Designing the Real-Valued Gene-pool Optimal Mixing Evolutionary Algorithm and Applying it to Substantially Improve the Efficiency of Multi-Objective Deformable Image Registration

Anton Bouter

M.Sc. Thesis

Designing the Real-Valued Gene-pool Optimal Mixing Evolutionary Algorithm and Applying it to Substantially Improve the Efficiency of Multi-Objective Deformable Image Registration

by

Anton Bouter

to obtain the degree of Master of Science
in Computer Science, within the field of Software Technology, specialized in Algorithmics,
at the Delft University of Technology,
to be defended publicly on Tuesday October 4th, 2016 at 1:00 PM.

Student number:	4167562		
Thesis committee:	Prof. Dr. C. Witteveen,	TU Delft,	Chair
	Dr. P. A. N. Bosman,	CWI,	Daily supervisor
	Dr. Ir. R. F. Remis,	TU Delft,	Committee member

Delft University of Technology
Faculty Electrical Engineering, Mathematics and Computer Science
Mekelweg 5
Delft, The Netherlands

Centrum Wiskunde & Informatica
Life Sciences Group
Science Park 123
Amsterdam, The Netherlands

Abstract

The recently introduced Gene-pool Optimal Mixing Evolutionary Algorithm (GOMEA) for discrete variables has been shown to be able to efficiently and effectively exploit the decomposability of optimization problems, especially in a grey-box setting, in which a solution can be efficiently updated after a modification of a subset of its variables. GOMEA is considered to be state of the art, but currently no version of GOMEA for real-valued variables exists. In this thesis, we design a real-valued version of GOMEA, for both single-objective and multi-objective optimization. Our novel GOMEA variant is then applied to the Deformable Image Registration (DIR) problem, which was adapted to allow for efficient partial evaluations. DIR concerns the calculation of a deformation that transforms one image to another, and is of great importance for many medical applications. Experiments are performed to assess GOMEA's performance in black-box and grey-box settings on a range of single-objective and multi-objective benchmark problems, including comparisons with it to the state-of-the-art real-valued optimization algorithm AMaLGaM. From the results of these experiments, we find that GOMEA performs substantially better on all considered single-objective and multi-objective benchmark problems in a grey-box setting, in terms of required time and number of evaluations. Moreover, the improvement becomes larger as problem dimensionality increases. In a black-box setting, GOMEA still performed better than AMaLGaM in terms of time, and comparable in terms of the number of evaluations. On DIR problems, GOMEA achieved solutions of similar quality while achieving a speed-up of up to a factor of 1600.

Preface

After five years of learning about many different aspects of Computer Science, and about many things that science in general has to offer, I feel like I have finally made my first tiny contribution to the scientific community. This contribution is this document describing my Master's Thesis on *Designing the Real-Valued Gene-pool Optimal Mixing Evolutionary Algorithm and Applying it to Substantially Improve the Efficiency of Multi-Objective Deformable Image Registration.* This topic regards the algorithmic design of an evolutionary algorithm, which is then applied to medical imaging. Results presented in this thesis show that the designed algorithm, the real-valued Gene-pool Optimal Mixing Evolutionary Algorithm (GOMEA), achieves excellent performance on artificial benchmark problems. Additionally, real-valued GOMEA greatly improves the efficiency of multi-objective deformable image registration applied to medical images, taking a major step towards a practical application. I am glad to have been allowed to work on this project as a trainee at the Centrum Wiskunde & Informatica (CWI) in Amsterdam as a part of the Life Sciences group under supervision of senior researcher Dr. Peter Bosman.

Finally, I would like to express my gratitude to all people who have supported and helped me throughout my project. In particular, I would like to thank my supervisor Dr. Peter Bosman for his extensive feedback and his incredible effort as the primary supervisor of my project. Secondly, I would like to thank Prof. Dr. Cees Witteveen for his general guidance throughout this project, and Dr. Tanja Alderliesten for her feedback on my submission for the SPIE medical imaging conference. A final word of thanks goes out to all my colleagues for their comments and all our fruitful discussions.

Anton Bouter
Amsterdam, The Netherlands
September 27, 2016

Contents

1 Introduction

With the continuing advancement of technology, more and more tasks are assigned to machines. Some of these tasks might be impossible for humans, otherwise very tedious, or time consuming. Using powerful computers, many of such problems can be tackled. However, even with the most powerful computers, many difficult problems would not have been able to be solved without the use of efficient algorithms. Additionally, improving the efficiency of an algorithm can have a much larger impact, possibly of several orders of magnitude, than a technical upgrade. For this reason, the development of efficient algorithms is a very important field of research.

Evolutionary Algorithms (EAs) are frequently used in engineering, because they can be applied to a wide range of (real-world) problems. The broad applicability of EAs is caused by the fact that they only require potential solutions to be encoded in a way that enables the EA to quantify the quality of any potential solution. How the algorithm then attempts to find the optimal solution is completely problem independent. In this scenario, we speak about Black-Box Optimization (BBO), because the EA virtually has zero knowledge about the problem that is being optimized, apart from the so-called objective function that defines the quality of any potential solution. Many state-of-the-art EAs are aimed at BBO, because this retains their broad applicability. However, in certain scenarios the structure of the optimization problem is known and can be exploited to enhance the performance of the EA. We focus in particular on Grey-Box Optimization (GBO), which allows so-called partial evaluations. Through partial evaluations the quality of a solution can correctly be recalculated after it is partially modified. Because such a partial evaluation only recalculates the objective value using only a fraction of a solution, it can often be performed more efficiently than the complete evaluation of a solution. An algorithm that exploits the benefits of GBO is the recently introduced Gene-pool Optimal Mixing Evolutionary Algorithm (GOMEA) [11, 42], which has been successfully applied to a range of discrete optimization problems and can be considered state of the art in this field. However, no evolutionary algorithm is known to exploit the GBO setting in the field of real-valued optimization. State-of-the-art EAs for real-valued optimization, such as CMA-ES [20, 21] and AMaLGaM [12], are generally aimed at BBO settings, which, in a GBO setting, makes them potentially slower than an algorithm tuned for GBO. Considering GOMEA's excellent performance on discrete optimization problems, in this thesis we design an adaptation of GOMEA that can be used to solve real-valued optimization problems. This adaptation is named the Real-Valued Gene-pool Optimal Mixing Evolutionary Algorithm. Furthermore, we evaluate the performance of real-valued GOMEA in both a GBO and BBO setting on a collection of well-known single-objective and multi-objective benchmark problems.

To test the applicability of real-valued GOMEA to a real-world problem, we consider the Deformable Image Registration (DIR) problem for medical images, which is a prime example of a problem where partial evaluations can be useful. Deformable image registration involves a source and a target image, and its goal is to find a deformation, which, when applied to the source image, would lead to an outcome that resembles the target image as closely as possible. However, very strong deformations, which are not physically plausible, can sometimes lead to a larger resemblance between the deformed source image and the target image, because of issues such as noise in the images. In other words, the deformation then starts to overfit the resemblance on noise. To suppress the probability of such unnatural deformations, the smoothness of the deformation must also be taken into account.

Applications of deformable image registration include, but are not limited to, applications in the field of radiotherapy [43]. For example, deformable image registration can be applied to update a treatment plan to align it with a different scan [16]. This can be useful for scans acquired in different positions, at different times, or with different modalities, such as CT or MRI.

Current state-of-the-art methods for deformable image registration, such as Elastix [25], assign weights to the objectives of similarity and smoothness, indicating their relative importance. The weighted combination of these objectives is then used in a single-objective optimization approach. These weights, however, need to be tuned manually for each problem type and instance, which is time-consuming and not very insightful. Therefore, a multi-objective approach that keeps individual objectives separated during optimization has recently gained attention, because this approach removes the requirement of manually tuning weights of the problem's objectives [2, 3].

The currently used algorithm for the multi-objective deformable image registration problem is iMAMaLGaM [37], which is capable of obtaining excellent results [4]. However, due to its very general approach that does not include any problem-specific enhancements, it is arguably too slow for practical use. Real-valued GOMEA is instead capable of exploiting partial evaluations, potentially resulting in enhanced performance that could cause multi-objective deformable image registration to be efficient enough to be used in real-world clinical practice.

This thesis is structured in the following way. The first three chapters describe the design of real-valued GOMEA for different types of problems and provide relevant background information. Specifically, first we describe the process of designing the real-valued GOMEA for solving single-objective optimization problems. We then describe the extension of this algorithm to solve multi-objective optimization problems. Subsequently, deformable image registration is introduced and problem specific adaptations of real-valued GOMEA for solving this problem in a multi-objective formulation are described. We then present the results of performing experiments with all developed versions of real-valued GOMEA aimed at determining the performance of our newly developed algorithm. Finally, we discuss the results from the experiments, draw conclusions based on this, and introduce potential topics for future work.

2

Designing the Single-Objective Real-Valued GOMEA

After introducing more detailed background information on single-objective optimization and EAs in general, the design of the real-valued GOMEA for single-objective optimization functions is discussed in detail in this section.

2.1. Background

This section starts with a general introduction to single-objective optimization, followed by a number of sections describing related work in the field of EAs, and the current state-of-the-art algorithms.

2.1.1. Optimization

An optimization problem is defined as a problem where the goal is to find the best possible solution(s) from a certain set of available solutions. Any solution $\boldsymbol{x}$ can often be represented as a vector of length ℓ where every component $\boldsymbol{x}_i$ of $\boldsymbol{x}$ is called a variable. Depending on the problem, each variable can be assigned values from a certain domain (e.g. binary, integer, real-valued). The joint domain that describes every possible combination of variables is called the search space or parameter space, because this is the domain in which we search for solutions to the optimization problem. For example, on a problem with three real-valued variables the search space would be defined as the three-dimensional continuous space $\mathbb{R}^3$. Optimization problems exist where different variables have different domains, but this work will be restricted to problems where every variable has the same domain. The quality of a solution is defined by the optimization function of the problem, denoted $f(\boldsymbol{x})$, which can consist of multiple objective functions $f_i(\boldsymbol{x})$. Each objective function calculates an objective value of a solution, which we assume to be a real-valued number. If a problem has a single objective function, it is known as a single-objective problem. Such problems are discussed in the current section, whereas problems having more than one objective function are discussed in Section 3.1.1. The optimization function basically defines the problem, because it defines the ranking of every possible solution to the problem. Each objective function $f_i(\boldsymbol{x})$ should either be minimized or maximized, but without loss of generality we assume minimization for the rest of this work.

The range of possible objective values constitutes the objective space of a problem. The optimization function essentially maps a solution from the parameter space to the objective space. For example, for a problem with three variables and one objective function, the domain of the objective space is $\mathbb{R}$, and the optimization function would map the position of a solution in the parameter space $\mathbb{R}^3$ to a position in the objective space $\mathbb{R}$, i.e., $f : \mathbb{R}^3 \mapsto \mathbb{R}$. A number of solutions could have very different values for each of their variables, but have a very similar objective value, meaning that these solutions would be far apart in the parameter space, but close to each other in the objective space.

The calculation of the objective values of a single solution is called an evaluation. Often the total number of evaluations required to reach a certain value is used as a measure of performance of an algorithm, because this measure is independent of the CPU speed of the machine the experiments were performed on. However, required time should also be measured, because this also includes overhead by the algorithm. This can be substantial, especially when evaluations can be performed relatively efficiently. The way in which evaluations are performed depends on the assumed information about the optimization problem. When an algorithm does not have any information about the optimization problem, we speak about a Black-Box Optimization (BBO) setting, because the evaluation function is essentially a black box. In a BBO setting, the objective values of a solution can only be calculated by feeding the solution to the evaluation function. On the opposite side of the spectrum is White-Box Optimization (WBO). This setting allows an algorithm to exploit any problem-specific features to improve performance. WBO is mostly used by algorithms specifically tuned for certain problems, while BBO is used by algorithms that use a more general approach in order to be able to solve a wide range of optimization problems. An additional setting, which we call Grey-Box Optimization (GBO), allows partial evaluations to be performed. A partial evaluation is defined as the efficient re-evaluation of a solution after the modification of only a subset of variables. This is done by evaluating the influence of the modified variables on the total objective value. For example, when the objective function is the sum of all variables, a partial evaluation can be performed after the modification of a single variable by subtracting this variable's previous value and adding its new value. This results in the correct computation of the total objective value without the requirement to access and process the value of each variable. When a GBO setting is used, a partial evaluation is only counted as a fraction of an evaluation, based on the number of variables that are re-evaluated. A partial evaluation that requires access of k variables in order to calculate the full objective value is registered as a fraction k/ℓ of an evaluation. Often k is equal to the number of modified variables, but it can be higher when a problem contains dependent variables.

A solution with which the best possible objective objective value is associated is called a global optimum. This solution is often denoted as $\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} f(\boldsymbol{x})$ and its objective value as $f(\boldsymbol{x}^*) = \min_{\boldsymbol{x}} f(\boldsymbol{x})$. Additionally, many problems have local optima, which are loosely defined as solutions from which the exploration of the nearby search space will not lead to an improved objective value. The notion of nearby search space is subjective to the algorithm that is being used, because different algorithms can traverse this space in a very different way. This means that two solutions could be considered nearby for one algorithm, while they are very far apart for a different algorithm.

Consider for example the deceptive trap function [1, 17], which, in case of minimization, is defined as follows for some solution $\boldsymbol{x} \in \{0,1\}^{\ell}$:

$$f_{\text{trap}}(\boldsymbol{x}) = \begin{cases} \frac{\#\text{ones}+1}{\ell} & \text{if } \#\text{ones } < \ell \\ 0 & \text{if } \#\text{ones} = \ell, \end{cases}$$

where #ones denotes the total number of ones in $\boldsymbol{x}$. The global optimum is $f(\boldsymbol{x}^*) = 0$ for $\boldsymbol{x}^* = \{1\}^{\ell}$. The name deceptive trap function comes from the fact that the objective value improves as the number of ones decreases, apart from the global optimum where the number of ones is maximized. For most algorithms that are exploring the search space, decreasing the number of ones will seem beneficial, because in most cases this will improve the objective value. However, this strategy will lead to the solution $\boldsymbol{x} = \{0\}^{\ell}$ with $f(\boldsymbol{x}) = \frac{1}{\ell}$. In general this state is considered a local optimum, because improving the objective value beyond this state requires the inversion of all ℓ variables.

2.1.2. Evolutionary Algorithms

Loosely defined, Evolutionary Algorithms (EAs) are optimization algorithms that are based on the principles of natural evolution. An EA operates on a set of so-called individuals that constitute a population $\boldsymbol{P}$, where each individual encodes a candidate solution to the optimization problem in question. For instance, in the subclass of genetic algorithms, each individual $\boldsymbol{x}$ is often a vector of binary variables $\boldsymbol{x} = \{0,1\}^{\ell}$.

The process of an EA then mainly consists of two mechanisms: selection and variation. The purpose of the variation mechanism is to generate new solutions by combining information available in all solutions that are currently in the population. The goal of the selection mechanism is then to guide the algorithm towards the optimum by selecting the highest-quality solutions and removing lower-quality solutions. Variation often consists of either mutation, recombination, or a combination of both. Both mutation and recombination generate offspring based on the existing population. Mutation alters a small selection of variables of an individual, creating an offspring solution that is very similar to an existing individual in the population. Recombination selects a number of parent individuals and a new offspring solution is then created by combining information represented by the parent solutions. These variation mechanisms are applied to the population to generate offspring. From the offspring and the population ultimately a new population for the next generation is selected.

The procedure governing a single generation in an EA typically starts off by selecting the best individuals, which will function as parent solutions for the variation processes. After performing a sufficient number of variation operations, the offspring solutions are evaluated and a selection of individuals in the population is replaced by a selection of solutions in the offspring. This constitutes the population of the following generation. New generations are iteratively created until one of the termination criteria is met, e.g., a solution of sufficiently high quality has been found, the variance of the average objective value has decreased below a certain threshold, or the algorithm has exceeded a limit on the time or number of evaluations.

Genetic Algorithms

The Genetic Algorithm (GA) [24] is an instance of an EA that uses crossover as the main variation operator, which is an operator that creates an offspring solution by selecting the values of variables from a pair of parent solutions. During each generation the GA generates a set of n new individuals, the offspring set $\boldsymbol{O}$, by applying crossover to $\frac{n}{2}$ pairs of randomly selected parent solutions from the population. Although many variations exist, in what is commonly called the simple GA, a uniform crossover operation is used, which means that each variable of the offspring solution $\boldsymbol{x}$ is selected from either of the parent solutions with equal probability. In addition to $\boldsymbol{x}$, the solution $\boldsymbol{x}'$ that consists of all parent variables that were not selected by $\boldsymbol{x}$, is also added to the offspring. Therefore, each pair of parent solutions produces a pair of offspring solutions and the occurrence of each variable is preserved, because each variable ends up in either of the offspring solutions. Selection is then applied to the $2n$ solutions of the population and the offspring combined, selecting the n best solutions to be in the population of the following generation.

Optimal Mixing Evolutionary Algorithms

When crossover is performed by a simple GA, all variables of the offspring solution are randomly selected from either of the parents. More efficient ways of recombination were introduced through Estimation of Distribution Algorithms (EDAs) [29, 30]. EDAs estimate a distribution based on statistics computed from the population and adapt their recombination operation based on these statistics. The recombination operation of GA is also sensitive to noise, because an improvement that was caused by the change of one variable can be nullified by the simultaneous change of a different variable. It was even shown that, for certain problems, random crossover causes the required population size of the GA to scale exponentially with the problem size [41].

```
Function $GOMEA(\boldsymbol{x})$ :
    $fitness[\boldsymbol{x}^{elitist}] \leftarrow -\infty$
    for $i \in \{0,1,\ldots,n-1\}$ do
        $\boldsymbol{P}_i \leftarrow$ newly generated solution
        $EvaluateFitnessAndUpdateElitist(\boldsymbol{P}_i)$
    end
    $\boldsymbol{x}^{best}(0) \leftarrow arg\,\max_{\boldsymbol{x}\in\boldsymbol{P}}\{fitness[\boldsymbol{x}]\}$
    $t \leftarrow t^{NIS} \leftarrow 0$
    while not termination criterion satisfied do
        $\boldsymbol{F} \leftarrow$ FOS model learned from $\boldsymbol{P}$
        for $i \in \{0,1,\ldots,n-1\}$ do
            $\boldsymbol{O}_i \leftarrow GOM(\boldsymbol{P}_i)$
        end
        $\boldsymbol{P} \leftarrow \boldsymbol{O}$
        $\boldsymbol{x}^{best}(t) \leftarrow arg\,\max_{\boldsymbol{x}\in\boldsymbol{P}}\{fitness[\boldsymbol{x}]\}$
        if $fitness[\boldsymbol{x}^{best}(t)] > fitness[\boldsymbol{x}^{best}(t-1)]$ then
            $t^{NIS} \leftarrow 0$
        else
            $t^{NIS} \leftarrow t^{NIS} + 1$
        end
        $t \leftarrow t+1$
    end
end
```

Algorithm 1: Pseudo-code for GOMEA with a population of size n. The best solution at the end of generation t is denoted $\boldsymbol{x}^{best}(t)$. The best solution ever evaluated is denoted $\boldsymbol{x}^{elitist}(t)$.

```
Function EvaluateFitnessAndUpdateElitist($\boldsymbol{x}$) :
    $fitness[\boldsymbol{x}] \leftarrow f(\boldsymbol{x})$
    if $fitness[\boldsymbol{x}] \geq fitness[\boldsymbol{x}^{elitist}]$ then
        $fitness[\boldsymbol{x}^{elitist}] \leftarrow fitness[\boldsymbol{x}]$
    end
end
```

Algorithm 2: Evaluating an individual and updating the elitist solution.

Instead of applying recombination to all variables of a solution before re-evaluating the quality of this solution, the Optimal Mixing (OM) [42] procedure applies recombination to subsets of variables separately. Each recombination operation inserts a subset of variables from one solution, the so-called donor, into a different solution, the so-called parent. Additionally, the changes caused by a recombination operation are only accepted if these changes cause an improvement of the objective value of the parent. Such recombination operations can be performed in a successful way if the subsets of variables that are mixed simultaneously are based on a so-called dependency structure, which models dependencies between variables. Previous models include some that model pair-wise dependencies [33], and multiple levels of dependencies [32, 34].

