (2004a). Evolutionäre Algorithmen zwischen experimenteller und theoretischer Analyse – Endbericht der Projektgruppe 427. Technical report, Universität Dortmund.

Briest, P., Brockhoff, D., Degener, S., Englert, M., Gunia, C., Heering, O., Jansen, T., Leifhelm, M., Plociennik, K., Röglin, H., Schweer, A., Sudholt, D., Tannenbaum, S., and Wegener, I. (2004b). FrEAK – Free Evolutionary Algorithm Kit. Available from http://sourceforge.net/projects/freak427/.

Bringmann, K. and Friedrich, T. (2010). An Efficient Algorithm for Computing Hypervolume Contributions. *Evolutionary Computation*, 18(3):383–402.

Bringmann, K. and Friedrich, T. (2012). Approximating the least hypervolume contributor: NP-hard in general, but fast in practice. *Theoretical Computer Science*, 425:104–116.

Brockhoff, D. (2012). Theoretical Aspects of Evolutionary Multiobjective Optimization. In *Theory of Randomized Search Heuristics — Foundations and Recent Developments*, volume 1, chapter 4, pages 101–139. World Scientific Publishing.

Castrogiovanni, M., Nicosia, G., and Rascuná, R. (2007). Experimental Analysis of the Aging Operator for Static and Dynamic Optimisation Problems. In *Knowledge-Based Intelligent Information and Engineering Systems*, pages 804–811. Springer Berlin Heidelberg.

Chaiyaratana, N., Piroonratana, T., and Sangkawelert, N. (2007). Effects of diversity control in single-objective and multi-objective genetic algorithms. *Journal of Heuristics*, 13(1):1–34.

Chiong, R., editor (2009). *Nature-Inspired Algorithms for Optimisation*, volume 193 of *Studies in Computational Intelligence*. Springer-Verlag Berlin Heidelberg, 1 edition.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to Algorithms*. The MIT Press, 3rd edition.

Corus, D. and Oliveto, P. S. (2017). Standard Steady State Genetic Algorithms Can Hillclimb Faster than Mutation-only Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation*. Early Access.

Cotta-Porras, C. (1998). A study of hybridisation techniques and their application to the design of evolutionary algorithms. *AI Communications*, 11(3/4).

Covantes Osuna, E., Gao, W., Neumann, F., and Sudholt, D. (2017). Speeding Up Evolutionary Multi-objective Optimisation Through Diversity-based Parent Selection. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '17, pages 553–560. ACM.

Covantes Osuna, E., Gao, W., Neumann, F., and Sudholt, D. (2018). Design and analysis of diversity-based parent selection schemes for speeding up evolutionary multi-objective optimisation. *Theoretical Computer Science*. To appear. Preprint available from http://arxiv.org/abs/1805.01221.

Covantes Osuna, E. and Sudholt, D. (2017). Analysis of the Clearing Diversity-Preserving Mechanism. In *Proceedings of the 14th ACM/SIGEVO Conference on Foundations of Genetic Algorithms*, FOGA '17, pages 55–63. ACM.

Covantes Osuna, E. and Sudholt, D. (2018a). Empirical Analysis of Diversity-Preserving Mechanisms on Example Landscapes for Multimodal Optimisation. In *Parallel Problem Solving from Nature – PPSN XV*, pages 207–219. Springer International Publishing.

Covantes Osuna, E. and Sudholt, D. (2018b). On the Runtime Analysis of the Clearing Diversity-Preserving Mechanism. *Evolutionary Computation*. To appear. Preprint available from http://arxiv.org/abs/1803.09715.

Covantes Osuna, E. and Sudholt, D. (2018c). Runtime Analysis of Probabilistic Crowding and Restricted Tournament Selection for Bimodal Optimisation. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '18, pages 929–936. ACM.

Črepinšek, M., Liu, S.-H., and Mernik, M. (2013). Exploration and Exploitation in Evolutionary Algorithms: A Survey. *ACM Comput. Surv.*, 45(3):35:1–35:33.

Cutello, V., Nicosia, G., and Pavone, M. (2003). A Hybrid Immune Algorithm with Information Gain for the Graph Coloring Problem. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '03, pages 171–182. Springer Berlin Heidelberg.

Cutello, V., Nicosia, G., and Pavone, M. (2007a). An immune algorithm with stochastic aging and kullback entropy for the chromatic number problem. *Journal of Combinatorial Optimization*, 14(1):9–33.

Cutello, V., Nicosia, G., Romeo, M., and Oliveto, P. S. (2007b). On the Convergence of Immune Algorithms. In *2007 IEEE Symposium on Foundations of Computational Intelligence*, FOCCI 2007, pages 409–415.

Dang, D.-C., Friedrich, T., Kötzing, T., Krejca, M. S., Lehre, P. K., Oliveto, P. S., Sudholt, D., and Sutton, A. M. (2016a). Emergence of Diversity and Its Benefits for Crossover in Genetic Algorithms. In *Parallel Problem Solving from Nature – PPSN XIV*, pages 890–900. Springer International Publishing.

Dang, D.-C., Friedrich, T., Kötzing, T., Krejca, M. S., Lehre, P. K., Oliveto, P. S., Sudholt, D., and Sutton, A. M. (2016b). Escaping Local Optima with Diversity Mechanisms and Crossover. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '16, pages 645–652. ACM.

Dang, D. C., Friedrich, T., Kötzing, T., Krejca, M. S., Lehre, P. K., Oliveto, P. S., Sudholt, D., and Sutton, A. M. (2017a). Escaping Local Optima Using Crossover with Emergent Diversity. *IEEE Transactions on Evolutionary Computation*, 22(3):484–497.

Dang, D.-C., Jansen, T., and Lehre, P. K. (2017b). Populations Can Be Essential in Tracking Dynamic Optima. *Algorithmica*, 78(2):660–680.