In GOMEA, the structure of dependencies between variables is modeled as a family of subsets (FOS), denoted $\boldsymbol{F} = \{\boldsymbol{F}_1, \boldsymbol{F}_2, \dots, \boldsymbol{F}_{|\boldsymbol{F}|}\}$, where each $\boldsymbol{F}_i$ is a subset of the set $\{1, 2, \dots, \ell\}$ that contains an identifier for each variable. Each FOS element $\boldsymbol{F}_i \subseteq \{1, 2, \dots, \ell\}$ indicates that dependencies between each pair of variables in this set are considered in the mixing process. The most simple FOS structure is the univariate structure with $\boldsymbol{F}_i = \{i\}$ that contains only subsets of single variables. This structure does not model any dependencies between variables. A simple FOS structure that does model dependencies is the marginal product (MP) structure, which contains any number of non-overlapping subsets of variables, i.e., for all $\boldsymbol{F}_i, \boldsymbol{F}_j (i \neq j) \in \boldsymbol{F} : \boldsymbol{F}_i \cap \boldsymbol{F}_j = \emptyset$. Learning an MP structure can for instance be performed by a greedy algorithm that counts frequencies of substructures and minimizes the minimum description length metric, a procedure that is also used by ECGA [22]. An even richer FOS structure is given by the linkage tree structure. The linkage tree structure is capable of defining multiple levels of dependencies, because each variable can occur in more than one subset of the FOS. A linkage tree is built by initializing the leaves of the tree as the set of univariate elements. Subsequently, the internal nodes of the tree are generated by iteratively adding new nodes as the parent of the pair of nodes with the most highly dependent variables. The newly added parent node contains the union of the sets of variables contained by its children. This process continues until the root of the tree is generated, which contains the full set of problem variables. Deciding which two nodes should be merged in order to form a parent node is based on the mutual information between the pairs of variables described by these nodes. The pair of nodes with the maximum mutual information is always merged. Mutual information (MI) is a measure of the dependency between the values of two sets of random variables, which is defined as:

$$MI(\boldsymbol{X}, \boldsymbol{Y}) = H(\boldsymbol{X}) + H(\boldsymbol{Y}) - H(\boldsymbol{X}, \boldsymbol{Y}),$$

where $H(\boldsymbol{X})$ is the marginal entropy of $\boldsymbol{X}$ and $H(\boldsymbol{X}, \boldsymbol{Y})$ is the joint entropy of $\boldsymbol{X}$ and $\boldsymbol{Y}$. These entropies are calculated by counting occurrences of the values of variables in the population. If one certain value of one variable often occurs alongside a certain value of a different variable, the mutual information of this pair of variables will be high. Sets of variables with maximum mutual information are merged in the linkage tree, because the occurrences of values of these variables are seemingly dependent, possibly caused by a dependency between these problem variables. The linkage tree has been completed when no more merges are possible. The FOS structure of the linkage tree is then described by the complete set of nodes of the linkage tree. This structure is characterized by the fact that every FOS element $\boldsymbol{F}_i$ that consists of more than one variable is in fact the union of two non-intersecting FOS elements $\boldsymbol{F}_j$ and $\boldsymbol{F}_k$, i.e., $\boldsymbol{F}_i = \boldsymbol{F}_j \cup \boldsymbol{F}_k$ where $\boldsymbol{F}_j \cap \boldsymbol{F}_k = \emptyset$ and $|\boldsymbol{F}_i| > 1$.

The OM procedure is implemented by the Gene-pool Optimal Mixing Evolutionary Algorithm (GOMEA) [42], which can be considered to be state of the art in the field of metaheuristic discrete optimization. Pseudo-code of GOMEA's basic structure is displayed in Algorithm 1, which uses the function displayed in Algorithm 2 to evaluate an individual and update the elitist solution. In particular, GOMEA performs the so-called Gene-pool Optimal Mixing (GOM) procedure, for which pseudo-code is displayed in Algorithm 3. This procedure recombines an individual, a parent, by iterating over all FOS elements in a random order and applying recombination to the parent based on each of these FOS elements and a donor. A donor is randomly selected from the population, which acts as the gene-pool, for each recombination operation. The values of variables that are present in the specified FOS element are copied from the donor and inserted into the parent. Only if the modification of these variables leads to an improved objective value this crossover operation is accepted. Otherwise, the parent solution is returned to its previous state. This process is applied to each individual in the population.

The performance of GOMEA was later enhanced by the introduction of the forced improvement phase in GOM [11], which performs an additional round of recombination on individuals that were not changed during the previous generation or if no improvement occurred for multiple generations. This round of recombination uses the elitist solution, i.e., the solution with the best objective value found so far, as the donor. If any improvement is obtained at any moment during the forced improvement phase, this phase is immediately terminated. However, if an individual still has not improved after forced improvements were performed, the individual is completely replaced by the elitist solution. The forced improvements phase is included in the pseudo-code in Algorithm 3.

AMaLGaM

Extending the concept of EAs to the domain of real-valued variables is not straightforward. When a set of real-valued variables needs to be optimized, we cannot simply rely on copying the variables of existing solutions, because the search space is now continuous. Only using values of variables that were initially generated will therefore exclude an infinitely large part of the search space. Instead of recombining solutions, the Adapted Maximum-Likelikhood Gaussian Model Iterated Density-Estimation Evolutionary Algorithm (AMaLGaM-IDEA or AMaLGaM for short) [12] uses a normal probability distribution to generate new values of variables, and belongs to the state of the art in real-valued optimization.

The so-called full AMaLGaM uses an ℓ-dimensional multivariate normal probability distribution to model possible dependencies between each pair of variables. This normal distribution can be defined as $\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})$ where $\boldsymbol{\mu}$ is an ℓ-dimensional vector and $\boldsymbol{\Sigma}$ is an $\ell \times \ell$ matrix. The matrix $\boldsymbol{\Sigma}$ defines the covariances $\text{cov}(\boldsymbol{x}_i,\boldsymbol{x}_j)$ for all $i, j \in \{1,2,\ldots,\ell\}$, where the covariance between two variables $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is defined as $\text{cov}(\boldsymbol{x}_i,\boldsymbol{x}_j) = \text{E}\left[(\boldsymbol{x}_i - \text{E}[\boldsymbol{x}_i])(\boldsymbol{x}_j - \text{E}[\boldsymbol{x}_j])\right]$ with $\text{E}[\boldsymbol{x}_i]$ the expected value of $\boldsymbol{x}_i$. The covariance defines how two variables are related. A positive covariance means that a positive change of one variable is expected to be paired with a positive change in the other variable, while a negative covariance means that a positive change of one variable is expected to come with a negative change of the other variable. Samples can be drawn from the ℓ-variate normal distribution $\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})$ by first sampling from the normal distribution $\mathcal{N}(\boldsymbol{0},\boldsymbol{I})$ and then transforming the samples to corresponding samples drawn from $\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})$. This transformation is performed by multiplying by the lower triangular matrix $\boldsymbol{L}$ that is the result of the Cholesky decomposition $\boldsymbol{D} = \boldsymbol{L}\boldsymbol{L}^*$ of the covariance matrix and then adding the mean $\boldsymbol{\mu}$ to the samples drawn from $\mathcal{N}(\boldsymbol{0},\boldsymbol{I})$.

To make the sampling operation less computationally expensive, it is possible to factorize the covariance matrix and sample only subsets of variables from the multivariate distribution considering only these variables. This means that only dependencies between variables that are sampled simultaneously will be considered, making it increasingly difficult to solve problems involving a large number of dependencies. Therefore, it is important to correctly factorize the covariance matrix such that the correct dependencies between variables are captured. One approach to factorizing the covariance matrix is the univariate factorization. The univariate AMaLGaM estimates a univariate normal probability distribution for each variable separately, meaning that dependencies are not modeled, because the value of each variable is sampled separately from the values of any other variable. Such a normal distribution $\mathcal{N}(\mu,\sigma^2)$ of a single variable is defined by the mean μ and standard deviation σ of this variable. A different approach consists of computing a Bayesian factorization, for which a greedy approximation algorithm is often used [9, 27, 31], because the problem of finding a Bayesian factorization is NP-hard by itself. Empirical results showed, however, that the Bayesian factorization is outperformed by the univariate factorization when the problem contains a small number of dependencies, and it is outperformed by the full multivariate distribution when dealing with a large number of dependencies [14]. Therefore, computing a Bayesian factorization is usually not worth the effort, because a performance gain is only obtained on problems with a moderately small number of dependencies.

The basic outline of AMaLGaM, for which pseudo-code is displayed in Algorithm 4, can roughly be partitioned into three phases: selection, estimation, generation. After generating the initial population uniformly in a specified initialization range, a fraction τ of the best individuals in the population is selected as the selection set S. The selection set is then used to estimate a normal distribution from which the next

```
Function $GOM(\boldsymbol{x})$ :
    $\boldsymbol{b} \leftarrow \boldsymbol{o} \leftarrow \boldsymbol{x}$
    $fitness[\boldsymbol{b}] \leftarrow fitness[\boldsymbol{o}] \leftarrow fitness[\boldsymbol{x}]$
    $changed \leftarrow false$
    for $i \in \{0, 1, \ldots, |\boldsymbol{F}| - 1\}$ do
        $\boldsymbol{d} \leftarrow \textsc{Random}(\{\boldsymbol{P}_0, \boldsymbol{P}_1, \ldots, \boldsymbol{P}_{n-1}\})$
        $\boldsymbol{o}_{\boldsymbol{F}_i} \leftarrow \boldsymbol{d}_{\boldsymbol{F}_i}$
        if $\boldsymbol{o}_{\boldsymbol{F}_i} \neq \boldsymbol{b}_{\boldsymbol{F}_i}$ then
            $EvaluateFitnessAndUpdateElitist(\boldsymbol{o})$
            if $fitness[\boldsymbol{o}] \geq fitness[\boldsymbol{b}]$ then
                $\boldsymbol{b}_{\boldsymbol{F}_i} \leftarrow \boldsymbol{o}_{\boldsymbol{F}_i}$
                $fitness[\boldsymbol{b}] \leftarrow fitness[\boldsymbol{o}]$
                $changed \leftarrow true$
            else
                $\boldsymbol{o}_{\boldsymbol{F}_i} \leftarrow \boldsymbol{b}_{\boldsymbol{F}_i}$
                $fitness[\boldsymbol{o}] \leftarrow fitness[\boldsymbol{b}]$
            end
        end
    end
    if not $changed$ or $t^{NIS} > 1 + \lfloor {}^{10}\log(n) \rfloor$ then
        $changed \leftarrow false$
        for $i \in \{0, 1, \ldots, |\boldsymbol{F}| - 1\}$ do
            $\boldsymbol{o}_{\boldsymbol{F}_i} \leftarrow \boldsymbol{x}^{elitist}_{\boldsymbol{F}_i}$
            if $\boldsymbol{o}_{\boldsymbol{F}_i} \neq \boldsymbol{b}_{\boldsymbol{F}_i}$ then
                $fitness[\boldsymbol{o}] \leftarrow f(\boldsymbol{o})$
                if $fitness[\boldsymbol{o}] > fitness[\boldsymbol{b}]$ then
                    $\boldsymbol{b}_{\boldsymbol{F}_i} \leftarrow \boldsymbol{o}_{\boldsymbol{F}_i}$
                    $fitness[\boldsymbol{b}] \leftarrow fitness[\boldsymbol{o}]$
                    $changed \leftarrow true$
                else
                    $\boldsymbol{o}_{\boldsymbol{F}_i} \leftarrow \boldsymbol{b}_{\boldsymbol{F}_i}$
                    $fitness[\boldsymbol{o}] \leftarrow fitness[\boldsymbol{b}]$
                end
            end
        end
        if $changed$ then break
    end
    if not $changed$ then
        $\boldsymbol{o} \leftarrow \boldsymbol{x}^{elitist}$
        $fitness[\boldsymbol{o}] \leftarrow fitness[\boldsymbol{x}^{elitist}]$
    end
    return $\boldsymbol{o}$
end
```

Algorithm 3: The GOM procedure.

generation will be sampled. This distribution is estimated with maximum-likelihood from the selected set of solutions, which means that a normal distribution is estimated that is most probable to generate this set of selected solutions. This distribution is the multivariate normal distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ over ℓ variables where $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are respectively the vector of calculated means and the matrix of covariances of the selection, which are defined as follows:

$$\begin{aligned}\boldsymbol{\mu} &= \frac{1}{|S|}\sum_{\boldsymbol{x}\in S}\boldsymbol{x}\\ \boldsymbol{\Sigma} &= (\mathrm{cov}_{i,j}) \in \mathbb{R}^{\ell\times\ell}\\ \mathrm{cov}_{i,j} &= \frac{1}{|S|}\sum_{\boldsymbol{x}\in S}\left(\boldsymbol{x}_i-\boldsymbol{\mu}_i\right)\left(\boldsymbol{x}_j-\boldsymbol{\mu}_j\right)\end{aligned}$$

Additionally, the Adaptive Variance Scaling (AVS), Standard Deviation Ratio (SDR) and Anticipated Mean Shift (AMS) mechanisms are applied to the sampling procedure to prevent premature convergence. These mechanisms will be discussed later in this section. Following the sampling and evaluation of a new population, the next generation of the algorithm again starts with the selection procedure. This three-phase cycle continues until one of the termination criteria is met.

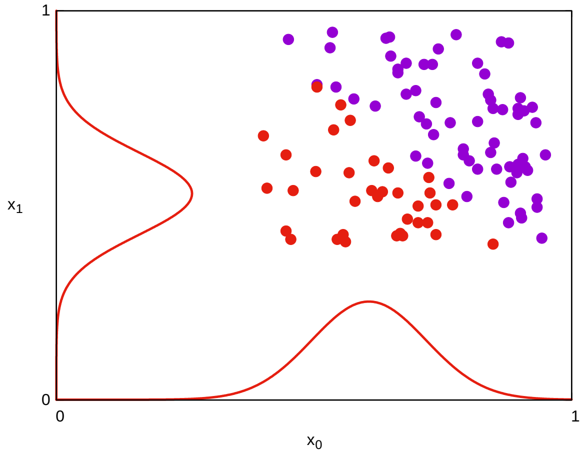


Figure 2.1: An example of the first generation of univariate AMaLGaM.

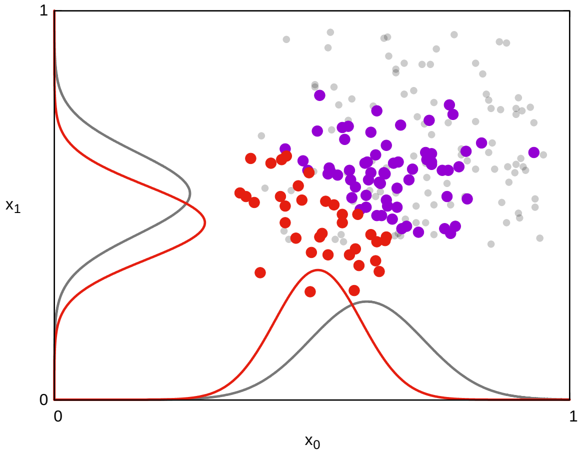


Figure 2.2: An example of the second generation of univariate AMaLGaM.

AVS

To illustrate the AVS mechanism, the first two generations of the univariate AMaLGaM applied to the two-dimensional problem with optimization function $f(\boldsymbol{x}) = \boldsymbol{x}_0 + \boldsymbol{x}_1$ are illustrated in Figures 2.1 and 2.2. Figure 2.1 displays a uniformly generated initial population with the selection marked in red and the estimated multivariate normal distribution projected onto each axis. Figure 2.2 shows the population in the next generation, with the population and distributions of the first generation displayed in grey. This new population was generated using the estimated normal distribution of the previous generation. Again the maximum-likelihood estimates of the selection are calculated and projected onto each axis. This normal distribution has higher peaks and therefore lower standard deviations, because the selection is less spread out than its originating population. If the initial population is generated far away from the optimum, repeated selection will cause the standard deviations to decrease too rapidly and the algorithm to converge prematurely. To counteract this issue of premature convergence AMaLGaM scales the covariances by the factor $c^{\text{Multiplier}}$. This factor is adapted based on the improvements that are obtained by the sampling procedure. $c^{\text{Multiplier}}$ is initialized at 1, but increases by a factor $\eta^{INC} = 1/\eta^{DEC}$ if the best known objective value improves in one generation. Previous work has shown that $\eta^{DEC} = 0.9$ gives good results [19]. If no improvement of the elitist solution is found in a generation, $c^{\text{Multiplier}}$ is decreased by multiplying by η^{DEC}, but $c^{\text{Multiplier}} < 1$ is only allowed after a fixed number of consecutive generations without finding an improvement. The parameter that defines the number of generations without improvement for which a distribution multiplier smaller than 1 is not allowed, is the maximum no improvement stretch (NIS^{MAX}) parameter, which is set to $25 + \ell$ by default.

SDR

The adaptation of $c^{\text{Multiplier}}$ is not always required and can even slow down convergence when it is applied incorrectly. When the global optimum is located near the mean of the estimated distribution, the variance of the distribution should be decreased in order to sample solutions that approach the global optimum with an ever increasing speed. However, in this scenario AMaLGaM is very likely to generate a solution that improves the best known objective value, which will allow the AVS mechanism to increase the variance of the estimated distribution. AVS is therefore adapted using the SDR mechanism, which first selects all solutions $\boldsymbol{X}^{\text{Improved}}$ that have improved the best known objective value. The average over $\boldsymbol{X}^{\text{Improved}}$ is then computed as:

$$\boldsymbol{x}^{\text{avg-imp}} = \frac{1}{|\boldsymbol{X}^{\text{Improved}}|} \sum_{\boldsymbol{x} \in \boldsymbol{X}^{\text{Improved}}} \boldsymbol{x}$$

The SDR mechanism only allows an increase of $c^{\text{Multiplier}}$ if the value of any variable $\boldsymbol{x}_i^{\text{Improved}}$ is far away from its mean. Specifically, the distance of a variable to its mean is defined in terms of its standard deviation ratio. To compute this, the standard deviation ratio of this variable must be computed after transformation back to the standard normal distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$, because we need to take into account the impact of covariances that cause the principle component axes of the normal distribution to rotate away from the variable-wise axes. Removing these dependencies by transforming back to the standard normal distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ allows the evaluation of the standard deviation ratio of each variable independently. This inverse transformation is performed by applying the inverse operation performed to sample from the normal distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, i.e., by first subtracting $\boldsymbol{\mu}$ and then multiplying by the matrix $\boldsymbol{L}^{-1}$. Only if the standard deviation ratio of at least one variable is larger than θ^{SDR}, an increase of $c^{\text{Multiplier}}$ is triggered. By default θ^{SDR} is set to a value of 1, therefore only triggering an increase of $c^{\text{Multiplier}}$ when the average improvement of at least one variable, after the application of the inverse transformation, is more than 1 standard deviation away from the mean of this variable.

AMS

Selection drives the density contours of the estimated distribution to become more aligned with the contours of the objective function. This means that in slope-like regions of the search space, the distribution is biased toward generating many solutions that have a very similar objective value. However, on slope-like regions in the search space, the biggest improvements are to be found in directions perpendicular to the contours of the objective function. For this reason, AMS shifts a portion of the newly generated solutions in the direction of the generational improvement, which is the difference between the means of subsequent generations $\boldsymbol{\mu}^{\text{Shift}}(t) = \boldsymbol{\mu}(t) - \boldsymbol{\mu}(t-1)$, where $\boldsymbol{\mu}(t)$ denotes the mean in generation t. Only a fraction of the population is subject to AMS, in order to center the estimated distribution between the shifted and the non-shifted solutions. This assumes that the shifted solutions are of better quality than the non-shifted solutions and are therefore in the selected fraction of the population. Elitist solutions that are copied to the next generation are never subject to AMS, making the fraction of AMS-solutions equal to $\alpha^{\text{AMS}} = \frac{1}{2}\tau\frac{n}{n-n^{\text{elitist}}}$, where n is the population size and $n^{\text{elitist}} = 1$ is the number of copied elitist solutions. Multiplying α^{AMS} by the number of potential subjects for AMS leads to $n^{\text{AMS}} = \lfloor\frac{1}{2}\tau n\rfloor$, which is the number of randomly selected individuals that AMS will be applied to. These individuals are adapted through $\boldsymbol{x} \leftarrow \boldsymbol{x} + c^{\text{Multiplier}}\delta^{\text{AMS}}\boldsymbol{\mu}^{\text{Shift}}$, where $\delta^{\text{AMS}} = 2$ to potentially have an exponential speedup of the AMS. The shift is also multiplied by $c^{\text{Multiplier}}$, because a high multiplier indicates that improvements are far away from the mean.

Considering the large number of parameters of AMaLGaM, an overview is displayed in Table 2.1. The guideline values were previously empirically determined [14, 19]. The most influential parameter is the population size, for which the optimal size largely depends on the difficulty of the problem. The guideline for the population size is however a relatively large estimation, so it should suffice for almost any problem, at the cost of reduced efficiency on relatively easy problems.

Although AMaLGaM is a very powerful and robust algorithm, it does have its limitations, especially with regard to problem-specific adaptations. One limitation is the sampling method to generate new solutions, because in the unfactorized case, every sample is generated based on the full $\ell \times \ell$ covariance matrix $\boldsymbol{C}$, which implies a possible dependency between any pair of variables. To sample from the distribution that is described by this matrix and the mean vectors, a Cholesky decomposition $\boldsymbol{D} = \boldsymbol{L}\boldsymbol{L}^*$ must be computed, which is a very expensive operation requiring $O(\ell^3)$ computation time.

An additional limitation is the relatively large population size that is required for AMaLGaM to function optimally. The optimal population size on typical smooth benchmark problems was empirically determined

```
Function AMaLGaM :
    $\boldsymbol{P} \leftarrow$ Uniformly generated initial population
    $c^{\text{Multiplier}} \leftarrow 1$
    $n^{\text{AMS}} \leftarrow \alpha^{\text{AMS}}(n-1)$
    $\text{NIS} \leftarrow 0$
    $t \leftarrow 0$
    do
        $\boldsymbol{S} \leftarrow$ the best $\lfloor \tau n \rfloor$ solutions in $\boldsymbol{P}$
        $\boldsymbol{\mu}(t) \leftarrow \frac{1}{|\boldsymbol{S}|}\sum_{i=0}^{|\boldsymbol{S}|-1} \boldsymbol{S}_i$
        $\boldsymbol{\Sigma}(t) \leftarrow \frac{1}{|\boldsymbol{S}|}\sum_{i=0}^{|\boldsymbol{S}|-1} (\boldsymbol{S}_i - \boldsymbol{\mu}(t))(\boldsymbol{S}_i - \boldsymbol{\mu}(t))^T$
        $\boldsymbol{\mu}^{\text{Shift}}(t) \leftarrow \boldsymbol{\mu}(t) - \boldsymbol{\mu}(t-1)$
        $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu}(t)$
        $\boldsymbol{\Sigma} \leftarrow c^{\text{Multiplier}} \boldsymbol{\Sigma}(t)$
        $\boldsymbol{L}\boldsymbol{L}^* \leftarrow$ Cholesky decomposition of $\boldsymbol{\Sigma}$
        $\boldsymbol{P}_0 \leftarrow$ the best solution in $\boldsymbol{S}$
        $\boldsymbol{P}_{1 \ldots n-1} \leftarrow n-1$ samples from $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \boldsymbol{\mu} + \boldsymbol{L}\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$
        for $n^{\text{AMS}}$ random solutions in $\boldsymbol{P}_j (1 \leq j \leq n-1)$ do
            $\boldsymbol{P}_j \leftarrow \boldsymbol{P}_j + \delta^{\text{AMS}} c^{\text{Multiplier}} \boldsymbol{\mu}^{\text{Shift}}(t)$
        end
        if any $\boldsymbol{P}_i$ better than $\boldsymbol{P}_0 (1 \leq i \leq n-1)$ then
            $\text{NIS} \leftarrow 0$
            if $c^{\text{Multiplier}} < 1$ then
                $c^{\text{Multiplier}} \leftarrow 1$
            end
            $\boldsymbol{x}^{\text{avg-imp}} \leftarrow$ average of all $\boldsymbol{P}_i$ better than $\boldsymbol{P}_0 (1 \leq i \leq n-1)$
            $\text{SDR} \leftarrow \max_{0 \leq i \leq \ell-1} \left\{ \left| \left( L^{-1}(\boldsymbol{x}^{\text{avg-imp}} - \boldsymbol{\mu}) \right)_i \right| \right\}$
            if $\text{SDR} > \theta^{\text{SDR}}$ then
                $c^{\text{Multiplier}} \leftarrow \eta^{\text{INC}} c^{\text{Multiplier}}$
            end
        else
            if $c^{\text{Multiplier}} \leq 1$ then
                $\text{NIS} \leftarrow \text{NIS} + 1$
            end
            if $\left(c^{\text{Multiplier}} > 1\right)$ or $\left(\text{NIS} \geq \text{NIS}^{\text{MAX}}\right)$ then
                $c^{\text{Multiplier}} \leftarrow \eta^{\text{DEC}} c^{\text{Multiplier}}$
            end
            if $\left(c^{\text{Multiplier}} < 1\right)$ and $\left(\text{NIS} < \text{NIS}^{\text{MAX}}\right)$ then
                $c^{\text{Multiplier}} \leftarrow 1$
            end
        end
        $t \leftarrow t + 1$
    while opt. not found, max. eval. not reached and $c^{\text{Multiplier}} \geq 10^{-10}$
end
```

Algorithm 4: AMaLGaM [13]

Parameter	**Symbol**	**Guideline value**
Population size	n	$17+3\ell\sqrt{\ell}$
Selection ratio	τ	0.35
Elitist solutions copied	n^{elitist}	1
Distribution multiplier decrease	η^{DEC}	0.9
Standard deviation ratio threshold	θ^{SDR}	1
AMS fraction	α^{AMS}	$\frac{1}{2}\tau n/(n-n^{\text{elitist}})$
AMS step size	δ^{AMS}	2
Max. no improvement stretch	NIS^{MAX}	$25+\ell$

Table 2.1: Overview of AMaLGaM's parameters and their guideline values.

to be $17+3\ell^{1.5}$ [5]. Previous efforts to reduce the population size resulted in the Incremental AMaLGaM (iAMaLGaM) [14] algorithm that incrementally builds the normal distribution, which requires a population size of only $10\ell^{0.5}$. However, this change did not necessarily improve the performance of the algorithm, because it only performed better on a certain selection of problems [14]. The original AMaLGaM was still recommended by the authors for black-box optimization, because it would still generally obtain better results using fewer function evaluations.

2.1.3. AMaLGaM-FOS

Currently, AMaLGaM is implemented with a univariate model, a model that uses a Bayesian factorization, and a model that considers all dependencies. There is however no version of AMaLGaM that processes dependencies given by any arbitrary FOS structure. Such a model has a large influence on not only the efficiency of the algorithm, but also its capability of solving certain problems. Because we require AMaLGaM to use a marginal product FOS in certain experiments that will be described later, we implement a version of AMaLGaM, denoted AMaLGaM-FOS, that is capable of using any marginal product FOS dependency structure. For each FOS element, this version of AMaLGaM learns a separate multivariate normal probability distribution and uses this to sample values for new variables. Whenever a new solution needs to be generated, we iterate over all FOS elements, and generate all variables contained in each FOS element through its own probability distribution. Therefore, there is no reason to use a FOS that is not a marginal product, i.e., one where an element can be contained within more than one FOS element, because sampling a variable for a second time will simply overwrite the previously sampled variable.

Because the introduction of a FOS allowed GOMEA to have a separate distribution multiplier $c^{\text{Multiplier}}$ for each FOS element, this change should also be applied to AMaLGaM-FOS in order to really evaluate the effect of GOMEA itself. Identical to the AVS-SDR used in GOMEA, each FOS element $\boldsymbol{F}_i$ uses a separate distribution multiplier $c^{\text{Multiplier}}_{\boldsymbol{F}_i}$, which is adapted based on the improvements achieved by the variables in $\boldsymbol{F}_i$.

2.1.4. Parameter-less scheme

Out of all parameters of an EA, the population size is certainly the most influential one. Using a population size that is too small will make it impossible to solve a problem, while making the population too large will significantly slow down the algorithm. A comparison between different EAs should have both algorithms using the optimal population size for each individual algorithm, because a single fixed population size would give one of the two algorithms an inherent advantage. Additionally, the optimal population size largely depends on the optimization problem and the number of variables that need to be optimized, so it should be optimized for every combination of these factors independently. This can be done by performing a so-called bisection, which runs the algorithm for different population sizes and finds the number of evaluations required to reach a certain value for each of these population sizes. Considering the parabolic shape of the relationship between the population size and the required number of evaluations, a large portion of the search space can be excluded after each step of the bisection. Each step therefore narrows down the range that includes the optimal population size. To mostly eliminate the random nature of the EAs, every data point of the bisection must however be an average over at least 30 runs of the algorithm.

Even then the optimal population size should preferably be the average over multiple bisections. Performing bisections for every utilized number of variables on every problem will therefore take a very large amount of time.

Instead, the population size parameter can be eliminated completely through the Harik and Lobo parameter-less scheme introduced in [23, 35]. This scheme removes the population size parameter by having multiple populations with different sizes running simultaneously. Generations of these populations are interleaved, with smaller populations performing generations more frequently. Small populations are terminated when a population of a larger size is deemed to be of higher quality. This can be done, because small populations converge more quickly than large populations. Therefore, a small population is not expected to catch up again to the large population.

```
Function Harik-Lobo($g$) :
    $i \leftarrow 0$
    while not terminated do
        P[i] ← new population with size $p_b \cdot 2^i$
        foreach $k \in \{0, 1, ..., i\}$ do
            repeat $g^{i-k}$ times
                if P[$k$] has not terminated then
                    Perform a single generation of P[$k$]
                end
            end
        end
        Terminate small populations of low quality
        $i \leftarrow i + 1$
    end
end
```

Algorithm 5: Parameter-less scheme by Harik and Lobo [23, 35], with g the subgeneration scaling factor and p_b the base population size.

Pseudo-code of the parameter-less scheme is displayed in Algorithm 5. During each iteration of this scheme, a new population is initiated with twice the population size of the previously largest population. This new population then enters the pool of active populations. All active populations perform a varying number of generations in each iteration, with the smallest populations performing a larger number of iterations. The factor with which the number of generations scales is the subgeneration scaling factor g, for which often the values 4 or 8 are used. For the single-objective experiments in this chapter, a subgeneration factor of 8 was used, and for the multi-objective experiments a factor of 4 was used. The lower factor for the multi-objective experiments was required due to a liberal termination criterion, which keeps small populations running for a relatively long period of time. Compared to a population with size p, the population with size $0.5p$ will perform g times as many generations in one iteration of the parameter-less scheme. As stated previously, smaller populations converge relatively fast, after which there is no reason to continue performing the algorithm on these populations. Due to the exponentially growing number of generations, the vast majority of time would be spent on small populations that are nearing convergence. To prevent this, we terminate a small population when a larger population has reached a higher quality. For this purpose, in the case of single-objective optimization, the quality of a population is defined by its average objective value. Whenever a population $P[i]$ is found to have a worse average objective value than some population $P[j]$ with $|P[j]| > |P[i]|$, $P[i]$ and any population having a smaller population size are terminated.

2.2. The Single-Objective Real-Valued GOMEA

To develop a real-valued adaptation of GOMEA, GOMEAs crossover operations must be adapted to the real-valued domain. Recombination can no longer be based only on a parent and a donor, because this will only copy existing values of variables and only capture a minuscule part of the real-valued search space. Instead, new solutions will be generated in a similar way to AMaLGaM, because AMaLGaM can be considered state of the art in real-valued optimization. Each FOS element will have its own model responsible for generating partial solutions, each consisting of a very own estimated probability distribution and a distribution multiplier. Each of the AVS, SDR and AMS procedures will also be applied to each partial solution separately. The partial solutions that are generated through the AMaLGaM procedure are then inserted into a parent solution based on GOMEA.

2.2.1. Basic structure

The basic structure of the single-objective real-valued GOMEA is displayed in Algorithm 6. In each generation, first, a selection $\boldsymbol{S}$ of the best solutions from the previous generation is made. During the first generation, this selection $\boldsymbol{S}$ is based on a population that is sampled uniformly in a range specified by the user.

A very important aspect of GOMEA is finding an appropriate FOS dependency structure. Since every problem involves a different set of dependencies, we should not limit ourselves to a fixed dependency structure, because this will work only well on a small selection of benchmark problems. A number of possible dependencies, namely the univariate, marginal product and linkage tree structures, were already introduced in Section 2.1.2. Of these structures the univariate structure should be the structure of choice for any problem with no dependencies between variables, because there is no point in trying to learn dependencies when there are none. If a problem does involve dependencies, we assume that the structure of these dependencies is largely known and can be represented by a marginal product FOS. This assumption can be made, because the focus of this thesis is GBO.

```
Function GOMEA:
    $\boldsymbol{P} \leftarrow$ Uniformly generated initial population
    NIS $\leftarrow 0$
    $t \leftarrow 0$
    while not terminated do
        $\boldsymbol{S} \leftarrow$ the best $\lfloor \tau n \rfloor$ individuals in $\boldsymbol{P}$
        for $\boldsymbol{F}_i \in \boldsymbol{F}$ do
            Estimate distributions $\mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{F}_i}, \boldsymbol{\Sigma}_{\boldsymbol{F}_i})$ with maximum-likelihood based on $\boldsymbol{S}$
            $\boldsymbol{\mu}^{\text{Shift}}_{\boldsymbol{F}_i}(t) \leftarrow \boldsymbol{\mu}_{\boldsymbol{F}_i}(t) - \boldsymbol{\mu}_{\boldsymbol{F}_i}(t-1)$
        end
        $\boldsymbol{P}_i(0 \leq i \leq n^{\text{elitist}}) \leftarrow$ the best $n^{\text{elitist}}$ individuals in $\boldsymbol{P}$
        $\boldsymbol{P} \leftarrow$ generateNewSolutions
        $t \leftarrow t+1$
    end
end
```

Algorithm 6: The basic structure of real-valued GOMEA

Because GOMEA independently recombines subsets of solutions, many of GOMEAs learning coefficients are learned separately for each FOS element. Most notably a multivariate normal probability distribution $\mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{F}_i}, \boldsymbol{\Sigma}_{\boldsymbol{F}_i})$ and a distribution multiplier $c^{\text{Multiplier}}_{\boldsymbol{F}_i}$ are learned separately for each FOS element $\boldsymbol{F}_i$. Learning a separate distribution means that the algorithm estimates a normal distribution based on the calculated means and covariances of each pair of variables in $\boldsymbol{F}_i$. This allows the algorithm to completely ignore any dependencies between variables that are not present in the same FOS element. Therefore, each probability distribution $\mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{F}_i}, \boldsymbol{\Sigma}_{\boldsymbol{F}_i}\right)$ is a multivariate distribution modeling all variables that are present in the respective FOS element $\boldsymbol{F}_i$. For any FOS element $\boldsymbol{F}_i$ of size k we require a k-dimensional vector $\boldsymbol{\mu}_{\boldsymbol{F}_i}$ and a $k \times k$ matrix $\boldsymbol{\Sigma}_{\boldsymbol{F}_i}$ to define the k-variate probability distribution $\mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{F}_i}, \boldsymbol{\Sigma}_{\boldsymbol{F}_i}\right)$. The means $\boldsymbol{\mu}_{\boldsymbol{F}_i}$ and covariances $\boldsymbol{\Sigma}_{\boldsymbol{F}_i}$ are estimated with maximum likelihood from the selection S, which means that we calculate the normal probability distribution that is the most likely to sample the current values of selection $\boldsymbol{S}$. Again this maximum likelihood estimate only considers variables that are present in the respective FOS element $\boldsymbol{F}_i$. For any two variables $u, v \in \boldsymbol{F}_i$ the maximum likelihood estimates of the means $\boldsymbol{\mu}_{\boldsymbol{F}_i}$ and covariances $\boldsymbol{\Sigma}_{\boldsymbol{F}_i}$ are formally defined as:

$$
\begin{aligned}
\left(\boldsymbol{\mu}_{\boldsymbol{F}_i}\right)_u &= \frac{1}{|S|}\sum_{\boldsymbol{s}\in S} \boldsymbol{s}_u \\
\left(\boldsymbol{\Sigma}_{\boldsymbol{F}_i}\right)_{(u,v)} &= \frac{1}{|S|}\sum_{\boldsymbol{s}\in S} \left(\boldsymbol{s}_u - (\boldsymbol{\mu}_{\boldsymbol{F}_i})_u\right)\left(\boldsymbol{s}_v - (\boldsymbol{\mu}_{\boldsymbol{F}_i})_v\right)
\end{aligned}
$$

For each covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{F}_i}$ a Cholesky decomposition $\boldsymbol{D}_{\boldsymbol{F}_i} = \boldsymbol{L}_{\boldsymbol{F}_i}\boldsymbol{L}^*_{\boldsymbol{F}_i}$ is computed, which is used by SDR and to sample new partial solutions. After estimating probability distributions for each element $\boldsymbol{F}_i$ of the FOS dependency structure, $n^{\text{elitist}} = 1$ elitist solutions are copied to the new population.

2.2.2. Generating New Solutions

A new population is generated by generating new partial solutions and inserting these into existing solutions. This entire procedure is displayed in Algorithm 7. The *generateNewSolutions* procedure starts off by, for each FOS element $\boldsymbol{F}_i$, generating new partial solutions $\boldsymbol{y}_{\boldsymbol{F}_i}$ for each individual $\boldsymbol{x}$ in the population. If $\boldsymbol{x}$ is selected as one of the $n^{\text{AMS}} = \frac{1}{2}\tau n$ subjects for AMS, AMS is applied to all variables of the partial solution $\boldsymbol{y}_{\boldsymbol{F}_i}$ by adding $c^{\text{Multiplier}}_{\boldsymbol{F}_i}\delta^{\text{AMS}}\boldsymbol{\mu}^{\text{Shift}}_{\boldsymbol{F}_i}(t)$ to $\boldsymbol{y}_{\boldsymbol{F}_i}$. Recombination of an existing solution $\boldsymbol{x}$ and a vector of newly generated values $\boldsymbol{y}_{\boldsymbol{F}_i}$ is indicated by $\boldsymbol{x} \odot \boldsymbol{y}_{\boldsymbol{F}_i}$. This recombination operator inserts the values of all variables $(\boldsymbol{y}_{\boldsymbol{F}_i})_u \in \boldsymbol{y}_{\boldsymbol{F}_i}$ into $\boldsymbol{x}$. If this recombination operation leads to an improvement of the objective value of $\boldsymbol{x}$, the modification of $\boldsymbol{x}$ is accepted. Otherwise, $\boldsymbol{x}$ is returned to its previous state. To eliminate the continuous build-up of small numerical errors caused by partial evaluations, the entire population is re-evaluated once every 50 generations. Iterating over the population within an iteration of a single FOS element is different from the original discrete GOMEA, as the order of the outer loops is reversed. The impact of this on diversity is very limited compared to discrete GOMEA, where the recombination operation copies variables from different individuals, because the real-valued GOMEA samples new values independently from existing solutions. Only when dependencies exist between variables in separate FOS elements, the reversing of the loops can influence whether a recombination operation is accepted. However, reversing the outer loops makes it relatively simple to find the improvements that were caused by each separate FOS element, which is required to adapt the distribution multipliers.

2.2.3. AVS and SDR

Each FOS element $\boldsymbol{F}_i$ now also has a separate distribution multiplier $c^{\text{Multiplier}}_{\boldsymbol{F}_i}$, which enables this FOS element to scale its covariance matrix based on improvements that were caused by this specific FOS element. SDR is then applied to each distribution multiplier separately, meaning that a distribution multiplier $c^{\text{Multiplier}}_{\boldsymbol{F}_i}$ is only multiplied by the incrementing factor η^{INC} if the average improvement of at least one variable $\boldsymbol{x}_u$ with $u \in \boldsymbol{F}_i$ was far away from the estimated mean of this variable. To calculate how far away from its mean the value of a variable was originally generated, we first calculate the average improvement vector $\boldsymbol{x}^{\text{avg-imp}}_{\boldsymbol{F}_i}$. For any $u \in \boldsymbol{F}_i$, this vector describes the average of each $\boldsymbol{x}_u$ with $\boldsymbol{x} \in \boldsymbol{X}^{\text{Improved}}_{\boldsymbol{F}_i}$. The set $\boldsymbol{X}^{\text{Improved}}_{\boldsymbol{F}_i}$ contains all individuals for which the recombination operation using $\boldsymbol{F}_i$ led to a solution that improved the best known objective value. The vector $\boldsymbol{x}^{\text{avg-imp}}_{\boldsymbol{F}_i}$ is transformed in order to describe the averages of the values that were originally generated by the distribution $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. This transformation is applied to $\boldsymbol{x}^{\text{avg-imp}}_{\boldsymbol{F}_i}$ by first subtracting $\boldsymbol{\mu}_{\boldsymbol{F}_i}$ and then multiplying by $\boldsymbol{L}^{-1}_{\boldsymbol{F}_i}$. If any element of $\boldsymbol{L}^{-1}_{\boldsymbol{F}_i}\left(\boldsymbol{x}^{\text{avg-imp}}_{\boldsymbol{F}_i} - \boldsymbol{\mu}_{\boldsymbol{F}_i}\right)$ is larger than 1, some improvements caused by $\boldsymbol{F}_i$ were far away from the mean, and $c^{\text{Multiplier}}_{\boldsymbol{F}_i}$ is increased by a factor η^{INC}. The complete procedure of generating new solutions is displayed in Algorithm 8.