Das, S., Maity, S., Qu, B.-Y., and Suganthan, P. (2011). Real-parameter evolutionary multimodal optimization — A survey of the state-of-the-art. *Swarm and Evolutionary Computation*, 1(2):71–88.

de Castro, L. N. and Timmis, J. (2002). *Artificial Immune Systems: A New Computational Intelligence Approach*. Springer-Verlag London, 1st edition.

De Felice, M., Meloni, S., and Panzieri, S. (2011). Effect of topology on diversity of spatially structured evolutionary algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '11, pages 1579–1586. ACM.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

Doerr, B. (2012). Analyzing Randomized Search Heuristics: Tools from Probability Theory. In *Theory of Randomized Search Heuristics*, chapter 1, pages 1–20. World Scientific.

Doerr, B. (2018). Probabilistic Tools for the Analysis of Randomized Optimization Heuristics. *ArXiv e-prints*. Preprint available from http://arxiv.org/abs/1801.06733.

Doerr, B., Fouz, M., and Witt, C. (2011). Sharp Bounds by Probability-generating Functions and Variable Drift. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '11, pages 2083–2090. ACM.

Doerr, B., Gao, W., and Neumann, F. (2016). Runtime Analysis of Evolutionary Diversity Maximization for OneMinMax. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '16, pages 557–564. ACM.

Doerr, B. and Goldberg, L. A. (2010). Drift Analysis with Tail Bounds. In *Parallel Problem Solving from Nature – PPSN XI*, pages 174–183. Springer Berlin Heidelberg.

Doerr, B., Hebbinghaus, N., and Neumann, F. (2007a). Speeding Up Evolutionary Algorithms through Asymmetric Mutation Operators. *Evolutionary Computation*, 15(4):401–410.

Doerr, B. and Johannsen, D. (2007). Adjacency List Matchings: An Ideal Genotype for Cycle Covers. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '07, pages 1203–1210. ACM.

Doerr, B., Johannsen, D., and Winzen, C. (2012). Multiplicative Drift Analysis. *Algorithmica*, 64(4):673–697.

Doerr, B., Klein, C., and Storch, T. (2007b). Faster Evolutionary Algorithms by Superior Graph Representation. In *2007 IEEE Symposium on Foundations of Computational Intelligence*, pages 245–250.

Droste, S., Jansen, T., and Wegener, I. (2002). On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science*, 276(1):51–81.

Eiben, A. E. and Smith, E. (2003). *Introduction to Evolutionary Computing*. Natural Computing Series. Springer-Verlag Berlin Heidelberg, 2nd edition.

Ewens, W. (2004). *Mathematical Population Genetics 1: Theoretical Introduction*. Interdisciplinary Applied Mathematics. Springer New York.

Fischer, S. and Wegener, I. (2005). The one-dimensional Ising model: Mutation versus recombination. *Theoretical Computer Science*, 344(2):208–225.

Floreano, D. and Mattiussi, C. (2008). *Bio-Inspired Artificial Intelligence Theories, Methods, and Technologies*. Intelligent Robotics and Autonomous Agents. MIT Press.

Fogel, D. B. (1995). *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. Wiley-IEEE Press, 3rd edition.

Fogel, D. B. (1997). The Advantages of Evolutionary Computation. In *Biocomputing and Emergent Computation: Proceedings of BCEC97*, pages 1–11. World Scientific Press.

Fogel, L. J. (1999). *Intelligence Through Simulated Evolution: Forty Years of Evolutionary Programming*. John Wiley & Sons, Inc.

Fonseca, C. M. and Fleming, P. J. (1995). An Overview of Evolutionary Algorithms in Multiobjective Optimization. *Evolutionary Computation*, 3(1):1–16.

Friedrich, T., Hebbinghaus, N., and Neumann, F. (2007). Rigorous Analyses of Simple Diversity Mechanisms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '07, pages 1219–1225. ACM.

Friedrich, T., Horoba, C., and Neumann, F. (2011). Illustration of fairness in evolutionary multi-objective optimization. *Theoretical Computer Science*, 412(17):1546–1556.

Friedrich, T., Kötzing, T., and Wagner, M. (2017). A Generic Bet-and-Run Strategy for Speeding Up Stochastic Local Search. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 801–807.

Friedrich, T., Oliveto, P. S., Sudholt, D., and Witt, C. (2009). Analysis of Diversity-preserving Mechanisms for Global Exploration. *Evolutionary Computation*, 17(4):455–476.

Galan, S. F. and Mengshoel, O. J. (2010). Generalized Crowding for Genetic Algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '10, pages 775–782. ACM.

Gao, W. and Neumann, F. (2014). Runtime Analysis for Maximizing Population Diversity in Single-objective Optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '14, pages 777–784. ACM.

García-Hernández, L., Palomo-Romero, J. M., Salas-Morera, L., Arauzo-Azofra, A., and Pierreval, H. (2015). A novel hybrid evolutionary approach for capturing decision maker knowledge into the unequal area facility layout problem. *Expert Systems with Applications*, 42(10):4697–4708.

García-Martínez, C., Lozano, M., and Rodríguez-Díaz, F. (2012). A simulated annealing method based on a specialised evolutionary algorithm. *Applied Soft Computing*, 12(2):573–588.

Gendreau, M. and Potvin, J.-Y., editors (2010). *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*. Springer US, 2 edition.

Ghosh, A., Tsutsui, S., and Tanaka, H. (1996). Individual aging in genetic algorithms. In *Proceedings of the Australian New Zealand Conference on Intelligent Information Systems*, ANZIIS 96, pages 276–279.

Giacobini, M., Tomassini, M., and Tettamanzi, A. (2005). Takeover Time Curves in Random and Small-world Structured Populations. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '05, pages 1333–1340. ACM.

Giel, O. (2003). Expected runtimes of a simple multi-objective evolutionary algorithm. In *IEEE Congress on Evolutionary Computation*, volume 3, pages 1918–1925.

Giel, O. and Lehre, P. K. (2010). On the Effect of Populations in Evolutionary Multi-objective Optimisation. *Evolutionary Computation*, 18(3):335–356.

Giel, O. and Wegener, I. (2003). Evolutionary Algorithms and the Maximum Matching Problem. In *STACS 2003*, pages 415–426. Springer Berlin Heidelberg.

Glibovets, N. N. and Gulayeva, N. M. (2013). A Review of Niching Genetic Algorithms for Multimodal Function Optimization. *Cybernetics and Systems Analysis*, 49(6):815–820.

Glover, F. and Laguna, M. (1997). *Tabu Search*. Springer US, 1 edition.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., 1st edition.

Goldberg, D. E. and Richardson, J. (1987). Genetic Algorithms with Sharing for Multimodal Function Optimization. In *Proceedings of the Second International Conference on Genetic Algorithms on Genetic Algorithms and Their Application*, pages 41–49. L. Erlbaum Associates Inc.

Hajek, B. (1982). Hitting-Time and Occupation-Time Bounds Implied by Drift Analysis with Applications. *Advances in Applied Probability*, 14(3):502–525.

Happ, E., Johannsen, D., Klein, C., and Neumann, F. (2008). Rigorous Analyses of Fitness-proportional Selection for Optimizing Linear Functions. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '08, pages 953–960. ACM.

Harik, G. R. (1995). Finding Multimodal Solutions Using Restricted Tournament Selection. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 24–31. Morgan Kaufmann Publishers Inc.

He, J. and Yao, X. (2001). Drift analysis and average time complexity of evolutionary algorithms. *Artificial Intelligence*, 127(1):57 – 85.

He, J. and Yao, X. (2003). Towards an analytic framework for analysing the computation time of evolutionary algorithms. *Artificial Intelligence*, 145(1):59 – 97.

He, J. and Yao, X. (2004). A study of drift analysis for estimating computation time of evolutionary algorithms. *Natural Computing*, 3(1):21–35.

Holland, J. H. (1973). Genetic algorithms and the optimal allocation of trials. *SIAM Journal on Computing*, 2(2):88–105.

Horoba, C., Jansen, T., and Zarges, C. (2009). Maximal Age in Randomized Search Heuristics with Aging. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '09, pages 803–810. ACM.

Horoba, C. and Neumann, F. (2008). Benefits and Drawbacks for the Use of Epsilon-dominance in Evolutionary Multi-objective Optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '08, pages 641–648. ACM.

Horoba, C. and Neumann, F. (2010). Approximating Pareto-Optimal Sets Using Diversity Strategies in Evolutionary Multi-Objective Optimization. In *Advances in Multi-Objective Nature Inspired Computing*, pages 23–44. Springer Berlin Heidelberg.

Hoyweghen, C. V., Goldberg, D. E., and Naudts, B. (2002). From TwoMax to the Ising Model: Easy and Hard Symmetrical Problems. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '02, pages 626–633. Morgan Kaufmann Publishers Inc.

Hutter, M. and Legg, S. (2006). Fitness uniform optimization. *IEEE Transactions on Evolutionary Computation*, 10(5):568–589.

Ising, E. (1925). Beitrag zur Theorie des Ferromagnetismus. *Zeitschrift für Physik*, 31(1):253–258. Translation to english available: https://goo.gl/sj4ibn.

Jägersküpper, J. (2007). Algorithmic analysis of a basic evolutionary algorithm for continuous optimization. *Theoretical Computer Science*, 379(3):329 – 347.

Jansen and Wegener (2002). The Analysis of Evolutionary Algorithms—A Proof That Crossover Really Can Help. *Algorithmica*, 34(1):47–66.

Jansen, T. (2002). On the analysis of dynamic restart strategies for evolutionary algorithms. In *Parallel Problem Solving from Nature – PPSN VII*, pages 33–43. Springer Berlin Heidelberg.

Jansen, T. (2013). *Analyzing Evolutionary Algorithms: The Computer Science Perspective*. Natural Computing Series. Springer-Verlag Berlin Heidelberg, 1 edition.

Jansen, T. and Wegener, I. (2001). Evolutionary algorithms - how to cope with plateaus of constant fitness and when to reject strings of the same fitness. *IEEE Transactions on Evolutionary Computation*, 5(6):589–599.

Jansen, T. and Wegener, I. (2005). Real royal road functions–where crossover provably is essential. *Discrete Applied Mathematics*, 149(1):111–125.

Jansen, T. and Zarges, C. (2009). Comparing Different Aging Operators. In *Artificial Immune Systems*, pages 95–108. Springer Berlin Heidelberg.

Jansen, T. and Zarges, C. (2010a). Aging Beyond Restarts. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '10, pages 705–712. ACM.

Jansen, T. and Zarges, C. (2010b). On the Benefits of Aging and the Importance of Details. In *Artificial Immune Systems*, pages 61–74. Springer Berlin Heidelberg.

Jansen, T. and Zarges, C. (2011a). Analyzing different variants of immune inspired somatic contiguous hypermutations. *Theoretical Computer Science*, 412(6):517–533.

Jansen, T. and Zarges, C. (2011b). On the role of age diversity for effective aging operators. *Evolutionary Intelligence*, 4(2):99–125.

Jansen, T. and Zarges, C. (2014). Evolutionary Algorithms and Artificial Immune Systems on a Bi-stable Dynamic Optimisation Problem. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '14, pages 975–982. ACM.

Jansen, T. and Zarges, C. (2016). Example Landscapes to Support Analysis of Multimodal Optimisation. In *Parallel Problem Solving from Nature – PPSN XIV*, pages 792–802. Springer International Publishing.

Johannsen, D. (2010). *Random Combinatorial Structures and Randomized Search Heuristics*. PhD thesis, Universität des Saarlandes, Saarbrücken, Germany and the Max-Planck-Institut für Informatik.

Kazarlis, S., Papadakis, S., Theocharis, J., and Petridis, V. (2001). Microgenetic algorithms as generalized hill-climbing operators for GA optimization. *IEEE Transactions on Evolutionary Computation*, 5(3):204–217.