2.2.4. Second-round AMS and Randomized Recombination Acceptation

Even though selecting only solutions that improve the objective value of their parent greatly improves the speed of convergence on a set of simple problems, this approach is more likely to get stuck in local minima, and to stall on problems where the search distribution does not create better solutions with very high probability due to a mismatch with the problem landscape. This problem was observed on benchmark problems where dependencies exist that are not captured by the structure of the FOS, most notably the Rosenbrock function with the univariate FOS. The previously discussed GOMEA was unable to solve the Rosenbrock function using a univariate structure, while the original AMaLGaM is able to solve it using the

```
Function generateNewSolutions:
    for $\boldsymbol{F}_i \in \boldsymbol{F}$ do
        for $\boldsymbol{x} \in \boldsymbol{P}$ do
            $\boldsymbol{b}_{\boldsymbol{F}_i} \leftarrow \boldsymbol{x}_{\boldsymbol{F}_i} \leftarrow (\boldsymbol{x}_u)$ for $u \in \boldsymbol{F}_i$
            $f_b \leftarrow f(\boldsymbol{x})$
            $\boldsymbol{y}_{\boldsymbol{F}_i} \leftarrow \mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{F}_i}, c^{\text{Multiplier}}_{\boldsymbol{F}_i}\boldsymbol{\Sigma}_{\boldsymbol{F}_i}\right)$
            if $\boldsymbol{x} \in$ {the first $\lfloor\frac{1}{2}\tau n\rfloor$ individuals in $\boldsymbol{P}$} then
                $\boldsymbol{y}_{\boldsymbol{F}_i} \leftarrow \boldsymbol{y}_{\boldsymbol{F}_i} + c^{\text{Multiplier}}_{\boldsymbol{F}_i}\delta^{\text{AMS}}\boldsymbol{\mu}^{\text{Shift}}_{\boldsymbol{F}_i}$
            end
            $\boldsymbol{x} \leftarrow \boldsymbol{x} \odot \boldsymbol{y}_{\boldsymbol{F}_i}$
            if not $f(\boldsymbol{x}) < f_b$ then
                $\boldsymbol{x} \leftarrow \boldsymbol{x} \odot \boldsymbol{b}_{\boldsymbol{F}_i}$
            end
        end
        AdaptMultiplier($c^{\text{Multiplier}}_{\boldsymbol{F}_i}$)
    end
    for $\boldsymbol{x} \in \boldsymbol{P}$ do
        if $\boldsymbol{x}$ has not improved since previous generation then
            $\boldsymbol{x} \leftarrow$ ForcedImprovements($\boldsymbol{x}$)
        end
    end
    return $\boldsymbol{P}$
end
```

Algorithm 7: GOMEA's method of generating new solutions, which was later modified to the method displayed in Algorithm 9.

```
Function AdaptMultiplier($c^{Multiplier}_{\boldsymbol{F}_i}$):
    if $\boldsymbol{X}^{\text{Improved}}_{\boldsymbol{F}_i} \neq \emptyset$ then
        NIS $\leftarrow 0$
        if $c^{\text{Multiplier}}_{\boldsymbol{F}_i} < 1$ then $c^{\text{Multiplier}}_{\boldsymbol{F}_i} \leftarrow 1$
        SDR $\leftarrow \max_{j \in \boldsymbol{F}_i}\left\{ \left| \left( L^{-1}_{\boldsymbol{F}_i}(\boldsymbol{x}^{\text{avg-imp}}_{\boldsymbol{F}_i} - \boldsymbol{\mu}_{\boldsymbol{F}_i}) \right)_j \right| \right\}$
        if SDR $> \theta^{\text{SDR}}$ then
            $c^{\text{Multiplier}}_{\boldsymbol{F}_i} \leftarrow \eta^{\text{INC}} c^{\text{Multiplier}}_{\boldsymbol{F}_i}$
        end
    else
        if $c^{\text{Multiplier}}_{\boldsymbol{F}_i} \leq 1$ then
            NIS $\leftarrow$ NIS $+ 1$
        end
        if $\left(c^{\text{Multiplier}}_{\boldsymbol{F}_i} > 1\right)$ or $\left(\text{NIS} \geq \text{NIS}^{\text{MAX}}\right)$ then
            $c^{\text{Multiplier}}_{\boldsymbol{F}_i} \leftarrow \eta^{\text{DEC}} c^{\text{Multiplier}}$
        end
        if $\left(c^{\text{Multiplier}}_{\boldsymbol{F}_i} < 1\right)$ and $\left(\text{NIS} < \text{NIS}^{\text{MAX}}\right)$ then
            $c^{\text{Multiplier}}_{\boldsymbol{F}_i} \leftarrow 1$
        end
    end
end
```

Algorithm 8: The function that adapts $c^{\text{Multiplier}}_{\boldsymbol{F}_i}$ of FOS element $\boldsymbol{F}_i$ depending on the improvements found by this FOS element.

same dependency structure. For this reason, a number of adaptations are made to GOMEA that enable it to escape local optima while minimizing the loss of efficiency on more trivial benchmark problems. Pseudo-code of GOMEA's *generateNewSolutions* method, after the aforementioned adaptations, is displayed in Algorithm 9.

Firstly, a second round of AMS is applied after the complete recombination procedure. This round of AMS is in fact very similar to that of AMaLGaM apart from the fact that GOMEA only accepts improvements during this round. For a selection of $\lfloor \frac{1}{2}\tau n \rfloor$ individuals in the offspring, AMS is applied by adding $\delta^{\text{AMS}}\mu^{\text{Shift}}$ to each of the variables. In contrast to the first round of AMS, no multiplication of μ^{Shift} by a distribution multiplier is performed, because there exists no multiplier that is attributed to the complete set of variables. Only if the application of AMS leads to an improvement of the the individual's objective value, or with a probability of p^{accept}, the change is accepted. Because this round of AMS is applied to all variables simultaneously it is capable of dealing with high-order dependencies that are not captured by the FOS. This allows the algorithm to find improvements when the adaptation of one variable can never lead to big improvements, but the adaptation of multiple variables can.

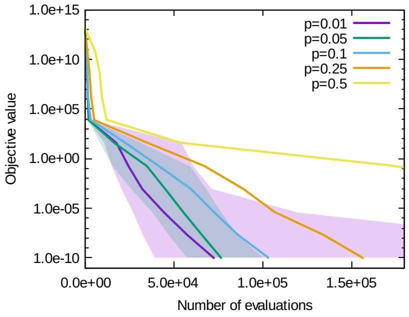


Figure 2.3: Different values of p^{accept} on the Rosenbrock function with 1000 variables.

Secondly, a small probability of accepting a solution regardless of it being an improvement is introduced. We call this adaptation the Randomized Recombination Acceptation (RRA). After generating a new partial solution to be inserted into a certain parent solution, the newly recombined solution is evaluated and compared to the parent solution. With a probability of p^{accept} the newly recombined solution is accepted regardless of the outcome of the comparison with its parent solution. Therefore, with a probability of $1 - p^{\text{accept}}$ the solution is accepted only if it improves the objective value of the parent. A p^{accept} that is too high will make GOMEA behave too much like AMaLGaM in the sense that it loses the aggressive selection procedure which makes it so efficient on the set of benchmark problems where all dependencies are accounted for. Instead setting p^{accept} too low will make GOMEA sometimes unable to solve the Rosenbrock function using a univariate FOS or it will simply take incredibly long for the algorithm to escape local minima to the point where using AMaLGaM would be more efficient. This behavior is shown in Figure 2.3, showing the interpolated medians of 30 independent runs. Interdecile ranges are only shown for $p^{\text{accept}} = 0.01$ and $p^{\text{accept}} = 0.05$ to increase readability. These two values of p^{accept} achieved a very similar median, but a large portion of the runs with $p^{\text{accept}} = 0.01$ converged very slowly, as indicated by the large interdecile range. Therefore, choosing p^{accept} to be a probability of 5% was found to be the most effective for solving the Rosenbrock function using a univariate FOS.

2.2.5. Forced Improvements

On certain problems it is possible for portions of the population to get stuck in local minima, because GOMEA allows modifications that do not lead to an improvement only very rarely. Individuals located near such local minima can influence the estimated normal distribution to be shifted towards the local minimum, possibly slowing down convergence or causing complete termination. Therefore, the Forced Improvements (FI) technique, that is also employed in the discrete GOMEA [11], is adapted to the real-valued domain. Pseudo-code of this technique is displayed in Algorithm 10. The forced improvements method selects individuals that have not improved during the last generation and forces them to improve by shifting their variables towards those of the elitist solution $\boldsymbol{x}^{\text{elitist}}$. In each round of the forced improvements, the individual's variables are replaced by a weighted average over its own variables and those of the elitist solution. The weight α of the individual's own variables starts at 0.5 and decreases with a factor 2 after each round until $\alpha < 0.05$. All FOS elements are considered separately and after replacing all variables of one FOS element $\boldsymbol{F}_i$, the individual is checked for improvement. If the newly adapted individual has obtained a better objective value than before, this change is accepted and the forced improvements method is terminated. Otherwise the individual is returned to its previous state and the algorithm continues with the next FOS element. If the forced improvements method reaches a value of $\alpha < 0.05$ without finding any improvements, all variables of the individual are replaced by those of the elitist solution $\boldsymbol{x}^{\text{elitist}}$.

```
Function generateNewSolutions:
    for $\boldsymbol{F}_i \in \boldsymbol{F}$ do
        for $\boldsymbol{x} \in \boldsymbol{P}$ do
            $\boldsymbol{b}_{\boldsymbol{F}_i} \leftarrow \boldsymbol{x}_{\boldsymbol{F}_i} \leftarrow (\boldsymbol{x}_u)$ for $u \in \boldsymbol{F}_i$
            $f_b \leftarrow f(\boldsymbol{x})$
            $\boldsymbol{y}_{\boldsymbol{F}_i} \leftarrow \mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{F}_i}, c^{\text{Multiplier}}_{\boldsymbol{F}_i}\boldsymbol{\Sigma}_{\boldsymbol{F}_i}\right)$
            if $\boldsymbol{x} \in$ {the first $\lfloor\frac{1}{2}\tau n\rfloor$ individuals in $\boldsymbol{P}$} then
                $\boldsymbol{y}_{\boldsymbol{F}_i} \leftarrow \boldsymbol{y}_{\boldsymbol{F}_i} + c^{\text{Multiplier}}_{\boldsymbol{F}_i}\delta^{\text{AMS}}\boldsymbol{\mu}^{\text{Shift}}_{\boldsymbol{F}_i}$
            end
            $\boldsymbol{x} \leftarrow \boldsymbol{x} \odot \boldsymbol{y}_{\boldsymbol{F}_i}$
            if not $\left(f(\boldsymbol{x}) < f_b \text{ or } U(0,1) < p^{\text{accept}}\right)$ then
                $\boldsymbol{x} \leftarrow \boldsymbol{x} \odot \boldsymbol{b}_{\boldsymbol{F}_i}$
            end
        end
        AdaptMultiplier($c^{\text{Multiplier}}_{\boldsymbol{F}_i}$)
    end
    for $\boldsymbol{x} \in \boldsymbol{P}$ do
        if $\boldsymbol{x}$ has not improved since previous generation then
            $\boldsymbol{x} \leftarrow$ ForcedImprovements($\boldsymbol{x}$)
        end
    end
    for $\boldsymbol{x} \in$ {the first $\frac{1}{2}\tau n$ individuals in $\boldsymbol{P}$} do
        $\boldsymbol{b} \leftarrow \boldsymbol{x}$
        $\boldsymbol{x} \leftarrow \boldsymbol{x} + \delta^{\text{AMS}}\boldsymbol{\mu}^{\text{Shift}}$
        if not $\left(f(\boldsymbol{x}) < f(\boldsymbol{b}) \text{ or } U(0,1) < p^{\text{accept}}\right)$ then
            $\boldsymbol{x} \leftarrow \boldsymbol{b}$
        end
    end
    return $\boldsymbol{P}$
end
```

Algorithm 9: GOMEA's method of generating new solutions, including second-round AMS and RRA.

```
Function ForcedImprovements($\boldsymbol{x}$):
    $\boldsymbol{x}_{\boldsymbol{F}_i} \leftarrow (\boldsymbol{x}_u)$ for $u \in \boldsymbol{F}_i$
    $\alpha \leftarrow 0.5$
    while $\alpha \geq 0.05$ do
        for $\boldsymbol{F}_i \in \boldsymbol{F}$ do
            $\boldsymbol{x}^{\text{elitist}}_{\boldsymbol{F}_i} \leftarrow \left(\boldsymbol{x}^{\text{elitist}}_u\right)$ for $u \in \boldsymbol{F}_i$
            $\boldsymbol{y} \leftarrow \boldsymbol{x} \odot \left(\alpha\boldsymbol{x}_{\boldsymbol{F}_i} + (1-\alpha)\boldsymbol{x}^{\text{elitist}}_{\boldsymbol{F}_i}\right)$
            if $f(\boldsymbol{y}) < f(\boldsymbol{x})$ then
                $\boldsymbol{x} \leftarrow \boldsymbol{y}$
                return
            end
        end
        $\alpha \leftarrow 0.5\alpha$
    end
    $\boldsymbol{x} \leftarrow \boldsymbol{x}^{\text{elitist}}$
end
```

Algorithm 10: The Forced Improvement function aimed at moving solutions out of local optima.

2.2.6. Termination

Finally, we have to determine when to stop the overall algorithm. The following termination conditions are used by default:

1. A maximum predefined number of evaluations has been reached.
2. A maximum predefined time limit has been reached.
3. Any solution has obtained an objective value better than a predefined value-to-reach.
4. The variance of the entire population's objective value has reached a value below a predefined fitness variance tolerance.
5. The distribution multiplier $c_{\boldsymbol{F}_i}^{\text{Multiplier}}$ of each FOS element $\boldsymbol{F}_i$ has reached a value below 10^{-10}.

The first three termination conditions allow the user to bound the total time or effort available to the algorithm, while the fourth condition allows the user to specify conditions under which the algorithm is believed to have converged. Finally the fifth condition terminates the algorithm when it has not found any improvements for an extended number of generations, because each $c^{\text{Multiplier}}$ is only decreased by a factor $\eta^{\text{DEC}} = 0.9$ if no improvements have been found during a single generation.

3

Designing the Multi-Objective Real-Valued GOMEA

Firstly, a general overview of multi-objective optimization is discussed in this chapter, after which the state of the art in multi-objective EAs is discussed. This is followed by the section that covers the adaptation of the real-valued GOMEA to the domain of multi-objective optimization.

3.1. Background

As introductory matter to the design of the multi-objective real-valued GOMEA, multi-objective optimization is first discussed, followed by the discussion of a number of state-of-the-art algorithms in the field of multi-objective optimization.

3.1.1. Multi-Objective Optimization

Certain problems do not just have one objective function, but a set of multiple conflicting objectives. In this case it is not possible to declare one solution $\boldsymbol{x}$ to be better than a different solution $\boldsymbol{y}$ unless $f_i(\boldsymbol{x}) < f_i(\boldsymbol{y})$ for at least one objective, and $f_i(\boldsymbol{x}) \leq f_i(\boldsymbol{y})$ for all others. We then say that $\boldsymbol{x}$ (Pareto) dominates $\boldsymbol{y}$, denoted as $\boldsymbol{x} \succ \boldsymbol{y}$ [15]. This is illustrated in Figure 3.1, where one arbitrarily selected point is highlighted in red, with the area it dominates shaded in red. Any solution within this area is dominated by the highlighted solution. Unlike single-objective optimization where there is exactly one globally optimal objective value, there now exists a *set* of optimal non-dominated solutions $\boldsymbol{X}^*$ for which:

$$\textbf{for all } \boldsymbol{x}, \boldsymbol{y} \in \boldsymbol{X}^* : \boldsymbol{x} \not\succ \boldsymbol{y} \textbf{ and for all } \boldsymbol{x} \in \boldsymbol{X}^* \textbf{ there is no } \boldsymbol{y} \textbf{ s.t. } \boldsymbol{y} \succ \boldsymbol{x}$$

A set of non-dominated solutions is called a Pareto set, which in the objective space forms a so-called Pareto front, of which two examples are displayed in Figure 3.1.

A common approach to solving multi-objective optimization problems consists of optimizing a weighted sum of all objectives, because this allows the use of single-objective algorithms. A Pareto set of solutions could be obtained by performing multiple runs of a single-objective algorithm, each using a different set of weights. However, assigning appropriate weights to all objectives can be very difficult by itself. To obtain a sufficiently sized Pareto set, a direct multi-objective approach will be more efficient than repeated single-objective optimization. Additionally, using a linear combination of weights, it is not possible to find solutions on concave parts of the Pareto front. A multi-objective approach that maintains a Pareto set of solutions is therefore often preferred over repeated single-objective optimization. Evolutionary Algorithms have previously been shown to be very suitable for such a multi-objective approach [15].

3.1.2. Multi-Objective Evolutionary Algorithms

The goal of a multi-objective EA (MOEA) is to approximate the set of all Pareto-optimal solutions as good as possible. Although different definitions exist of what makes for a good approximation front, generally speaking this means finding a set of solutions that is close to the optimal Pareto front in all regions of the

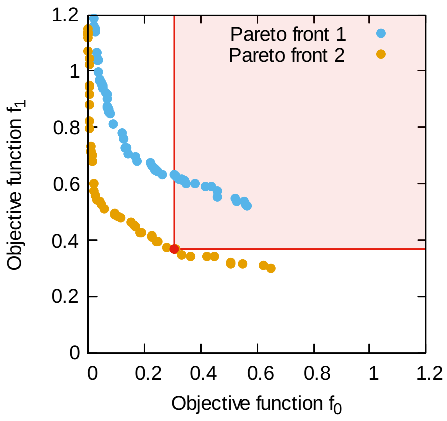


Figure 3.1: Two arbitrary Pareto fronts. An arbitrary solution is highlighted in red and the space that it dominates is marked in red.

front, i.e., it is moreover well-spread. This can be achieved by a niching mechanism, such as sharing [18], that introduces search bias towards different parts along the Pareto front in order to prevent the population from converging to a single point. Additionally, instead of keeping track of the best-known solution as in a single-objective EA, an MO-EA requires keeping track of a complete set of best-known non-dominating solutions, which are stored in the so-called elitist archive. Whenever a new solution is encountered, it is compared to each solution in the elitist archive. If the new solution is not dominated by any existing solution in the elitist archive, it is inserted into the archive and every previously existing solution that is now dominated, is removed from the archive.

The creation of a selection of the best individuals in the population now also has to account for multiple objectives. Because multiple objectives exist, the quality of an individual is determined relative to the rest of the population. A commonly adopted approach is the so-called nondominated sorting method [40]. For this method, a domination count is calculated for each individual, indicating the number of times it is dominated by different solutions in the population. Each individual that is not dominated by any other individual in the population is then assigned the rank 0. During the following iteration of the ranking procedure, domination counts are recomputed for all individuals that have not yet been ranked, resulting in a different set of non-dominated solutions to which the rank 1 is assigned. This procedure repeats until each individual has been ranked. During each iteration, the rank that is assigned to a set of non-dominated solutions is incremented by one. A selection can then be composed of a fraction of individuals with the lowest rank. For one specific rank this can cause some solutions to be selected while others of the same rank are not. From the solutions of this rank, a set is selected to maximize diversity by first selecting one individual that is a maximum in a random dimension and then iteratively adding the individuals that are the farthest away from the already selected individuals. This process terminates when the combined set of individuals selected by this process and the individuals of lower ranks have reached the desired size.