Kötzing, T., Lissovoi, A., and Witt, C. (2015). (1+1) EA on Generalized Dynamic OneMax. In *Proceedings of the 2015 ACM Conference on Foundations of Genetic Algorithms XIII*, FOGA '15, pages 40–51. ACM.

Kötzing, T. and Molter, H. (2012). ACO Beats EA on a Dynamic Pseudo-Boolean Function. In *Parallel Problem Solving from Nature – PPSN XII*, pages 113–122. Springer Berlin Heidelberg.

Kötzing, T., Sudholt, D., and Theile, M. (2011). How Crossover Helps in Pseudo-Boolean Optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '11, pages 989–996. ACM.

Koumousis, V. and Katsaras, C. (2006). A saw-tooth genetic algorithm combining the effects of variable population size and reinitialization to enhance performance. *IEEE Transactions on Evolutionary Computation*, 10(1):19–28.

Koza, J. R. (1994). Genetic programming as a means for programming computers by natural selection. *Statistics and Computing*, 4(2):87–112.

Kubota, N. and Fukuda, T. (1997). Genetic algorithms with age structure. *Soft Computing*, 1(4):155–161.

Lässig, J. and Sudholt, D. (2010). The Benefit of Migration in Parallel Evolutionary Algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '10, pages 1105–1112. ACM.

Lässig, J. and Sudholt, D. (2013). Design and analysis of migration in parallel evolutionary algorithms. *Soft Computing*, 17(7):1121–1144.

Lässig, J. and Sudholt, D. (2014). Analysis of speedups in parallel evolutionary algorithms and (1+λ) EAs for combinatorial optimization. *Theoretical Computer Science*, 551:66–83.

Laumanns, M., Thiele, L., Deb, K., and Zitzler, E. (2002). Combining Convergence and Diversity in Evolutionary Multiobjective Optimization. *Evolutionary Computation*, 10(3):263–282.

Laumanns, M., Thiele, L., and Zitzler, E. (2004). Running time analysis of multiobjective evolutionary algorithms on pseudo-Boolean functions. *IEEE Transactions on Evolutionary Computation*, 8(2):170–182.

Laumanns, M., Zitzler, E., and Thiele, L. (2001). On The Effects of Archiving, Elitism, and Density Based Selection in Evolutionary Multi-objective Optimization. In *Evolutionary Multi-Criterion Optimization*, pages 181–196. Springer Berlin Heidelberg.

Lee, K. Y. and El-Sharkawi, M. A., editors (2008). *Modern Heuristic Optimization Techniques: Theory and Applications to Power Systems*. IEEE Press Series on Power Engineering. Wiley - IEEE Press.

Lehre, P. K. and Witt, C. (2012). Black-Box Search by Unbiased Variation. *Algorithmica*, 64(4):623–642.

Li, J.-P., Balazs, M. E., Parks, G. T., and Clarkson, P. J. (2002). A Species Conserving Genetic Algorithm for Multimodal Function Optimization. *Evolutionary Computation*, 10(3):207–234.

Li, J.-P., Balazs, M. E., Parks, G. T., and Clarkson, P. J. (2003). Erratum: A species conserving genetic algorithm for multimodal function optimization. *Evolutionary Computation*, 11(1):107–109.

Li, X., Engelbrecht, A., and Epitropakis, M. G. (2013). Benchmark Functions for CEC'2013 Special Session and Competition on Niching Methods for Multimodal Function Optimization. Technical report, Evolutionary Computation and Machine Learning Group, RMIT University, Australia.

Liang, Y. and Leung, K.-S. (2011). Genetic Algorithm with adaptive elitist-population strategies for multimodal function optimization. *Applied Soft Computing*, 11(2):2017–2034. The Impact of Soft Computing for the Progress of Artificial Intelligence.

Lissovoi, A. and Witt, C. (2015). On the Utility of Island Models in Dynamic Optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '15, pages 1447–1454. ACM.

Lissovoi, A. and Witt, C. (2017). A Runtime Analysis of Parallel Evolutionary Algorithms in Dynamic Optimization. *Algorithmica*, 78(2):641–659.

Liu, S.-H., Mernik, M., and Bryant, B. R. (2009). To Explore or to Exploit: An Entropy-driven Approach for Evolutionary Algorithms. *Int. J. Know.-Based Intell. Eng. Syst.*, 13(3,4):185–206.

Lozano, M. and García-Martínez, C. (2010). Hybrid metaheuristics with evolutionary algorithms specializing in intensification and diversification: Overview and progress report. *Computers & Operations Research*, 37(3):481–497.

Lozano, M., Herrera, F., and Cano, J. R. (2005). Replacement Strategies to Maintain Useful Diversity in Steady-State Genetic Algorithms. In *Soft Computing: Methodologies and Applications*, volume 32 of *Advances in Soft Computing*, pages 85–96. Springer Berlin Heidelberg.

Lozano, M., Herrera, F., Krasnogor, N., and Molina, D. (2004). Real-coded Memetic Algorithms with Crossover Hill-climbing. *Evolutionary Computation*, 12(3):273–302.

Mahfoud, S. W. (1995). *Niching methods for genetic algorithms*. PhD thesis, University of Illinois at Urbana-Champaign.

Mann, H. B. and Whitney, D. R. (1947). On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. *The Annals of Mathematical Statistics*, 18(1):50–60.

Mengsheol, O. and Goldberg, D. (1999). Probabilistic Crowding: Deterministic Crowding with Probabilistic Replacement. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '99, pages 409–416.

Mengshoel, O. J., Galán, S. F., and de Dios, A. (2014). Adaptive generalized crowding for genetic algorithms. *Information Sciences*, 258:140–159.

Mengshoel, O. J. and Goldberg, D. E. (2008). The Crowding Approach to Niching in Genetic Algorithms. *Evolutionary Computation*, 16(3):315–354.

Mitchell, M. (1998). *An Introduction to Genetic Algorithms*. MIT Press.

Mitzenmacher, M. and Upfal, E. (2005). *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press.

Motwani, R. and Raghavan, P. (1995). *Randomized Algorithms*. Cambridge University Press.

Naudts, B. and Naudts, J. (1998). The effect of spin-flip symmetry on the performance of the simple GA. In *Parallel Problem Solving from Nature – PPSN V*, pages 67–76. Springer Berlin Heidelberg.

Neumann, F. (2008). Expected runtimes of evolutionary algorithms for the Eulerian cycle problem. *Computers & Operations Research*, 35(9):2750–2759.

Neumann, F., Oliveto, P. S., Rudolph, G., and Sudholt, D. (2011). On the Effectiveness of Crossover for Migration in Parallel Evolutionary Algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '11, pages 1587–1594. ACM.

Neumann, F., Oliveto, P. S., and Witt, C. (2009). Theoretical Analysis of Fitness-proportional Selection: Landscapes and Efficiency. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '09, pages 835–842. ACM.

Neumann, F. and Witt, C. (2010). *Bioinspired Computation in Combinatorial Optimization: Algorithms and Their Computational Complexity*. Natural Computing Series. Springer-Verlag Berlin Heidelberg, 1 edition.

Nguyen, A. Q., Sutton, A. M., and Neumann, F. (2015). Population size matters: Rigorous runtime results for maximizing the hypervolume indicator. *Theoretical Computer Science*, 561:24–36.

Noman, N. and Iba, H. (2008). Accelerating Differential Evolution Using an Adaptive Local Search. *IEEE Transactions on Evolutionary Computation*, 12(1):107–125.

Oliveto, P. S., He, J., and Yao, X. (2008). Analysis of population-based evolutionary algorithms for the vertex cover problem. In *IEEE Congress on Evolutionary Computation*, pages 1563–1570.

Oliveto, P. S. and Sudholt, D. (2014). On the Runtime Analysis of Stochastic Ageing Mechanisms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '14, pages 113–120. ACM.

Oliveto, P. S., Sudholt, D., and Zarges, C. (2014). On the Runtime Analysis of Fitness Sharing Mechanisms. In *Parallel Problem Solving from Nature – PPSN XIII*, pages 932–941. Springer International Publishing.

Oliveto, P. S. and Witt, C. (2011). Simplified Drift Analysis for Proving Lower Bounds in Evolutionary Computation. *Algorithmica*, 59(3):369–386.

Oliveto, P. S. and Witt, C. (2012). Erratum: Simplified Drift Analysis for Proving Lower Bounds in Evolutionary Computation. *ArXiv e-prints*.

Oliveto, P. S. and Witt, C. (2015). Improved time complexity analysis of the Simple Genetic Algorithm. *Theoretical Computer Science*, 605:21–41.

Oliveto, P. S. and Zarges, C. (2015). Analysis of diversity mechanisms for optimisation in dynamic environments with low frequencies of change. *Theoretical Computer Science*, 561:37–56.

Pelikan, M. and Goldberg, D. E. (2000). Genetic Algorithms, Clustering, and the Breaking of Symmetry. In *Parallel Problem Solving from Nature – PPSN VI*, pages 385–394. Springer Berlin Heidelberg.

Pérez Heredia, J. (2017). *A Computational View on Natural Evolution: On the Rigorous Analysis of the Speed of Adaptation*. PhD thesis, University of Sheffield.

Pétrowski, A. (1996). A clearing procedure as a niching method for genetic algorithms. In *Proceedings of IEEE International Conference on Evolutionary Computation*, pages 798–803.

Pétrowski, A. (1997a). A New Selection Operator Dedicated to Speciation. In *Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA97)*, pages 144–151. Morgan Kaufmann.

Pétrowski, A. (1997b). An Efficient Hierarchical Clustering Technique for Speciation. In *Artificielle-1997*, pages 22–29.

Phan, D. H., Suzuki, J., and Boonma, P. (2011). SMSP-EMOA: Augmenting SMS-EMOA with the Prospect Indicator for Multiobjective Optimization. In *2011 IEEE 23rd International Conference on Tools with Artificial Intelligence*, pages 261–268.

Preuss, M. (2015). *Multimodal Optimization by Means of Evolutionary Algorithms*. Natural Computing Series. Springer International Publishing, 1st edition.

Qian, C., Tang, K., and Zhou, Z.-H. (2016). Selection Hyper-heuristics Can Provably Be Helpful in Evolutionary Multi-objective Optimization. In *Parallel Problem Solving from Nature – PPSN XIV*, pages 835–846. Springer International Publishing.

Qian, C., Yu, Y., and Zhou, Z.-H. (2013). An analysis on recombination in multi-objective evolutionary optimization. *Artificial Intelligence*, 204:99–119.

Qu, B. Y. and Suganthan, P. N. (2010). Novel multimodal problems and differential evolution with ensemble of restricted tournament selection. In *IEEE Congress on Evolutionary Computation*, pages 1–7.

Rai, D. and Tyagi, K. (2013). Bio-inspired Optimization Techniques: A Critical Comparative Study. *SIGSOFT Softw. Eng. Notes*, 38(4):1–7.

Rohlfshagen, P., Lehre, P. K., and Yao, X. (2009). Dynamic Evolutionary Optimisation: An Analysis of Frequency and Magnitude of Change. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '09, pages 1713–1720. ACM.

Rowe, J. E. and Sudholt, D. (2014). The choice of the offspring population size in the $(1, \lambda)$ evolutionary algorithm. *Theoretical Computer Science*, 545:20–38.

Sareni, B. and Krahenbuhl, L. (1998). Fitness sharing and niching methods revisited. *IEEE Transactions on Evolutionary Computation*, 2(3):97–106.

Scharnow, J., Tinnefeld, K., and Wegener, I. (2005). The analysis of evolutionary algorithms on sorting and shortest paths problems. *Journal of Mathematical Modelling and Algorithms*, 3(4):349–366.