MAMaLGaM

In order to deal with multiple conflicting objectives the multi-objective AMaLGaM (MAMaLGaM) was introduced [7, 37]. Its pseudo-code is displayed in Algorithm 11. MAMaLGaM uses an elitist archive to keep track of a Pareto set of solutions. To keep the elitist archive from growing to exceptional sizes, a target elitist archive size must be supplied. Whenever at the end of a generation the size of the archive is higher than 1.25 times the target archive size, the archive removes solutions that are very close to other solutions in the archive. This is done by discretizing the objective space into a uniform grid. For each cell of the grid only a single random solution is maintained in the archive, removing all other solutions in this cell. Different resolutions of the grid are attempted through a binary search method, rebuilding the archive for each resolution that is attempted. The binary search is terminated when a resolution has been found that achieves an archive size between 0.75 and 1.25 times the target size. This grid resolution is then maintained until the elitist archive has again grown to a size above the upper bound, meaning that new solutions are not added to the elitist archive if a solution already exists in the same grid cell, unless the new solution dominates the existing one.

To more efficiently find different solutions at different parts along the Pareto front, MAMaLGaM applies

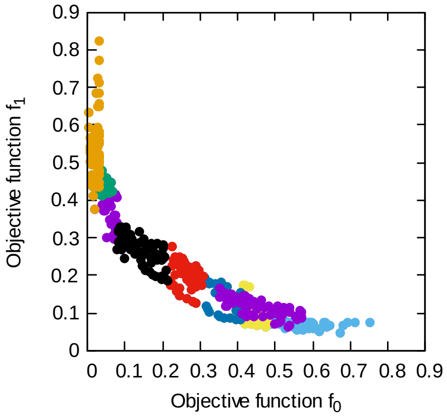


Figure 3.2: Each of MAMaLGaM's clusters displayed in a different color.

niching by utilizing a set of clusters $\boldsymbol{C} = \{\boldsymbol{C}_1, \boldsymbol{C}_2, \ldots, \boldsymbol{C}_q\}$ that each cover a different part of the Pareto front. A separate model is learned for each cluster to be able to sample high quality solutions in different sections of the Pareto front. Each model of a cluster $\boldsymbol{C}_j$ includes one multi-variate normal probability distribution $\mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{C}_j}, \boldsymbol{\Sigma}_{\boldsymbol{C}_j})$ and a distribution multiplier $c^{\text{Multiplier}}_{\boldsymbol{C}_j}$.

Clusters are created through the k-leader-means clustering method [6]. At the start of each generation, all solutions from the selection are distributed into q clusters of size $n^{\boldsymbol{C}} = \lfloor 2\lfloor \tau n \rfloor / q \rfloor$. For each objective, first a single-objective cluster is created, which contains the best $n^{\boldsymbol{C}}$ solutions in that objective. Such single-objective clusters force the population towards the edges of the Pareto front by only selecting based on one specific objective. The remaining clusters are created by first selecting q cluster leaders that are heuristically selected to maximize the distance between them. The first leader is chosen by selecting a point that has a maximum value in a randomly selected dimension. Every subsequent leader is then iteratively selected as the point that is the most distant from the set of already selected leaders. For every cluster leader the Euclidean distance to every solution in the selection is then computed after which the $n^{\boldsymbol{C}}$ closest solutions to each cluster are assigned to it. This means that some solutions can be assigned to multiple clusters while others may be assigned to none. The cluster size $n^{\boldsymbol{C}}$ is selected to have a substantial probability of overlapping clusters, because this increases the likelihood of solutions being more evenly spread out across the Pareto front. After a set of q clusters has been created, for each cluster a separate multi-variate normal probability distribution is estimated with maximum likelihood from the individuals in this cluster. Because each cluster inhabits a different part of the Pareto front, they could converge at different speeds and require a separate distribution multiplier $c^{\text{Multiplier}}_{\boldsymbol{C}_j}$.

Before the procedure of sampling new individuals is performed, a set of elitist solutions is first copied into the offspring. Each elitist individual is first assigned to its nearest cluster, after which $n^{\text{elitist}}_{\boldsymbol{C}_j} \leq \lfloor \lfloor \tau n \rfloor / q \rfloor$ of these individuals are copied into the offspring per cluster. If the number of elitist solutions assigned to a cluster exceeds the upper limit, a selection of exactly $\lfloor \lfloor \tau n \rfloor / q \rfloor$ elitist solutions of this cluster is made to be inserted in the population. This selection procedure attempts to maximize the distance between each of these solutions. It follows the exact same procedure as the clustering technique, by first selecting as a starting point a solution that is the maximum in a randomly chosen dimension. New solutions are then selected until the desired size has been reached, by choosing the most distant nearest neighbor, i.e., the solution that has the maximum minimal distance, to the set of previously selected solutions.

During the phase of generating new individuals, each cluster $\boldsymbol{C}_j$ samples $\lfloor n/q \rfloor - n^{\text{elitist}}_{\boldsymbol{C}_j}$ new individuals from its own probability distribution $\mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{C}_j}, \boldsymbol{\Sigma}_{\boldsymbol{C}_j}\right)$, applying AMS only if the individual is among the first $\frac{\tau n}{2q}$ individuals to be sampled by cluster $\boldsymbol{C}_j$ during this generation. After a full set of new individuals has been sampled, the distribution multiplier of each cluster is updated. Newly generated individuals from one cluster could however be drifting away from the cluster they were generated by and be located in the space nearer to a different cluster. Therefore an individual is first redistributed to its closest cluster before checking for improvements. Finally, the distribution multiplier $c^{\text{Multiplier}}_{\boldsymbol{C}_j}$ of each cluster $\boldsymbol{C}_j$ is updated using the SDR mechanism based on the improvements of all solutions that were redistributed to cluster $\boldsymbol{C}_j$.

Because solutions are redistributed to determine the improvements of each cluster, this can cause a large imbalance in the cluster sizes. These clusters are therefore not directly re-used in the following generation, but the clustering procedure is completely restarted using the new selection of individuals. We do however require the mean $\mu_{\boldsymbol{C}_j}$ and the distribution multiplier $c^{\text{Multiplier}}_{\boldsymbol{C}_j}$ of each cluster's previous generation, because these are required to perform AMS and AVS, respectively. Each new cluster is therefore paired with the most appropriate cluster of the previous generation, minimizing the total distance between clusters. The following procedure is performed to consistently perform a decent pairing of clusters while avoiding an exponential increase in computation time. First a random starting cluster is selected from the two clusters in the current generation that are the farthest apart. The starting cluster and its $r = 6$ nearest neighbors are then selected to be paired with a set of clusters in the previous generation. The set of clusters in the previous generation contains the cluster closest to the starting cluster and also its r closest neighbors. Subsequently each of the the $r+1$ clusters in the current generation are paired with one of the $r+1$ clusters in the previous generation, by considering all $(r+1)!$ possible cluster pairings and choosing the one that minimizes the total distance between pairs of clusters. This cluster registration procedure repeats by registering $r+1$ clusters until each cluster has been paired with a cluster in the previous generation.

Multi-Objective GOMEA

A multi-objective adaptation of the discrete GOMEA, MO-GOMEA, was introduced in [28]. MO-GOMEA uses an elitist archive to keep track of non-dominated solutions, and uses a k-leader-means clustering procedure that is also used in MAMaLGaM. However, MO-GOMEA applies this clustering procedure to the entire population instead of just to the selection. For each cluster, a separate FOS linkage model is learned and used for recombination. If recombination is applied to a solution that is not assigned to a cluster, the linkage model of the solution's closest cluster is used. For any solution that is assigned to multiple clusters, ties are broken randomly. The recombination operation of MO-GOMEA only accepts modifications of a solution if one of three criteria is met, either when the modified solution dominates the non-modified solution, when the modified solution has the same objective values as the non-modified solution, or when the modified solution is not dominated by any solution in the elitist archive. Finally, during the forced improvements phase of MO-GOMEA a donor is now randomly selected from the elitist archive. For each recombination with a single FOS element, a new donor is selected. When the forced improvement phase results in a solution that dominates the pre-existing solution, this solution is accepted and the forced improvements phase is terminated.

Multi-Objective Parameter-less scheme

When the parameter-less scheme introduced in Section 2.1.4 is applied to a multi-objective EA, the number of clusters is increased by one for each new population, starting from a predefined base number of clusters [28]. For multi-objective EAs, no clear criterion for termination of relatively small populations is known. We decide to terminate populations based on the number of solutions that they contribute to the approximation set, which is the set of non-dominated solutions in the combined set of all populations and the elitist archive. Any population for which each individual is dominated by a solution in the approximation set, if each smaller population is also completely dominated, is terminated. This is quite a liberal termination criterion, but it reasonably prevents the smallest populations from performing an exponentially growing number of generations.

3.2. The Multi-Objective Real-Valued GOMEA

We now introduce the multi-objective real-valued GOMEA, which uses the same multi-objective framework that was used by MAMaLGaM previously. Pseudo-code for this algorithm is displayed in Algorithm 12. Multi-objective GOMEA keeps track of a set of clusters that each aim to approach a different part of the Pareto front. Each cluster again has a separate model, but in contrast to MAMaLGaM where every cluster estimates a single multi-variate normal probability distribution over all variables, GOMEA estimates a probability distribution for each of its FOS elements. Also the distribution multiplier $c^{\text{Multiplier}}_{\boldsymbol{C}_j}$ of each cluster $\boldsymbol{C}_j$ is separated into a multiplier $c^{\text{Multiplier}}_{(\boldsymbol{F}_i,\boldsymbol{C}_j)}$ for each FOS element $\boldsymbol{F}_i \in \boldsymbol{F}$.

```
Function MAMaLGaM:
    $\boldsymbol{P} \leftarrow$ Uniformly generated initial population
    $c^{\text{Multiplier}}_{\boldsymbol{C}_j}(0 \le j \le q-1) \leftarrow 1$
    $\text{NIS} \leftarrow 0$
    $t \leftarrow 0$
    do
        $\boldsymbol{S} \leftarrow \tau n$ least dominated individuals in $\boldsymbol{P}$
        Perform clustering procedure, assigning each $\boldsymbol{s} \in \boldsymbol{S}$ to one of $q$ clusters $\boldsymbol{C}_j \in \boldsymbol{C}(t)$
        Register clusters such that the distance between $\boldsymbol{C}_j(t-1)$ and $\boldsymbol{C}_j(t)$ is minimized
        for $\boldsymbol{C}_j \in \boldsymbol{C}(t)$ do
            $\boldsymbol{S}_{\boldsymbol{C}_j} \leftarrow \boldsymbol{S} \cap \boldsymbol{C}_j$
            Estimate distributions $\mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{C}_j}(t), \boldsymbol{\Sigma}_{\boldsymbol{C}_j}(t))$ with maximum-likelihood based on $\boldsymbol{S}_{\boldsymbol{C}_j}$
            $\boldsymbol{\mu}^{\text{Shift}}_{\boldsymbol{C}_j}(t) \leftarrow \boldsymbol{\mu}_{\boldsymbol{C}_j}(t) - \boldsymbol{\mu}_{\boldsymbol{C}_j}(t-1)$
            $\boldsymbol{\mu}_{\boldsymbol{C}_j} \leftarrow \boldsymbol{\mu}_{\boldsymbol{C}_j}(t)$
            $\boldsymbol{\Sigma}_{\boldsymbol{C}_j} \leftarrow c^{\text{Multiplier}}_{\boldsymbol{C}_j} \boldsymbol{\Sigma}_{\boldsymbol{C}_j}(t)$
            $\boldsymbol{L}_{\boldsymbol{C}_j} \boldsymbol{L}^*_{\boldsymbol{C}_j} \leftarrow$ Cholesky decomposition of $\boldsymbol{\Sigma}_{\boldsymbol{C}_j}$
            Copy $n^{\text{elitist}}_{\boldsymbol{C}_j} \le \lfloor \lfloor \tau n \rfloor / q \rfloor$ elitist solutions closest to $\boldsymbol{C}_j$ to $\boldsymbol{P}$
        end
        for $\boldsymbol{C}_j \in \boldsymbol{C}(t)$ do
            $\boldsymbol{O}_{\boldsymbol{C}_j} \leftarrow \lfloor n/q \rfloor - n^{\text{elitist}}_{\boldsymbol{C}_j}$ samples from $\mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{C}_j}, \boldsymbol{\Sigma}_{\boldsymbol{C}_j}\right) = \boldsymbol{\mu}_{\boldsymbol{C}_j} + \boldsymbol{L}_{\boldsymbol{C}_j} \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$
            for $\alpha^{\text{AMS}}\left(\lfloor n/q \rfloor - n^{\text{elitist}}_{\boldsymbol{C}_j}\right)$ random solutions $\boldsymbol{x} \in \boldsymbol{O}_j$ do
                $\boldsymbol{x} \leftarrow \boldsymbol{x} + \delta^{\text{AMS}} c^{\text{Multiplier}}_{\boldsymbol{C}_j} \boldsymbol{\mu}^{\text{Shift}}_{\boldsymbol{C}_j}(t)$
            end
            $\boldsymbol{P} \leftarrow \boldsymbol{P} \cup \boldsymbol{O}_{\boldsymbol{C}_j}$
        end
        $\boldsymbol{E}(t) \leftarrow$ Non-dominated solutions in $\boldsymbol{P}$
        for $\boldsymbol{C}_j \in \boldsymbol{C}(t)$ do
            if any $\boldsymbol{x} \in \boldsymbol{O}_j$ is not Pareto dominated by any $\boldsymbol{E}_i \in \boldsymbol{E}(t-1)$ then
                $\text{NIS} \leftarrow 0$
                if $c^{\text{Multiplier}}_{\boldsymbol{C}_j} < 1$ then $c^{\text{Multiplier}}_{\boldsymbol{C}_j} \leftarrow 1$
                $\boldsymbol{x}^{\text{avg-imp}} \leftarrow$ average of all $\boldsymbol{x} \in \boldsymbol{O}_j$ for which there is no $\boldsymbol{E_i} \in \boldsymbol{E}$ s.t. $\boldsymbol{E}_i \succ \boldsymbol{x}$
                $\text{SDR} \leftarrow \max_{0 \le i \le \ell-1} \left\{ \left| \left( L^{-1}_{\boldsymbol{C}_j} (\boldsymbol{x}^{\text{avg-imp}} - \boldsymbol{\mu}_{\boldsymbol{C}_j}) \right)_i \right| \right\}$
                if $\text{SDR} > \theta^{\text{SDR}}$ then $c^{\text{Multiplier}}_{\boldsymbol{C}_j} \leftarrow \eta^{\text{INC}} c^{\text{Multiplier}}_{\boldsymbol{C}_j}$
            else
                if $c^{\text{Multiplier}}_{\boldsymbol{C}_j} \le 1$ then $\text{NIS} \leftarrow \text{NIS} + 1$
                if $\left(c^{\text{Multiplier}}_{\boldsymbol{C}_j} > 1\right)$ or $\left(\text{NIS} \ge \text{NIS}^{\text{MAX}}\right)$ then $c^{\text{Multiplier}}_{\boldsymbol{C}_j} \leftarrow \eta^{\text{DEC}} c^{\text{Multiplier}}_{\boldsymbol{C}_j}$
                if $\left(c^{\text{Multiplier}}_{\boldsymbol{C}_j} < 1\right)$ and $\left(\text{NIS} < \text{NIS}^{\text{MAX}}\right)$ then $c^{\text{Multiplier}}_{\boldsymbol{C}_j} \leftarrow 1$
            end
        end
        $t \leftarrow t + 1$
    while sufficiently good Pareto front not found, max. eval. not reached and each $c^{\text{Multiplier}}_{\boldsymbol{C}_{0 \ldots q-1}} \ge 10^{-10}$
end
```

Algorithm 11: MAMaLGaM

```
Function Multi-Objective GOMEA:
    $\boldsymbol{P} \leftarrow$ Uniformly generated initial population
    $\boldsymbol{E} \leftarrow$ All non-dominated solutions in $\boldsymbol{P}$
    $c^{\text{Multiplier}}_{(\boldsymbol{F}_i,\boldsymbol{C}_j)} \leftarrow 1 \quad (0 \le j \le q-1; 0 \le i \le |\boldsymbol{F}|-1)$
    $\text{NIS} \leftarrow t \leftarrow 0$
    while not terminated do
        $\boldsymbol{S} \leftarrow \tau n$ least dominated individuals in $\boldsymbol{P}$
        Perform selection clustering procedure, assigning each $\boldsymbol{S}_k \in \boldsymbol{S}$ to one of $q$ clusters $\boldsymbol{C}^{\text{S}}_j \in \boldsymbol{C}^{\text{S}}(t)$
        Register clusters such that the distance between $\boldsymbol{C}^{\text{S}}_j(t-1)$ and $\boldsymbol{C}^{\text{S}}_j(t)$ is minimized
        Perform population clustering procedure, assigning each $\boldsymbol{P}_i \in \boldsymbol{P}$ to one of $q$ clusters $\boldsymbol{C}^{\text{P}}_j \in \boldsymbol{C}^{\text{P}}$
        for $\boldsymbol{C}^{\text{S}}_j \in \boldsymbol{C}^{\text{S}}$ do
            for $\boldsymbol{F}_i \in \boldsymbol{F}$ do
                Estimate $\mathcal{N}\left(\boldsymbol{\mu}_{(\boldsymbol{F}_i,\boldsymbol{C}_j)}(t), \boldsymbol{\Sigma}_{(\boldsymbol{F}_i,\boldsymbol{C}_j)}(t)\right)$ with maximum-likelihood
                $\boldsymbol{\mu}^{\text{Shift}}_{(\boldsymbol{F}_i,\boldsymbol{C}_j)}(t) \leftarrow \boldsymbol{\mu}_{(\boldsymbol{F}_i,\boldsymbol{C}_j)}(t) - \boldsymbol{\mu}_{(\boldsymbol{F}_i,\boldsymbol{C}_j)}(t-1)$
                $\boldsymbol{\mu}_{(\boldsymbol{F}_i,\boldsymbol{C}_j)} \leftarrow \boldsymbol{\mu}_{(\boldsymbol{F}_i,\boldsymbol{C}_j)}(t)$
                $\boldsymbol{\Sigma}_{(\boldsymbol{F}_i,\boldsymbol{C}_j)} \leftarrow c^{\text{Multiplier}}_{(\boldsymbol{F}_i,\boldsymbol{C}_j)} \boldsymbol{\Sigma}_{(\boldsymbol{F}_i,\boldsymbol{C}_j)}(t)$
                $\boldsymbol{L}_{(\boldsymbol{F}_i,\boldsymbol{C}_j)} \boldsymbol{L}^{*}_{(\boldsymbol{F}_i,\boldsymbol{C}_j)} \leftarrow$ Cholesky decomposition of $\boldsymbol{\Sigma}_{(\boldsymbol{F}_i,\boldsymbol{C}_j)}$
            end
            Copy $n^{\text{elitist}}_{\boldsymbol{C}_j} \le \tau|\boldsymbol{C}^{\text{S}}_j|$ elitist solutions to $\boldsymbol{P}$
        end
        for $\boldsymbol{F}_i \in \boldsymbol{F}$ do
            for $\boldsymbol{C}^{\text{P}}_j \in \boldsymbol{C}^{\text{P}}$ do
                for $\boldsymbol{x} \in \boldsymbol{C}^{\text{P}}_j$ do
                    $\boldsymbol{b}_{\boldsymbol{F}_i} \leftarrow \boldsymbol{x}_{\boldsymbol{F}_i} \leftarrow (\boldsymbol{x}_u)$ for $u \in \boldsymbol{F}_i$
                    $\boldsymbol{b} \leftarrow \boldsymbol{x}$
                    $\boldsymbol{y}_{\boldsymbol{F}_i} \leftarrow \mathcal{N}\left(\boldsymbol{\mu}_{(\boldsymbol{F}_i,\boldsymbol{C}_j)}, \boldsymbol{\Sigma}_{(\boldsymbol{F}_i,\boldsymbol{C}_j)}\right)$
                    if $\boldsymbol{x} \in \{$the first $\lfloor \frac{1}{2}\tau|\boldsymbol{C}^{\text{P}}_j|\rfloor$ individuals in $\boldsymbol{C}^{\text{P}}_j\}$ then
                        $\boldsymbol{y}_{\boldsymbol{F}_i} \leftarrow \boldsymbol{y}_{\boldsymbol{F}_i} + \delta^{\text{AMS}} \boldsymbol{\mu}^{\text{Shift}}_{(\boldsymbol{F}_i,\boldsymbol{C}_j)}(t)$
                    end
                    $\boldsymbol{x} \leftarrow \boldsymbol{x} \odot \boldsymbol{y}_{\boldsymbol{F}_i}$
                    if not $(\boldsymbol{x} \succ \boldsymbol{b}$ or $\boldsymbol{e} \not\succ \boldsymbol{x}$ for all $\boldsymbol{e} \in \boldsymbol{E})$ then
                        $\boldsymbol{x} \leftarrow \boldsymbol{x} \odot \boldsymbol{b}_{\boldsymbol{F}_i}$
                    end
                end
                AdaptMultiplier$\left(c^{\text{Multiplier}}_{(\boldsymbol{F}_i,\boldsymbol{C}_j)}\right)$
                Add new non-dominated solutions to $\boldsymbol{E}$ and remove dominated solutions
            end
        end
        for $\boldsymbol{x} \in \boldsymbol{P}$ do
            if $\boldsymbol{x}$ has not improved this generation then $\boldsymbol{x} \leftarrow$ ForcedImprovements($\boldsymbol{x}$)
        end
        for $\boldsymbol{x} \in \{$the first $\frac{1}{2}\tau n$ individuals in $\boldsymbol{P}\}$ do
            $\boldsymbol{b} \leftarrow \boldsymbol{x}$
            $\boldsymbol{x} \leftarrow \boldsymbol{x} + \delta^{\text{AMS}} \boldsymbol{\mu}^{\text{Shift}}(t)$
            if not $(\boldsymbol{x} \succ \boldsymbol{b}$ or $\boldsymbol{e} \not\succ \boldsymbol{x}$ for all $\boldsymbol{e} \in \boldsymbol{E})$ then
                $\boldsymbol{x} \leftarrow \boldsymbol{b}$
            end
        end
        $t \leftarrow t+1$
    end
end
```

Algorithm 12: Multi-Objective GOMEA

3.2.1. Clustering

The main problem of applying the multi-objective framework to GOMEA arises due to the fact that the framework only assigns individuals in the selection to a cluster. These individuals can even be assigned to more than one or even no cluster at all. The remaining individuals in the population are never assigned to a cluster. This becomes a problem, because GOMEA iteratively improves individuals instead of completely generating new individuals. A newly generated partial solution should be generated by the same cluster as the individual it is inserted in, otherwise separate parts of the individual are aimed at different sections of the Pareto front. Assigning an individual to multiple clusters must therefore be avoided as well, because newly generated partial solutions must be generated based on the model of a single cluster. Having an individual that is assigned to multiple clusters would make it unclear from which cluster partial solutions for this solution must be generated. Due to the fact that individuals cannot be assigned to multiple clusters, the current clustering procedure cannot simply be extended by enlarging the cluster size, because this will still cause some individuals to be assigned to more than one or no clusters. Instead a new technique is employed to distribute every individual in the population to exactly one cluster. To not interfere with the estimation of the probability distributions, for which it should still be possible to have overlapping clusters to encourage a smooth spreading of the Pareto front, we make the two clustering procedures completely independent. The newly introduced clustering procedure will assign any individual in the population to exactly one so-called population cluster. In addition to a population cluster, each individual in the selection will also be assigned to zero or more so-called selection clusters. The population cluster of an individual is exclusively used to determine which cluster's model should be used to sample a new partial solution that can be inserted into this individual. In contrast, all solutions assigned to a selection cluster are exclusively used to estimate the normal probability distribution of this cluster.

The procedure of creating population clusters first requires the distance between each individual in the population and each selection cluster, which was already computed in order to create the selection clusters. Subsequently, the clustering procedure performs $\frac{2\tau n}{q}$ rounds, assigning the closest non-assigned individual to each cluster during each round. The single-objective clusters get to select their closest individual first, followed by each of the remaining clusters assigning their closest individual to themselves. After these rounds, every cluster will have a size of exactly $\frac{2\tau n}{q}$, after which any remaining non-assigned solution is assigned to its closest cluster. The lower bound on the size $n^{\boldsymbol{C}_j}$ of a cluster $\boldsymbol{C}_j$ is required to consistently find improvements. A very small cluster can struggle to find improvements, similar to a too small population for a single-objective optimization problem, causing the distribution multiplier to be increased or decreased erroneously, possibly leading to slower or premature convergence.

3.2.2. Copying Elitist Solutions

Before any new solutions are generated, first a set of individuals from the elitist archive are copied into the population, in a similar way to that of MAMaLGaM. However, copying elitist solutions into the population means that some existing individuals must be replaced. For MAMaLGaM it is irrelevant which individual is replaced, because any newly generated solution replaces the existing individual without being influenced by it. Instead, because GOMEA improves existing individuals, the quality of an existing individual influences how it is modified by the newly generated partial solutions. Hence, when a part of a solution is not replaced by a newly generated partial solution, it retains its current state. An arbitrary individual $\boldsymbol{x}$ that has a better objective value than an arbitrary individual $\boldsymbol{y}$, is expected to still have a better objective value after recombination, because every partial solution of $\boldsymbol{x}$ that was not replaced, has on average a better objective value than a partial solution of $\boldsymbol{y}$ that was not replaced. Therefore it should be beneficial to replace the most low-quality individuals of clusters by elitist solutions, where the quality of an individual is in this sense defined by its rank. The process of copying elitist solutions then proceeds by copying elitist solutions into the population, each replacing the lowest ranked individual in the nearest population cluster. However, only up to $\tau|\boldsymbol{C}_j^{\mathrm{P}}|$ elitist solution may be copied into any cluster $\boldsymbol{C}_j^{\mathrm{P}}$. If a larger number of elitist solutions would be copied into a cluster, a selection of these solutions is made in an identical manner as previously stated. Starting from a random point that is the maximum in a randomly chosen dimension, more points are added by iteratively selecting the point that has the largest minimal distance to the set of already selected points, until the desired size $\tau|\boldsymbol{C}_j^{\mathrm{P}}|$ has been reached. As an extra criterion, an elitist solution that would be replacing an individual in a single-objective cluster only does so when its objective value in the respective objective is better than that of the existing individual. If not, the existing individual is maintained in the population, but no additional elitist solution is selected to be inserted into this cluster.

3.2.3. Generating New Solutions

After the above procedure of copying elitist solutions into the population, new partial solutions are generated to improve the existing population. This is performed on every individual in the population, even those elitist solutions that have just been copied into the population. First, each population cluster $\boldsymbol{C}_i^{\mathrm{P}}$ is iterated over, followed by the iteration over each FOS element $\boldsymbol{F}_i \in \boldsymbol{F}$ and the insertion of new partial solutions generated by $\boldsymbol{F}_i$ into each individual $\boldsymbol{x} \in \boldsymbol{C}_j$. If an individual $\boldsymbol{x}$ is among the first $\frac{\tau|\boldsymbol{C}_j^{\mathrm{P}}|}{2}$ individuals in population cluster $\boldsymbol{C}_j$, AMS is applied to it, adding $\delta^{\mathrm{AMS}} c^{\mathrm{Multiplier}}_{(F_i,C_j)} \boldsymbol{\mu}^{\mathrm{Shift}}_u$ to each variable $\boldsymbol{x}_u$ for each $u \in \boldsymbol{F}_i$. A new partial solution $\boldsymbol{y}_{\boldsymbol{F}_i}$ is only accepted as a modification of an existing solution $\boldsymbol{x}$ if inserting the values of $\boldsymbol{y}_{\boldsymbol{F}_i}$ into $\boldsymbol{x}$, i.e., $\boldsymbol{x} = \boldsymbol{x} \odot \boldsymbol{y}_{\boldsymbol{F}_j}$, is deemed an improvement over $\boldsymbol{x}$. For this purpose, the recombination with solution $\boldsymbol{y}_{\boldsymbol{F}_j}$ is regarded as an improvement if it dominates $\boldsymbol{x}$ or if it can be added into the elitist archive. Additionally, an individual in a single-objective cluster is regarded as an improvement if the newly combined solution $\boldsymbol{y}$ has a better objective value in the respective objective of this single-objective cluster. After a FOS element $\boldsymbol{F}_i$ has attempted to improve all solutions in cluster $\boldsymbol{C}_j$, $c^{\mathrm{Multiplier}}_{(\boldsymbol{F}_i,\boldsymbol{C}_j)}$ is updated to reflect the improvements that $\boldsymbol{F}_i$ has achieved on individuals in cluster $\boldsymbol{C}_j$. Subsequently, the elitist archive is updated, to avoid making it more difficult for the most recently generated solutions to be inserted into the elitist archive.

3.2.4. AVS and SDR

The AVS-SDR technique that is responsible for updating the distribution multipliers now manages a multiplier $c^{\mathrm{Multiplier}}_{(\boldsymbol{F}_i,\boldsymbol{C}_j)}$ for each pair $(\boldsymbol{F}_i, \boldsymbol{C}_j)$ in the Cartesian product of the set of clusters and FOS. For each variable in $\boldsymbol{F}_i$, the mean is calculated of all solutions in $\boldsymbol{C}_j$ that are not dominated by the current elitist archive, resulting in $\boldsymbol{x}^{\text{avg-imp}}$. Identical to the single-objective SDR, the means of each variable are subtracted from $\boldsymbol{x}^{\text{avg-imp}}$, after which it is multiplied by $\boldsymbol{L}^{-1}_{(\boldsymbol{F}_i,\boldsymbol{C}_j)}$, which results in a vector of SDR values. Only if at least one element of the SDR vector is larger than 1, $c^{\mathrm{Multiplier}}_{(\boldsymbol{F}_i,\boldsymbol{C}_j)}$ is multiplied by a factor η^{INC}.

3.2.5. Second-round AMS

Identical to single-objective GOMEA, the multi-objective GOMEA also employs a second round of the AMS procedure to improve efficiency on problems involving dependencies that are not captured by the dependency structure. This round is started after the complete sampling procedure, applying AMS to a fraction $\frac{\tau|\boldsymbol{C}_j^{\mathrm{P}}|}{2}$ of the individuals in each population cluster $\boldsymbol{C}_j$. To each variable $\boldsymbol{x}_u$ of each individual $\boldsymbol{x}$ that is subject to this round of AMS, $\delta^{\mathrm{AMS}} \boldsymbol{\mu}^{\mathrm{Shift}}_u$ is added. Due to AMS being applied to all variables instead of only those in a certain FOS element, no distribution multiplier is applied to the magnitude of the AMS, because each distribution multiplier is now tied to its respective FOS element. However, only if the application of this AMS to an individual leads to an improvement of this individual, this change is accepted.

3.2.6. Forced Improvements

At the end of each generation, the forced improvements procedure is applied to each individual $\boldsymbol{x}$ that has not changed during the current generation. From the elitist archive the solution closest to $\boldsymbol{x}$ is then selected as a donor. This donor is then used for a forced improvements procedure identical to that of the single-objective GOMEA, iterating over each FOS element $\boldsymbol{F}_i \in \boldsymbol{F}$ and replacing the variables in $\boldsymbol{F}$ with a weighted average of those in the donor and those in $\boldsymbol{x}$. Only replacements that can be entered into the elitist archive, or which dominate $\boldsymbol{x}$ are accepted, at which point the forced improvements procedure terminates. If no improvement is still found when the weight α of the variables of $\boldsymbol{x}$ has reached a value below 0.05, the individual $\boldsymbol{x}$ is completely overwritten by its donor.

3.2.7. Termination

Identical to the single-objective GOMEA, termination conditions can be set in terms of a maximum number of seconds or a maximum number of evaluations. A VTR in terms of the $D_{\boldsymbol{P}_f \leftarrow \boldsymbol{S}}$, described in Section 5.1.2 that defines the distance of an arbitrary Pareto front to the optimal Pareto front, can be used to terminate GOMEA when a desired Pareto front quality has been achieved. The single-objective termination termination condition based on the fitness variance is no longer used. Finally, when each distribution multiplier $c^{\mathrm{Multiplier}}_{(\boldsymbol{F}_i,\boldsymbol{C}_j)}$ has reached a value below 10^{-10}, GOMEA is believed to have converged, and is terminated.

4

Designing a Grey-Box Version of Multi-Objective Deformable Image Registration

To greatly improve the efficiency of GOMEA for the multi-objective deformable image registration problem, we design a grey-box version of this problem, which will allow for partial evaluations to be performed. In this chapter, deformable image registration will first be introduced along with background information and related work. This is followed by the design of the grey-box adaptation of deformable image registration, and problem-specific adaptations of GOMEA.

4.1. Background

The deformable image registration problem concerns finding the correct transformation of a source image to a target image. Several transformation models are can be used for this problem, and a frequently used model is the so-called B-spline transformation model [38]. This model defines a global transformation model in terms of a rigid matrix transformation, and a local transformation model in terms of an underlying mesh of points. Changing the location of control points in this mesh affects the transformation of nearby points. The `elastix` [25] toolbox, which can be considered the current state of the art, uses this B-spline transformation model. `elastix` is a toolbox for intensity-based medical image registration that consists of a collection of commonly used algorithms. However, multiple objectives of interest exist and `elastix` condenses all these objectives into a single objective by calculating a weighted sum of all objective values. This approach requires manual problem-specific tuning of the weight of each objective, which is uninsightful and time-consuming in practice.

For the recently introduced multi-objective approach to deformable image registration, iMAMaLGaM [2, 4, 7] can be considered state of the art, basically having no competitors. iMAMaLGaM previously used a transformation model based on a pair of triangulated grids. This transformation model is created by overlaying the source image with a triangulated grid, and overlaying the target image with a triangulated grid with an identical topology. Due to the grids having an identical topology, the area within each triangle in one grid can be mapped to the area of the triangle with the same initial position in the opposing grid. Deforming the source image can then be done by moving a point of the source grid, changing the shape of the adjacent triangles, and therefore how the image contents in these triangles are mapped to the opposing grid. An advantage of this approach is that it can find transformations with disappearing structures or other large deformations, which would be hard or impossible to find with a B-spline transformation model. Some structure contained in the image can for example be enlarged by increasing the size of its encompassing triangle, or it can be completely removed by bringing together the vertices defining this triangle. A downside of the triangulated grid transformation model is that transformations defined by such a model are non-smooth. Higher-order meshes are required to approximate smoothness.

Recent research has shown that also using a deformable grid to overlay the target image results in more flexibility, allowing the model to be able to deal with larger differences between the target and the source image, e.g. (dis)appearing structures [3]. In Figures 4.1a and 4.1b the source and target images of a commonly used problem instance are displayed. These images both display a 2D slice taken from breast MRI scans, but the source image was acquired in a prone position and the target image was acquired in a supine position. In this example, a 5 × 5 grid is applied to the images, but a higher-resolution grid can be used on more difficult problems with very refined structures. In the initial state of the grids, which are displayed in Figures 4.1a and 4.1b, each point of the source grid is mapped to the point in the target grid with the same relative position. The same principle applies to all triangles contained in the grid, mapping every triangle in the source grid to one triangle in the target grid. Possible grids at a later stage of the optimization process are displayed in Figures 4.1c and 4.1d. Additionally, Figure 4.1d is overlaid with the source image deformed according to the displayed pair of grids.

In the dual-dynamic grid transformation model, the position of a transformed point is defined in terms of a transformation that maps a pixel p_s in the source image to the corresponding position p_s^c in the target image based on the pair of triangulated grids. Similarly, a transformation can be applied to a pixel p_t in the target image to map its position in the target grid to the corresponding position p_t^c in the source grid. Calculating the position of a transformed point is done by applying the so-called barycentric coordinate system. Barycentric coordinates define a location inside a triangle by assigning weights $\lambda_1, \lambda_2, \lambda_3 \in [0,1]$ to each of the triangle's vertices, with $\lambda_1 + \lambda_2 + \lambda_3 = 1$. From a set of barycentric weights $(\lambda_1, \lambda_2, \lambda_3)$ and three vertices $(x_1, y_1), (x_2, y_2)$ and (x_3, y_3) of a triangle, the Cartesian coordinates of a transformed point (x_t, y_t) can be determined by calculating the weighted sum over the triangle's vertices in the following way:

$$x_t = \lambda_1 x_1 + \lambda_2 x_2 + \lambda_3 x_3$$
$$y_t = \lambda_1 y_1 + \lambda_2 y_2 + \lambda_3 y_3$$

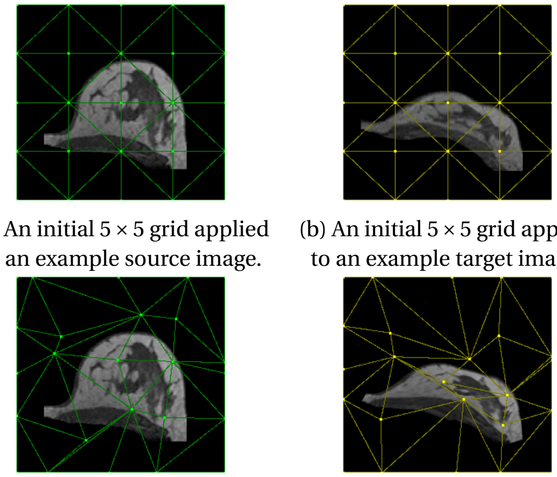
Conversion from a Cartesian coordinate (x, y) to the barycentric system requires the calculation of the three weights $(\lambda_1, \lambda_2, \lambda_3)$ as follows:

$$\lambda_1 = \frac{(y_2 - y_3)(x - x_3) + (x_3 - x_2)(y - y_3)}{(y_2 - y_3)(x_1 - x_3) + (x_3 - x_2)(y_1 - y_3)}$$
$$\lambda_2 = \frac{(y_3 - y_1)(x - x_3) + (x_1 - x_3)(y - y_3)}{(y_2 - y_3)(x_1 - x_3) + (x_3 - x_2)(y_1 - y_3)}$$
$$\lambda_3 = 1 - \lambda_1 - \lambda_2$$

The goal of deformable image registration is to find the best match possible between the deformed source image and the target image, but also to minimize the effort required to perform the deformation. These two goals are conflicting, because a deformation that results in a better match often requires more effort to perform. Therefore, we ultimately want to find a solution that has an optimal trade-off between the match quality and the required effort to perform the deformation. Any solution is an optimal trade-off when there exists no other solution that achieves a better match quality with less deformation effort. Therefore, no single optimal solution exists, but instead there exists a Pareto set of optimal solutions for which no solution is strictly better than any other solution, i.e., the problem is by nature multi-objective. However, a deformation resulting in the best possible image similarity is not necessarily the most clinically desirable solution, because such a deformation often requires strong deformations that are not physically plausible. This is often caused by overfitting to noise in medical images.

In many existing models and algorithms for deformable image registration the trade-off between the two objectives is dealt with by defining a single objective that is a linear combination of both objectives, but this approach requires manual tuning of the weight of each objective for each new problem instance, i.e., each patient case. Such weight tuning in deformable image registration is non-trivial, and it was shown that this approach is less successful than a multi-objective approach on problems with increasing difficulty [36]. This demonstrates the inherent strength of a pure multi-objective approach, because such an approach does not require the tuning of any weights and results in a range of solutions with different deformation magnitudes [2, 7]. Instead, multi-objective optimization can be applied to obtain a Pareto set of solutions that can be insightfully post-processed to find the most interesting solution for the selected application.

From a multi-objective perspective, multiple objectives of interest may now be considered. One objective requires the maximization of the similarity between the transformed source image and the target image, but it will instead be defined as the minimization of the dissimilarity between these images. To define



(a) An initial 5 × 5 grid applied to an example source image.

(b) An initial 5 × 5 grid applied to an example target image.

(c) A deformed 5 × 5 grid applied to an example source image.

(d) A deformed 5 × 5 grid overlaying the deformed source image.