Schwefel, H.-P. and Rudolph, G. (1995). Contemporary evolution strategies. In *Advances in Artificial Life*, pages 891–907. Springer Berlin Heidelberg.

Shir, O. M. (2012). Niching in Evolutionary Algorithms. In *Handbook of Natural Computing*, pages 1035–1069. Springer Berlin Heidelberg.

Singh, G. and Deb, K. (2006). Comparison of Multi-modal Optimization Algorithms Based on Evolutionary Algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '06, pages 1305–1312. ACM.

Smith, R. E., Forrest, S., and Perelson, A. S. (1993). Searching for Diverse, Cooperative Populations with Genetic Algorithms. *Evolutionary Computation*, 1(2):127–149.

Squillero, G. and Tonda, A. (2016). Divergence of character and premature convergence: A survey of methodologies for promoting diversity in evolutionary optimization. *Information Sciences*, 329:782–799.

Storch, T. and Wegener, I. (2004). Real royal road functions for constant population size. *Theoretical Computer Science*, 320(1):123 – 134.

Storn, R. and Price, K. (1997). Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *Journal of Global Optimization*, 11(4):341–359.

Stützle, T. G. (1999). *Local search algorithms for combinatorial problems - analysis, improvements, and new applications*, volume 220 of *DISKI. Dissertationen zur Künstlichen Intelligenz [Dissertations on Artificial Intelligence]*. Infix, Sankt Augustin.

Sudholt, D. (2005). Crossover is Provably Essential for the Ising Model on Trees. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '05, pages 1161–1167. ACM.

Sudholt, D. (2008). *Computational Complexity of Evolutionary Algorithms, Hybridizations, and Swarm Intelligence*. PhD thesis, Technische Universität Dortmund.

Sudholt, D. (2009). The impact of parametrization in memetic evolutionary algorithms. *Theoretical Computer Science*, 410(26):2511–2528.

Sudholt, D. (2011a). Hybridizing Evolutionary Algorithms with Variable-Depth Search to Overcome Local Optima. *Algorithmica*, 59(3):343–368.

Sudholt, D. (2011b). Using Markov-chain Mixing Time Estimates for the Analysis of Ant Colony Optimization. In *Proceedings of the 11th Workshop Proceedings on Foundations of Genetic Algorithms*, FOGA '11, pages 139–150. ACM.

Sudholt, D. (2012). Crossover Speeds Up Building-block Assembly. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '12, pages 689–702. ACM.

Sudholt, D. (2015). Parallel Evolutionary Algorithms. In *Springer Handbook of Computational Intelligence*, pages 929–959. Springer Berlin Heidelberg.

Sudholt, D. (2017). How Crossover Speeds up Building Block Assembly in Genetic Algorithms. *Evolutionary Computation*, 25(2):237–274.

Sudholt, D. (2018). The Benefits of Population Diversity in Evolutionary Algorithms: A Survey of Rigorous Runtime Analyses. *ArXiv e-prints*. http://arxiv.org/abs/1801.10087.

Sudholt, D. and Witt, C. (2016). Update Strength in EDAs and ACO: How to Avoid Genetic Drift. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '16, pages 61–68. ACM.

Talbi, E.-G. (2002). A Taxonomy of Hybrid Metaheuristics. *Journal of Heuristics*, 8(5):541–564.

Talbi, E.-G. (2009). *Metaheuristics: From Design to Implementation*, volume 74 of *Wiley Series on Parallel and Distributed Computing*. John Wiley & Sons.

Talbi, E.-G. and Bachelet, V. (2006). COSEARCH: A Parallel Cooperative Metaheuristic. *Journal of Mathematical Modelling and Algorithms*, 5(1):5–22.

Tsafarakis, S. (2016). Redesigning product lines in a period of economic crisis: a hybrid simulated annealing algorithm with crossover. *Annals of Operations Research*, 247(2):617–633.

Turing, A. M. (1948). Intelligent machinery. Technical report, National Physical Laboratory (NPL).

Ursem, R. K. (2002). Diversity-Guided Evolutionary Algorithms. In *Parallel Problem Solving from Nature – PPSN VII*, pages 462–471. Springer Berlin Heidelberg.

Watson, R. A. and Jansen, T. (2007). A Building-block Royal Road Where Crossover is Provably Essential. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '07, pages 1452–1459. ACM.

Wegener, I. (2002). Methods for the Analysis of Evolutionary Algorithms on Pseudo-Boolean Functions. In *Evolutionary Optimization*, pages 349–369. Springer US.

Wegener, I. and Witt, C. (2005). On the Optimization of Monotone Polynomials by Simple Randomized Search Heuristics. *Combinatorics, Probability and Computing*, 14(1-2):225–247.

Witt, C. (2006). Runtime Analysis of the (μ+1) EA on Simple Pseudo-Boolean Functions. *Evolutionary Computation*, 14(1):65–86.

Wu, J., Polyakovskiy, S., Wagner, M., and Neumann, F. (2018). Evolutionary Computation Plus Dynamic Programming for the Bi-objective Travelling Thief Problem. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '18, pages 777–784. ACM.

Yagiura, M. and Ibaraki, T. (2001). On metaheuristic algorithms for combinatorial optimization problems. *Systems and Computers in Japan*, 32(3):33–55.

Yang, X. (2010). *Engineering Optimization: An Introduction with Metaheuristic Applications*. John Wiley & Sons, 1 edition.

Yin, X. and Germay, N. (1993). A Fast Genetic Algorithm with Sharing Scheme Using Cluster Analysis Methods in Multimodal Function Optimization. In *Artificial Neural Nets and Genetic Algorithms*, pages 450–457. Springer Vienna.

Yu, E. and Suganthan, P. (2010). Ensemble of niching algorithms. *Information Sciences*, 180(15):2815–2833.

Yu, X. and Gen, M. (2012). *Introduction to Evolutionary Algorithms*. Springer Publishing Company, Incorporated, 1st edition.

Zitzler, E. and Künzli, S. (2004). Indicator-Based Selection in Multiobjective Search. In *Parallel Problem Solving from Nature – PPSN VIII*, pages 832–842. Springer Berlin Heidelberg.

Zitzler, E., Laumanns, M., and Thiele, L. (2001). SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In *Proceedings of the EUROGEN '2001 Conference*, pages 95–100. International Center for Numerical Methods in Engineering.

Appendix A

Mathematical Background

This section is devoted to some mathematical material that is used throughout this report. We start by reviewing box and whiskers plots. Then we give a review of basic ideas from probability theory from the following main sources: Motwani and Raghavan (1995), Mitzenmacher and Upfal (2005) and Doerr (2018).

A.1 Box and Whiskers Plots (Box Plots)

Box and whisker plots are a form of representing statistical data. They provide the median together with upper and lower quartiles as well as the minimum and maximum value of the data. We use an extended variant where additionally outliers are identified. Box plots are non-parametric: they display variation in samples of a statistical population without making any assumptions of the underlying statistical distribution. The spacings between the different parts of the box indicate the degree of dispersion (spread) and skewness in the data, and show outliers.

An example for a box and whiskers plot as used in this thesis is depicted in Figure A.1. The lower and upper (from left to right) quartiles form a rectangle (box). The bottom and top of the box plot are always the first and third quartiles (Q1 and Q3, respectively), and the band inside the box is the second quartile or the median (M). Two lines attached to the box indicate the extreme values of the data (whiskers). The length of these whiskers is at most 1.5 times the so-called interquartile range (IQR), which is defined as the difference of the upper and lower quartile. More extreme points are considered outliers and drawn as circles.

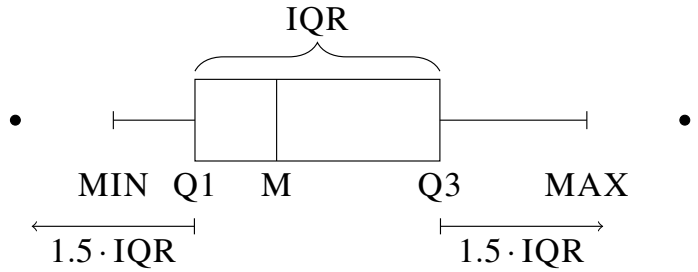


Figure A.1: Visualisation of a box and whisker plot.

A.2 Probability Theory

This section contains some formal mathematical settings for analysing the randomized algorithm. All statements, some proofs can be found in Mitzenmacher and Upfal (2005).

A.2.1 Axioms of Probability

Any probabilistic statement must refer to the underlying probability space.

Definition A.1 (Probability Space)**.** *A probability space has three components:*

1. *a sample space* Ω*, which is the set of all possible outcomes of the random process modelled by the probability space;*
2. *a family of sets* $\mathscr{F}$ *representing the allowable events, where each set in* $\mathscr{F}$ *is a subset of the sample space* Ω*; and*
3. *a probability function* $\text{Prob} : \mathscr{F} \to \mathbb{R}$ *satisfying Definition A.2.*

An element of Ω *is called a simple or elementary event.*

Definition A.2 (Probability Function)**.** *A probability function is any function* $\text{Prob} : \mathscr{F} \to \mathbb{R}$ *that satisfies the following conditions:*

1. *for any event* E*,* $0 \leq \text{Prob}(E) \leq 1$*;*
2. $\text{Prob}(\Omega) = 1$*; and*
3. *for any finite or countable infinite sequence of pairwise mutually disjoint (no element in common or they cannot occur at the same time, another word that means mutually disjoint is exclusive) events* $E_1, E_2, E_3, \ldots,$

$$\text{Prob}\left(\bigcup_{i\geq 1} E_i\right) = \sum_{i\geq 1} \text{Prob}(E_i).$$

Lemma A.3 (Union Bound)**.** *For any finite or countably infinite sequence of events* $E_1, E_2, \ldots,$

$$\text{Prob}\left(\bigcup_{i\geq 1} E_i\right) \leq \sum_{i\geq 1} \text{Prob}(E_i).$$

Notice that Lemma A.3 differs from the third part of Definition A.2 in that Definition A.2 is an equality and requires the events to be pairwise mutually disjoint.

Definition A.4 (Independent Events)**.** *Two events* E *and* F *are independent (the occurrence of one does not affect the probability of the other) if and only if*

$$\text{Prob}(E \cap F) = \text{Prob}(E) \cdot \text{Prob}(F)$$

More generally, events $E_i, E_2, \ldots, E_k$ *are mutually independent if and only if, for any subset* $I \subseteq [1,k]$*,*

$$\text{Prob}\left(\bigcap_{i\in I} E_i\right) = \prod_{i\in I} \text{Prob}(E_i).$$

Definition A.5 (Conditional Probability)**.** *The conditional probability that event* E *occurs given that event* F *occurs is*

$$\text{Prob}(E \mid F) = \frac{\text{Prob}(E \cap F)}{\text{Prob}(F)}.$$

The conditional probability is well-defined only if $\text{Prob}(F) > 0$*.*

Intuitively, we are looking for the probability of $E \cap F$ within the set of events defined by F. Because F defines our restricted sample space, we normalize the probabilities by dividing by $\text{Prob}(F)$, so that the sum of the probabilities of all events is 1. When $\text{Prob}(F) > 0$, the definition can also be written in the useful form

$$\text{Prob}(E \mid F)\text{Prob}(F) = \text{Prob}(E \cap F).$$

Notice that, when E and F are independent and $\text{Prob}(F) \neq 0$, we have

$$\text{Prob}(E \mid F) = \frac{\text{Prob}(E \cap F)}{\text{Prob}(F)} = \frac{\text{Prob}(E)\text{Prob}(F)}{\text{Prob}(F)} = \text{Prob}(E).$$

This is a property that conditional probability should have; intuitively, if two events are independent, then information about one event should not affect the probability of the second event.