Figure 4.1: Illustrations of triangulated grids, defining the deformation of the source image.

the dissimilarity between two images, a simple measure is used which computes the squared intensity difference between a pair of corresponding pixels. This dissimilarity measure has limited real-world applicability when, for example, the source and target image have different intensities, but a more refined measure, such as the cross-correlation measure, could potentially be used when images of different modalities are considered. The dissimilarity is computed for each pair of pixels and is then normalized to account for the total number of pixels. Let I_s be the set of pixels defining the source image, with $I_s(p_s)$ the intensity of the pixel $p_s = (p_x, p_y)$. The target image I_t is defined in an identical way, along with some transformation that maps the location of any point in the source image to the corresponding point in the target image. Then the dissimilarity objective $f_{\text{dissimilarity}}$ is defined as follows:

$$f_{\text{dissimilarity}} = \frac{1}{|I_s| + |I_t|}\left[\sum_{p_s \in I_s}\left(I_s(p_s) - I_t(p_s^c)\right)^2 + \sum_{p_t \in I_t}\left(I_t(p_t) - I_s(p_t^c)\right)^2\right]$$

A second objective requires the minimization of the effort required to perform the transformation and will be referred to as the deformation magnitude objective. Since the application of this deformable image registration model is mostly focused on medical images, such as CT scans, minimizing the deformation magnitude will lead to more natural looking images. Ignoring this objective would instead lead to heavily distorted images that would not be very useful in practice. The deformation magnitude is calculated based on the deformation vector field, as described in [4]. The deformation magnitude for one point p can be calculated by considering the set of vectors $\boldsymbol{V}(p)$ between p and a set of three very nearby points to p. These points are obtained by adding $\varepsilon = 10^{-10}$ to the barycentric weights of p. The deformation magnitude is then calculated as the magnitude by which the lengths of these vectors are changed by the application of the transformation. This gives an indication of how much the space around p is being deformed. For this, a transformation can be applied to a vector $\boldsymbol{v}$ between two arbitrary points p_a and p_b to obtain $\boldsymbol{v}^c$, defined as the vector between the transformed points p_a^c and p_b^c. The average deformation vector length for a point p is then computed as the average difference in length caused by the transformation for all vectors in $\boldsymbol{V}(p)$. Finally the complete deformation objective is computed by summing the squares of the average deformation vectors of all pixels in I_s and in I_t, which is defined as follows:

$$f_{\text{deformation}} = \frac{1}{|I_s| + |I_t|}\left[\sum_{p_s \in I_s}\left[\frac{1}{|\boldsymbol{V}(p_s)|}\sum_{\boldsymbol{v}_s \in \boldsymbol{V}(p_s)}\left|\|\boldsymbol{v}_s\| - \|\boldsymbol{v}_s^c\|\right|\right]^2 + \sum_{p_t \in I_t}\left[\frac{1}{|\boldsymbol{V}(p_t)|}\sum_{\boldsymbol{v}_t \in \boldsymbol{V}(p_t)}\left|\|\boldsymbol{v}_t\| - \|\boldsymbol{v}_t^c\|\right|\right]^2\right]$$

A third objective, first introduced in [4], is used to guide a solution towards a predefined set of guidance image pairs, one for the source and one for the target image. Such guidance images can contain sets of points or singular points to represent contours or points of interest. This third objective, the guidance objective, is defined as the dissimilarity between the transformed source guidance image and the target guidance image. Therefore, the guidance objective is optimal when the transformed source guidance image exactly matches the target guidance image. The dissimilarity of the two guidance images is computed by using the measure that finds the sum of Euclidean distances of each point in the transformed source guidance image to its closest guidance point in the target image. Similarly, the closest distance to some point in the source guidance image is computed for each point in the inversely transformed target guidance image, after which the total sum distances is normalized to the number of guidance points. Finally, the sum over all guidance image pairs is used as the value for the guidance objective. Let $\boldsymbol{G}$ be a set of guidance image pairs, G_s some source guidance image and G_t some target guidance image. Then, the guidance objective is formally defined as:

$$f_{\text{guidance}} = \sum_{(\boldsymbol{G}_s,\boldsymbol{G}_t)\in\boldsymbol{G}} \frac{1}{|\boldsymbol{G}_s| + |\boldsymbol{G}_t|} \left[\sum_{p_s\in\boldsymbol{G}_s} \min_{p_t\in\boldsymbol{G}_t} \{d(p_s^c, p_t)\} + \sum_{p_t\in\boldsymbol{G}_t} \min_{p_s\in\boldsymbol{G}_s} \{d(p_t^c, p_s)\} \right]$$

Not every possible instance of a pair of grids is however a valid transformation. Moving a vertex within the perimeter of a different triangle will cause the areas of two triangles to overlap, meaning that the transformation of the points in this overlapped area is ambiguous. Transforming it relative to one triangle will result in a different position than transforming it relative to the other triangle. Therefore any triangulated grid where any point is located inside a different triangle is considered infeasible. Equivalent to this condition is the condition of having intersecting edges. Therefore the number of intersecting edges is used as the measure of infeasibility of a solution. Only when there exist no intersecting edges in both the source and the target grid the triangulation defines a feasible solution.

4.2. Grey-Box Multi-Objective Deformable Image Registration

We now describe the design of the grey-box version of multi-objective deformable image registration, which will allow for partial evaluations to be performed. This will potentially increase the efficiency of GOMEA by a substantial amount. As introduced in Section 4.1, solving the deformable image registration problem requires at least the similarity and deformation objectives while the optional guidance makes it much easier to obtain a high quality solution. With AMaLGaM all variables of any new individual are generated completely anew, so any new individual must be completely evaluated to obtain its objective values. In contrast to AMaLGaM, GOMEA iteratively improves existing solutions and only replaces a limited number of variables while keeping the rest of the individual intact. Therefore, completely re-evaluating an individual after this sampling procedure is very inefficient and requires a lot of unnecessary work. Instead defining a method of partially re-evaluating a solution based on the changes that took place should greatly improve the efficiency of GOMEA on the deformable image registration problem. This section aims to define a certain substructure for which all three objective functions can be defined independently from all other substructures. Because the deformation is defined by the coordinates of the points that compose the triangulated grid, any modification of a grid point leads to a change in deformation in any triangle that includes this grid point. Therefore, any pixel in this triangle could have a different intensity value than before, requiring the similarity objective to be recalculated for this entire triangle. Because it is unavoidable to re-evaluate the objectives in a triangle for which at least one point has changed, it makes sense to define one triangle as the smallest substructure that can be evaluated. A requirement for this is that all objective values of any triangle can be calculated independently of any other triangle.

Partially re-evaluating substructures of a solution also requires the redefinition of how an image is deformed, i.e. how the coordinates of a transformed pixel are calculated. Previously, a deformation field was calculated for this entire image, which defines for every pixel the x and y translation that needs to be applied to transform the respective pixel. Calculating the deformation field was done by randomly sampling a set of barycentric coordinates inside every triangle. For each of these coordinates, the displacement is calculated by converting them from barycentric to Cartesian. The displacement of a point is then distributed among all pixels neighboring this point according to a weight determined by the distance of the pixel to the transformed point, finally leading to a two dimensional matrix containing the x and y displacements of every pixel. When only transforming a small selection of pixels at a time, building a deformation field matrix for the entire image is not very efficient. Therefore, the transformation of a pixel is now directly defined by its barycentric coordinates, at the cost of losing the smoothing effect caused by the influence of each pixel on its neighbors' deformation field.

4.2.1. Objective Functions

All objective functions that were first introduced in Section 4.1 are now redefined as a function over all triangles in the deformation grids in order to allow for partial evaluations. We define the set of triangles in the source grid as Δ_s and the set of triangles in the target grid as Δ_t, with the complete set of triangles being $\Delta = \Delta_s \cup \Delta_t$. After changing the objective values of one triangle δ the objective value $f(\Delta)$ over all triangles can be recalculated through the following means, based on the previous total objective value $f_{\text{prev}}(\Delta)$, the previous objective value $f_{\text{prev}}(\delta)$ of triangle δ and the newly evaluated objective value $f(\delta)$:

$$f(\Delta) = f_{\text{prev}}(\Delta) - f_{\text{prev}}(\delta) + f(\delta)$$

Calculating objective values first requires us to define how any point is transformed. This transformation function is defined by the complete set of variables containing the coordinate of each source and target grid point. Applying the transformation to an ordinary point p_s in the source image transforms this point based on its barycentric coordinates relative to the triangle δ_s it is contained in. These barycentric coordinates are applied as weights to the vertices of the corresponding target triangle δ_t to find the transformed point p_s^c.

Calculating the dissimilarity of a triangle and its counterpart is done by comparing the intensity of all pixels inside one triangle with the intensities of these pixels after applying the transformation. For this purpose, a pixel is considered inside a triangle when its center is, with all pixels inside a triangle δ denoted as $px(\delta)$. This procedure is performed symmetrically for the triangle in the source grid, as well as for the corresponding triangle in the target grid. Only iterating over all the pixels in either one of the triangles could lead to inaccuracies when a triangle of a very small size is deformed into a very large triangle, because the large triangle would then only be evaluated based on a very small number of pixels. For each pixel p that is considered, the squared difference in intensity $I(p)$ is calculated between this pixel and its transformed counterpart. Since the transformed coordinates of a pixel unavoidably are real-valued numbers, we apply a bi-linear interpolation to calculate its intensity. This calculates a weighted sum over the four coordinates obtained by flooring and ceiling each of the point's coordinates. Finally, the total objective of a triangle is the sum of squared difference of intensities over all pixels inside this triangle normalized for the total number of pixels. The total dissimilarity objective can formally be defined as:

$$f_{\text{dissimilarity}} = \frac{1}{|I_s| + |I_t|}\left[\sum_{\delta_s \in \Delta_s}\left[\sum_{p_s \in px(\delta_s)} \left(I_s(p_s) - I_t(p_s^c)\right)^2\right] + \sum_{\delta_t \in \Delta_t}\left[\sum_{p_t \in px(\delta_t)} \left(I_t(p_t) - I_s(p_t^c)\right)^2\right]\right]$$

The deformation magnitude was previously based on the derivative of deformation [4], but further in the past it had been calculated based on differences in edge lengths [2]. Both approaches have been experimented with and the approach based on edge lengths led to good solutions in significantly less time, because evaluating one triangle with this approach requires much less time than evaluating one triangle with the opposing approach. Therefore, the deformation magnitude is defined in terms of edge deformations by comparing the length of each edge in the source grid to the lengths of the corresponding edges in the target grid. Each of these length differences is squared, because of the underlying idea of Hooke's law. The final deformation magnitude of one triangle δ is the sum of squared edge length differences normalized over the total number of edges, which is formally defined as:

$$f_{\text{deformation}} = \frac{1}{3\left(|\Delta_s| + |\Delta_t|\right)}\left[\sum_{\delta_s \in \Delta_s}\left[\sum_{e_s \in \text{edges}(\delta_s)} \left(\|e_s\| - \|e_s^c\|\right)^2\right] + \sum_{\delta_t \in \Delta_t}\left[\sum_{e_t \in \text{edges}(\delta_t)} \left(\|e_t\| - \|e_t^c\|\right)^2\right]\right]$$

For the third and final objective, the guidance objective, the guidance information is again supplied by means of a set G of pairs of guidance contours (G_s, G_t) of the source and the target image. Instead of evaluating the entire guidance objective at once, it is now required to calculate it for a single triangle. To properly evaluate this objective, sampling a random set of points will not be sufficient, because very few of these points will actually land on one of the guidance contours. Instead, all pixels exactly on one of the contours must be evaluated. When evaluating a triangle, this is done by iterating over all pixels inside this triangle and checking whether the pixel lies on one of the guidance contours. If it does, the minimum distance to the opposing guidance contour is calculated and added to the objective value. The set of points inside δ that lie on a contour G are then denoted by $G(\delta)$, allowing the formal definition of the guidance objective in the following way:

$$f_{\text{guidance}} = \sum_{\delta_s \in \Delta_s} \left[\sum_{(G_s,G_t)\in G} \frac{1}{|G_s|+|G_t|} \sum_{p_s \in G_s(\delta_s)} \min_{p_t \in G_t} \{d(p_s^c, p_t)\} \right] + \sum_{\delta_t \in \Delta_t} \left[\sum_{(G_s,G_t)\in G} \frac{1}{|G_s|+|G_t|} \sum_{p_t \in G_t(\delta_t)} \min_{p_s \in G_s} \{d(p_t^c, p_s)\} \right]$$

Iterating over all pixels inside a triangle is done through a basic scan line algorithm. This algorithm starts from the topmost point in the triangle and iterates over all centers of pixels from top to bottom and left to right. It travels downwards along the edges of the triangle and performs a horizontal scan line for every line of pixel centers. The horizontal scan line is aborted when the currently selected pixel is outside the respective triangle. At this point the algorithm continues with the next horizontal scan line, starting at the leftmost edge of the triangle exactly one pixel below the previous scan line.

4.2.2. Feasibility

Previously, a solution was checked for infeasibility by counting the number of intersecting edges, because any pair of intersecting edges makes a solution infeasible. This approach is however very inefficient when only a very slight number of grid points is relocated simultaneously. Because any newly generated grid point could be located anywhere on the image, any edge attached to such a point could now be intersecting any other existing edge. Therefore, checking edge intersections for a single substructure has time complexity $O(l)$ even though only $O(1)$ points have been generated. This process will make the algorithm very inefficient when a large number of variables is used. To prevent this, we introduce a method of calculating the constraint value of a solution in $O(1)$ time. If and only if a solution is feasible the result of this new constraint value will be 0. Otherwise it will likely result in a value that is not related to the number of intersecting edges, but still defines a measure of feasibility. To find the constraint value of a single point p we first define two sets of edges: a set of inner edges and a set of outer edges. Inner edges of p are defined as edges between p and one of p's neighbors. Outer edges are edges for which both end points are neighbors of p. In Figure 4.2 p is displayed as a red point alongside inner edges in blue and outer edges in red. As long as p changes position within the boundaries of the outer edges the grid will stay feasible, because no pair of edges will be able to intersect. However, p moving outside the boundary of the polygon defined by the outer edges will always lead to an inner edge intersecting an outer edge. Checking whether p is within the boundaries of the outer polygon can be performed by drawing a horizontal scan from $-\infty$ to p. Because this scan line starts at $-\infty$ it is initially outside of the boundary polygon of p. For every outer edge that it intersects it changes from outside to inside or vice versa. Therefore, an odd number of intersections from $-\infty$ to p means that p must be inside the boundary polygon of outer edges. Consequently, an even number of intersections indicates that p lies outside this boundary meaning that the triangulated grid is infeasible.

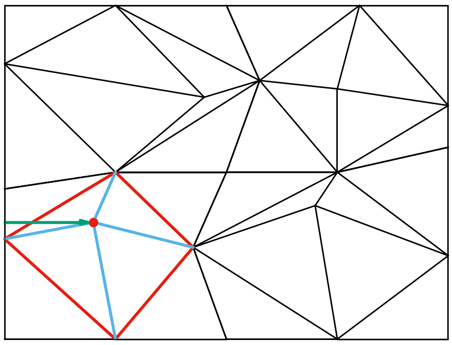


Figure 4.2: A valid triangulated grid, because the green scan line has an odd number of intersections with the outer edges in red.

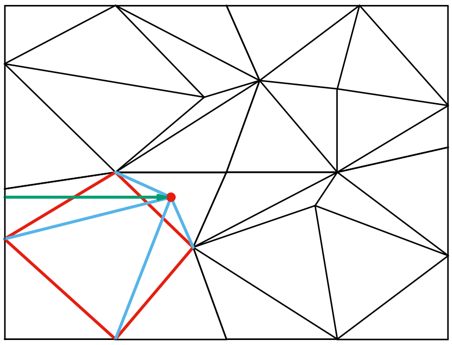


Figure 4.3: An invalid triangulated grid, because an inner edge (blue) intersects an outer edge (red). The green scan line here intersects an even number of outer edges.

For a point $p = (x_p, y_p)$ and the set of its outer edges $E_{\text{out}}(p)$, which are defined by their end points, we can now find the constraint value $f_{\text{constraint}}(p)$ of p by using the size of the set of intersecting edges as illustrated below. $f_{\text{constraint}}(p)$ is equal to 0 if the number of intersections of the scan line to p with $E_{\text{out}}(p)$ is odd, 1 if it is even. Finally, the total constraint value $f_{\text{constraint}}(P)$ of a solution can be found by calculating the sum of constraint values over all grid points.

$$\begin{aligned}
n_{\text{intersect}}(p) &= \left|\left\{\big((x_a, y_a), (x_b, y_b)\big) \in E_{\text{out}}(p) \mid \big((y_a > y_p) \neq (y_b > y_p)\big) \wedge \left(x_p > x_a + \frac{(x_b - x_a) \cdot (y_p - y_a)}{y_b - y_a}\right)\right\}\right| \\
f_{\text{constraint}}(p) &= 1 - n_{\text{intersect}}(p) \pmod 2 \\
f_{\text{constraint}}(P) &= \sum_{p \in P} f_{\text{constraint}}(p)
\end{aligned}$$

The outcome of $f_{\text{constraint}}(p)$ is not only influenced by the position of p, but also by that of its neighbors. Changing the position of a point neighboring p therefore also requires the recalculation of $f_{\text{constraint}}(p)$. After sampling a set of points we must therefore calculate the new constraint value of all these points and their neighbors. In an identical fashion as the recalculation of an objective value, the old constraint value must be subtracted from the total constraint value after which the new constraint value can be added to it. Calculating $f_{\text{constraint}}(p)$ for a single point p now has a time complexity of $O(1)$, because this operation scales with the number of outer edges of p, which is bounded by a constant.

5 Experiments

In order to determine the performance, both absolute and relative to other algorithms, of our newly designed algorithms and related adaptations of the multi-objective deformable image registration model, we perform computational experiments. From these experiments, we wish to learn conditions under which GOMEA does or does not perform well on certain optimization problems. GOMEA's performance is likely highly dependent on a combination of the type of optimization problem, the utilized dependency structure and possibly a set of different conditions. In summary, the main goal of these experiments is to get insight into:

1. Which type of optimization problems can GOMEA efficiently solve?
2. With which dependency structure is the best performance obtained for GOMEA, independently considering each of the benchmark problems?
3. Is GOMEA also worth considering for black-box optimization?

The benchmark problems used to evaluate the performance of GOMEA are first introduced in Section 5.1. Subsequently, an experimental set-up is designed that allows for a fair comparison between GOMEA and AMaLGaM. This process is described in Section 5.2. All experiments and their results are then described in Section 5.3.

5.1. Benchmark Problems

This section will contain the description of all benchmark problems that are used for our experiments. Firstly, the single-objective benchmark problems are described, followed by the multi-objective and deformable image registration benchmark problems.

5.1.1. Single-Objective Benchmark Problems

We use a set of well-known single-objective benchmark problems, consisting of a number of decomposable problems of different difficulties and with different properties. These benchmark problems are each defined as a function of a set of variables $\boldsymbol{x} = (\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_i, \ldots, \boldsymbol{x}_{\ell-1})$, which are formally defined in this section. Additionally, to indicate the shape of the objective landscape of these functions, two-dimensional heat maps of the objective values of these functions are displayed in Figure 5.1. Each color indicates a certain objective value $f(\boldsymbol{x})$ given the values of $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$ on the two axes. Darker colors indicate lower and therefore better values, because all functions have to be minimized.