Theorem A.6 (Law of Total Probability)**.** *Let $E_1, E_2, \ldots, E_n$ be mutually disjoint events in the sample space Ω, and let $\bigcup_{i=1}^{n} E_i = \Omega$. Then*

$$\text{Prob}(B) = \sum_{i=1}^{n} \text{Prob}(B \cap E_i) = \sum_{i=1}^{n} \text{Prob}(B \mid E_i)\text{Prob}(E_i).$$

A.2.2 Random Variables and Expectation

When studying a random event, we are often interested in some value associated with the random event rather than in the event itself. For example, in tossing two dice we are often interested in the sum of the two dice rather than the separate value of each die.

Definition A.7 (Random Variable)**.** *A random variable X on a sample space Ω is a real-value function on Ω; that is, $X : \Omega \to \mathbb{R}$. A discrete random variable is a random variable that takes on only a finite or countably infinite number of values*

For a discrete random variable X and a real value a, the event "$X = a$" includes all the basic events of the sample space in which the random variable X assumes the value of a. That is, "$X = a$" represents the set $\{s \in \Omega \mid X(s) = a\}$. We denote the probability of that event by

$$\text{Prob}(X = a) = \sum_{s \in \Omega : X(s) = a} \text{Prob}(s).$$

Definition A.8 (Expectation of a Discrete Random Variable)**.** *The expectation of a discrete random variable X, denoted by $\mathrm{E}[X]$, is given by*

$$\mathrm{E}[X] = \sum_{i}^{\infty} \text{Prob}(X \geq i),$$

if X takes values in $(-\infty, 0] \cup \mathbb{N}$, then $\mathrm{E}[X] \leq \sum_{i=1}^{\infty} \text{Prob}(X \geq i)$ still holds.

Theorem A.9 (Linearity of Expectations)**.** *For any finite collection of discrete random variables $X_1, X_2, \ldots, X_n$ with finite expectations.*

$$\mathrm{E}\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} \mathrm{E}[X_i].$$

Definition A.10 (Binomial Random Variable)**.** *A binomial random variable X with parameters n and p, denoted by* $\mathrm{Bin}(n,p)$*, is defined by the following probability distribution on* $j = 0,1,2,\ldots,n$*:*

$$\mathrm{Prob}(X = j) = \binom{n}{j} p^j (1-p)^{n-j}.$$

That is, the binomial random variable X equals j when there are exactly j successes and $n-j$ *failures in n independent experiments, each of which is successful with probability p. This independent experiment need to satisfy these four conditions:*

- *A fixed number of trials.*
- *Each trial is independent of the others.*
- *There are only two outcomes.*
- *The probability of each outcome remains constant from trial to trial.*

Definition A.11 (Expectation of a Binomial Random Variable)**.** *Let X be a binomial random variable describing n trials with success probability p. Then* $\mathrm{E}[X] = pn$*.*

Theorem A.12 (Law of Total Expectation)**.**

$$\mathrm{E}[Y] = \mathrm{E}[\mathrm{E}[Y \mid Z]]$$

Definition A.13 (Geometric Random Variable)**.** *A geometric random variable X with parameter p is given by the following probability distribution on* $n = 1,2,\ldots$*:*

$$\mathrm{Prob}(X = n) = (1-p)^{n-1} p.$$

That is, for the geometric random variable X to equal n, there must be $n-1$ *failures, following by a success (counts the number of attempts needed to obtain the first success).*

Definition A.14 (Expectation of a Geometric Random Variable)**.** *Let X be a geometric random variable with success probability p. Then* $\mathrm{E}[X] = 1/p$*.*

A.2.3 Chernoff Bounds

Lemma A.15 (Chernoff Bounds)**.** *Let* $X = X_1 + \ldots + X_n$ *be the sum of independent random variable with* $X_i \in \{0,1\}$ *for all* $1 \leq i \leq n$*. Then*

$$\begin{aligned}
&\text{Prob}(X \geq (1+\delta)\text{E}[X]) < \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\text{E}[X]} && \text{for} \quad \delta > 0 \\
&\text{Prob}(X \leq (1-\delta)\text{E}[X]) \leq \left(\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\right)^{\text{E}[X]} && \text{for} \quad 0 < \delta < 1 \\
&\text{Prob}(X \geq (1+\delta)\text{E}[X]) \leq e^{-\text{E}[X]\delta^2/3} && \text{for} \quad 0 < \delta < 1 \\
&\text{Prob}(X \leq (1-\delta)\text{E}[X]) \leq e^{-\text{E}[X]\delta^2/2} && \text{for} \quad 0 < \delta < 1
\end{aligned}$$

A.3 Useful Combinatorial Inequalities

Lemma A.16. *For all $n \in \mathbb{N}$,*

$$\left(1-\frac{1}{n}\right)^n \leq \frac{1}{e} \leq \left(1-\frac{1}{n}\right)^{n-1}.$$

Lemma A.17 (Harmonic Number)**.** *For any $n \in \mathbb{N}$,the nth Harmonic number H_n is defined as follows:*

$$\text{H}_n = 1 + \frac{1}{2} + \cdots + \frac{1}{n} = \sum_{i=1}^{n} \frac{1}{i}.$$

For any $n \in \mathbb{N}$, the nth Harmonic number is

$$\text{H}_n = \ln n + \Theta(1).$$

Lemma A.18 (Stirling's approximation)**.** *For $n \in \mathbb{N}$,*

$$\sqrt{2\pi n} \cdot \left(\frac{n}{e}\right)^n \leq n! \leq \sqrt{2\pi n} \cdot \left(\frac{n}{e}\right)^n \cdot e^{1/(12n)}.$$

In particular,

$$n! \geq \left(\frac{n}{e}\right)^n.$$

The following estimations are well known; the last inequality follows from Stirling's approximation.

Lemma A.19 (Binomial coefficients)**.** *For $k, n \in \mathbb{N}$ with $k \leq n$,*

$$\frac{n^k}{k^k} \leq \binom{n}{k} \leq \frac{n^k}{k!} \leq \left(\frac{ne}{k}\right)^k.$$