Firstly, we use the sphere function, for which a two-dimensional heat map is displayed in Figure 5.1a. This function stands out as the easiest function, because each variable can be optimized independently and it has no local optima. The sphere function is defined as follows:

$$f_{\text{sphere}}(\boldsymbol{x}) = \sum_{i=0}^{\ell-1} \boldsymbol{x}_i^2$$

Secondly, the Rosenbrock function is used as a benchmark problem. A two-dimensional heat map for this function is displayed in Figure 5.1b. The global minimum at $(1,1)$ of the Rosenbrock function is very difficult to find when each variable is optimized independently. This is caused by the parabolic shape of the so-called valley, because it requires both variables to be optimized simultaneously to travel along the contours of the valley. The Rosenbrock function is formally defined as:

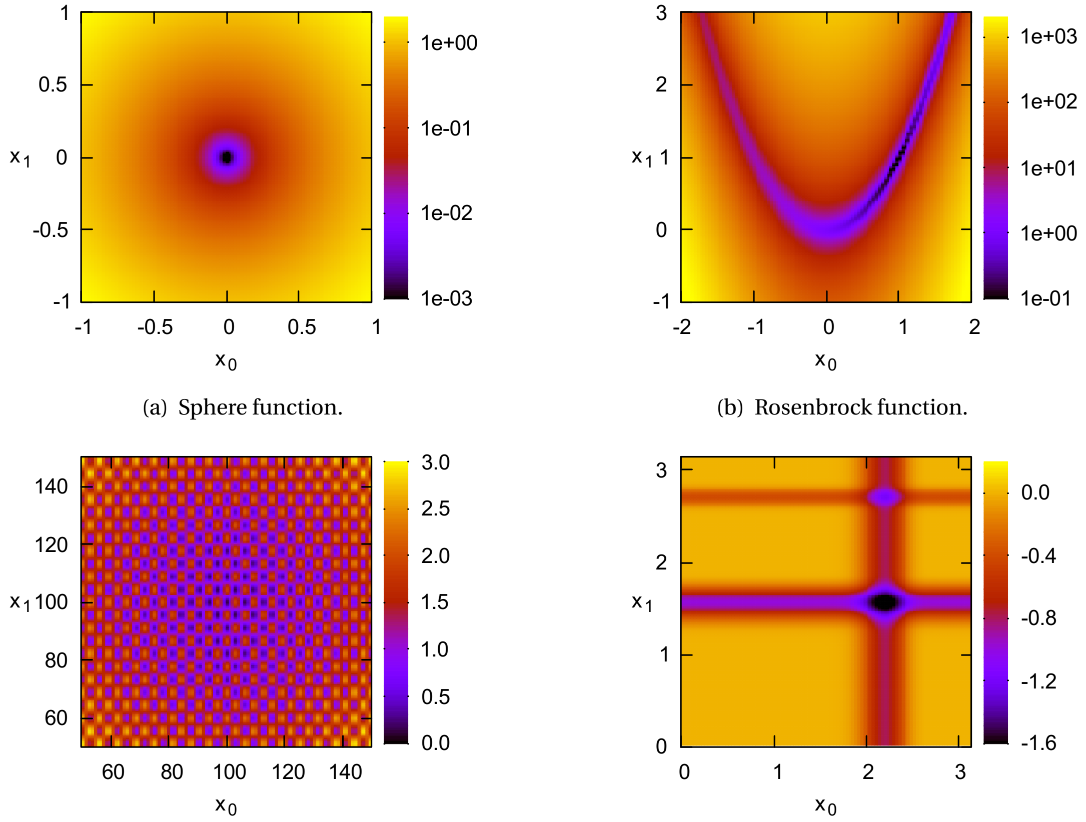
$$f_{\text{Rosenbrock}}(\boldsymbol{x}) = \sum_{i=0}^{\ell-2} \left[100(\boldsymbol{x}_{i+1} - \boldsymbol{x}_i^2)^2 + (\boldsymbol{x}_i - 1)^2\right]$$

The Griewank function is characterized by its many local minima, which makes it difficult to optimize for algorithms that employ local search methods, while it should be less difficult for EAs due to the population of solutions scattered across the search space. A heat map of the Griewank function is displayed in Figure 5.1c, and it is formally defined as:

$$f_{\text{Griewank}}(\boldsymbol{x}) = \frac{1}{4000} \sum_{i=0}^{\ell-1} \left[(\boldsymbol{x}_i - 100)^2\right] - \prod_{i=0}^{\ell-1} \left[\cos\left(\frac{\boldsymbol{x}_i - 100}{\sqrt{i+1}}\right)\right] + 1$$

Like the sphere function, the Michalewicz function is also completely separable in each of its dimensions, i.e., each variable can be optimized independently. The Michalewicz function is however not as easy as the sphere function, and it is characterized by the property that trying to model too high-order dependencies can greatly hinder convergence. A heat map of the Michalewicz function is displayed in Figure 5.1d, and the formal definition of this function is as follows:

$$f_{\text{Michalewicz}}(\boldsymbol{x}) = \sum_{i=0}^{\ell-1} \left[-\sin(\boldsymbol{x}_i) \cdot \sin\left(\frac{(i+1)\boldsymbol{x}_i^2}{\pi}\right)^{20}\right]$$



(a) Sphere function.

(b) Rosenbrock function.

(c) Griewank function.

(d) Michalewicz function.

Figure 5.1: Two-dimensional heat maps of four well-known optimization functions.

Finally, the sum of rotated ellipsoid blocks (SoREB) benchmark is used, because the objective of this benchmark only contains tight dependencies between small predefined sets of variables. In contrast, pairs of variables that are not contained in the same set have no dependency whatsoever. This structure seems similar to that of the deformable image registration problem, where there are strong dependencies between a vertex and its neighbors while there is almost no dependency between such a vertex and other distant vertices. To define the sum of rotated ellipsoid blocks function, we first define the ellipsoid function as follows:

$$f_{\text{ellipsoid}}(\boldsymbol{x}) = \sum_{i=0}^{\ell} 10^{\frac{6i}{\ell-1}} \boldsymbol{x}_i^2$$

This function sums the squares of all variables, but weighs each variable by a factor dependent on its index. It is still very easy to minimize this function, because the global optimum can be found by minimizing every variable independently. Therefore, we introduce dependencies between variables by applying a rotation function, which rotates every pair of variables over an angle θ in the 2D-plane of these two variables. For any pair of variables $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and an angle θ this rotation function can be defined as:

$$R_\theta(\boldsymbol{x}_i, \boldsymbol{x}_j) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_i \\ \boldsymbol{x}_j \end{bmatrix}$$

Rotating each pair of variables before calculating the ellipsoid function makes it much more difficult to minimize it. This is illustrated in Figure 5.2, where a two-dimensional heat map of a function similar to $f_{\text{ellipsoid}}((\boldsymbol{x}_0, \boldsymbol{x}_1))$ is displayed. The color of the heat map indicates the objective value of the ellipsoid function, which is minimized in the point $(0,0)$. Even for a set of two variables, this rotated function is much more difficult to optimize than the non-rotated function, because minimizing each of the variables individually will lead to a location near the diagonal line $\boldsymbol{x}_0 = \boldsymbol{x}_1$. From such a point, optimizing each variable independently is quite inefficient.

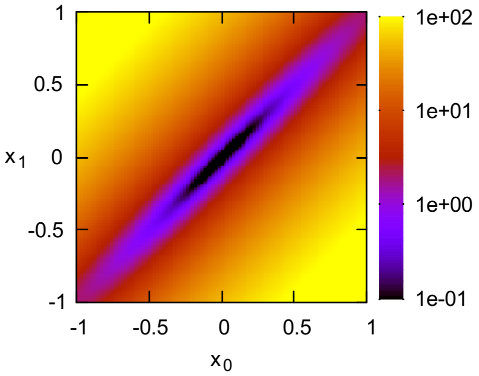


Figure 5.2: A heat map of the objective function $f((\boldsymbol{x}_{R_0}, \boldsymbol{x}_{R_1})) = \boldsymbol{x}_{R_0} + 10^2 \boldsymbol{x}_{R_1}$, where $(\boldsymbol{x}_{R_0}, \boldsymbol{x}_{R_1}) = R_\theta(\boldsymbol{x}_0, \boldsymbol{x}_1)$ and $\theta = \frac{-\pi}{4}$.

To construct small groups of dependent variables, we will use an objective function that is equal to the sum of a number of rotated ellipsoid functions. We therefore first define the function that sums the objective values of multiple ellipsoid functions of constant size k. For any set of variables $\boldsymbol{x}$ and some k that is the size of any group of dependent variables, the sum of ellipsoid blocks (SoEB) function is defined as follows:

$$f_{\text{SoEB}}(\boldsymbol{x}, k) = \sum_{i=0}^{(\ell/k-1)} f_{\text{ellipsoid}}(\boldsymbol{x}_{ki} \ldots \boldsymbol{x}_{k(i+1)-1})$$

The above function $f_{\text{ellipsoid}}$ is still trivially solvable, because no dependencies between variables exist. Instead of rotating every pair $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ to create dependencies, only every pair of variables in the same group is rotated, i.e. every pair $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ for which $\lfloor i/k \rfloor = \lfloor j/k \rfloor$. Performing this rotation before calculating the sum of ellipsoids function will result in a certain number of disjoint groups of variables where each variable has a dependency with every other variable within that group, but none with variables in different groups. This new function, named sum of rotated ellipsoid blocks (SoREB) is formally defined as follows:

$$f_{\text{SoREB}}(\boldsymbol{x}, k) = \sum_{i=0}^{(\ell/k-1)} f_{\text{ellipsoid}}(R_\theta(\boldsymbol{x}_{ki} \ldots \boldsymbol{x}_{k(i+1)-1}))$$

An overview of the range, optimal solutions and their objective values of all utilized optimization functions is given in Table 5.1.

Function	**Range**	**Optimal solution**	**Optimal objective value**
Sphere	$\boldsymbol{x}_i \in [-\infty,\infty]$	$\boldsymbol{x}^* = \{0\}^\ell$	$f(\boldsymbol{x}^*) = 0$
Rosenbrock	$\boldsymbol{x}_i \in [-\infty,\infty]$	$\boldsymbol{x}^* = \{1\}^\ell$	$f(\boldsymbol{x}^*) = 0$
Griewank	$\boldsymbol{x}_i \in [-\infty,\infty]$	$\boldsymbol{x}^* = \{100\}^\ell$	$f(\boldsymbol{x}^*) = 0$
Michalewicz	$\boldsymbol{x}_i \in [0,\pi]$	$\boldsymbol{x}_0^* \approx 2.203$ $\boldsymbol{x}_1^* \approx 1.571$ $\boldsymbol{x}_2^* \approx 1.285$ $\boldsymbol{x}_3^* \approx 1.923$ $\dots$	$f(\boldsymbol{x}^*) \approx \begin{cases} -1.801 & \ell = 2 \\ -9.660 & \ell = 10 \\ -99.620 & \ell = 100 \\ \dots \end{cases}$
Sum of rotated ellipsoid blocks	$\boldsymbol{x}_i \in [-\infty,\infty]$	$\boldsymbol{x}^* = \{0\}^\ell$	$f(\boldsymbol{x}^*) = 0$

Table 5.1: Properties of all utilized single-objective optimization functions.

5.1.2. Multi-Objective Benchmark Problems

To compare outcomes of different multi-objective optimizers, i.e., Pareto fronts, we use two well-known metrics to define the quality of a Pareto set, which will enable us to directly compare two Pareto sets that were found by different algorithms. The $D_{\boldsymbol{P}_f \to \boldsymbol{S}}$ metric [10], also known as inverse generational distance, is defined as the average minimal distance of solutions in the optimal Pareto $\boldsymbol{P}_f$ front to an arbitrary Pareto front $\boldsymbol{S}$. Because the optimal Pareto front consists of infinitely many solutions, we make a selection of many equally spread out solutions from the optimal Pareto front. First, for each solution in the optimal Pareto front the Euclidean distance in the objective space to its nearest neighbor in $\boldsymbol{S}$ is computed, which is illustrated in Figure 5.3. Subsequently, the $D_{\boldsymbol{P}_f \to \boldsymbol{S}}$ metric is defined as the average of all these minimal distances. Note that this metric requires that the optimal Pareto front is known. Therefore, we use an additional metric that can also be used when the optimal Pareto front is not known.

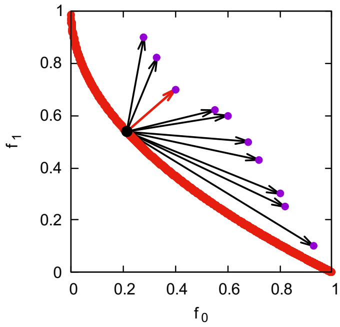


Figure 5.3: The minimal distance, marked by a red arrow, from a point in the optimal Pareto front to its nearest neighbor in an arbitrary Pareto front.

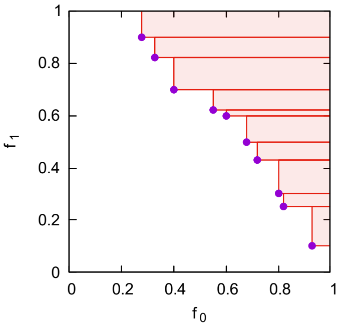


Figure 5.4: An arbitrary Pareto front, with its hypervolume shaded in red.

The second metric that will be used is the hypervolume metric [26], which defines the total k-dimensional volume in the objective space that is dominated by a Pareto front, which is illustrated in Figure 5.4. This metric is used, because it gives a clear indication of the diversity of a Pareto front and its progress towards the optimal front. Any solution, however, dominates an infinitely large volume, because any objective could be arbitrarily large. Therefore, an upper bound is set for each objective, leading to a k-dimensional cuboid with a finite volume enclosing the objective space that is considered by the hypervolume metric. The point that has the upper bound of each objective as its coordinates is the so-called Nadir point. Generally, objective values are normalized such that each objective value of any solution on the optimal Pareto front is between 0 and 1. In this case, the upper bound of each objective is set to 1.1, i.e., the Nadir point is $(1.1, 1.1, 1.1, \dots)$.

Evaluating the performance of multi-objective GOMEA will be done using five benchmark problems, four of which are some of the well-known multi-objective Zitzler-Deb-Thiele's (ZDT) optimization problems, namely ZDT1, ZDT2, ZDT3, and ZDT6. All of these optimization functions have two objective functions, $f_1(\boldsymbol{x})$ and $f_2(\boldsymbol{x})$, which are defined in Figures 5.5 to 5.8. For each of these problems the optimal Pareto front is known, which are displayed in Figure 5.10.

$$\begin{aligned}
f_1(\boldsymbol{x}) &= \boldsymbol{x}_0 \\
f_2(\boldsymbol{x}) &= g(\boldsymbol{x}) \cdot h(f_1(\boldsymbol{x}), g(\boldsymbol{x})) \\
g(\boldsymbol{x}) &= 1 + \frac{9}{\ell - 1} \sum_{i=1}^{\ell-1} \boldsymbol{x}_i \\
h(f_1(\boldsymbol{x}), g(\boldsymbol{x})) &= 1 - \sqrt{\frac{f_1(\boldsymbol{x})}{g(\boldsymbol{x})}}
\end{aligned}$$

Figure 5.5: ZDT1 function.

$$\begin{aligned}
f_1(\boldsymbol{x}) &= \boldsymbol{x}_0 \\
f_2(\boldsymbol{x}) &= g(\boldsymbol{x}) \cdot h(f_1(\boldsymbol{x}), g(\boldsymbol{x})) \\
g(\boldsymbol{x}) &= 1 + \frac{9}{\ell - 1} \sum_{i=1}^{\ell-1} \boldsymbol{x}_i \\
h(f_1(\boldsymbol{x}), g(\boldsymbol{x})) &= 1 - \left(\frac{f_1(\boldsymbol{x})}{g(\boldsymbol{x})}\right)^2
\end{aligned}$$

Figure 5.6: ZDT2 function.

$$\begin{aligned}
f_1(\boldsymbol{x}) &= \boldsymbol{x}_0 \\
f_2(\boldsymbol{x}) &= g(\boldsymbol{x}) \cdot h(f_1(\boldsymbol{x}), g(\boldsymbol{x})) \\
g(\boldsymbol{x}) &= 1 + \frac{9}{\ell - 1} \sum_{i=1}^{\ell-1} \boldsymbol{x}_i \\
h(f_1(\boldsymbol{x}), g(\boldsymbol{x})) &= 1 - \sqrt{\frac{f_1(\boldsymbol{x})}{g(\boldsymbol{x})}} - \frac{f_1(\boldsymbol{x})}{g(\boldsymbol{x})} \sin\left(10\pi f_1(\boldsymbol{x})\right)
\end{aligned}$$

Figure 5.7: ZDT3 function.

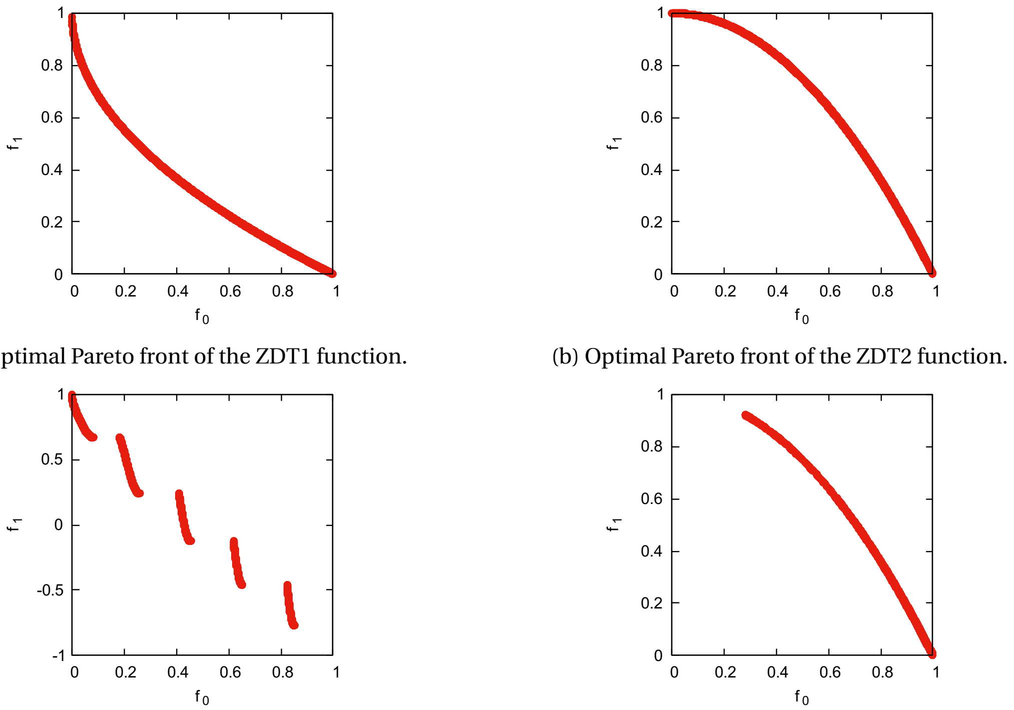
$$\begin{aligned}
f_1(\boldsymbol{x}) &= 1 - \exp(-4\boldsymbol{x}_0) \sin(6\pi\boldsymbol{x}_0)^6 \\
f_2(\boldsymbol{x}) &= g(\boldsymbol{x}) \cdot h(f_1(\boldsymbol{x}), g(\boldsymbol{x})) \\
g(\boldsymbol{x}) &= 1 + 9\left(\sum_{i=1}^{\ell-1} \frac{\boldsymbol{x}_i}{\ell - 1}\right)^{\frac{1}{4}} \\
h(f_1(\boldsymbol{x}), g(\boldsymbol{x})) &= 1 - \left(\frac{f_1(\boldsymbol{x})}{g(\boldsymbol{x})}\right)^2
\end{aligned}$$

Figure 5.8: ZDT6 function.

The fifth benchmark is a multi-objective adaptation of the sum of rotated ellipsoid blocks function, which was introduced in Section 5.1.1 as a single-objective optimization problem. This benchmark is comprised of two objectives, f_0 and f_1, which are defined in Figure 5.9. The first of these objectives is the sum of rotated ellipsoid blocks function with block size k, having an optimal objective value of $f(\boldsymbol{x}^*) = 0$ for $\boldsymbol{x}^* = \{0\}^\ell$. The second objective is the sphere function, which was also introduced in Section 5.1.1. However, in order to conflict with the prior objective, this sphere function is translated to bring its optimum to $\boldsymbol{x}^* = \{1\}^\ell$ with $f(\boldsymbol{x}^*) = 0$. To bring the range of the optimal Pareto front of this function within the range between 0 and 1, the first objective is normalized by dividing by $f_0(\{1\}^\ell)$. Even though the range of the optimal Pareto front is known, the exact points on it are not. Therefore comparisons using this benchmark must be done using the hypervolume metric.

$$\begin{aligned}
f_1(\boldsymbol{x}, k) &= \frac{1}{f_{\text{SoREB}}(\{1\}^\ell, k)} f_{\text{SoREB}}(\boldsymbol{x}, k) \\
f_2(\boldsymbol{x}) &= \sum_{i=0}^{\ell-1} (\boldsymbol{x}_i - 1)^2
\end{aligned}$$

Figure 5.9: Sphere-SoREB function.



(a) Optimal Pareto front of the ZDT1 function.

(b) Optimal Pareto front of the ZDT2 function.

(c) Optimal Pareto front of the ZDT3 function.

(d) Optimal Pareto front of the ZDT6 function.

Figure 5.10: Optimal Pareto fronts of four well-known multi-objective optimization functions.

5.1.3. Deformable Image Registration Benchmark Problems

Evaluating the performance of the EAs on the deformable image registration is done through a set of benchmark images with different properties. These images are displayed in Figure 5.11. The first pair of benchmark images, displayed in Figures 5.11a and 5.11b, concerns a pair of artificial images containing a large disappearing structure. Also the second pair of images contains a disappearing structure. The source image is a pre-operative breast CT scan, while the target image is a post-operative CT scan of the same breast. Finally, the third pair of images consists of two 2D slices from different MRI scans. Of these two images, the source image was acquired with the subject being in a prone position, while the target image was acquired with the subject being in a supine position. For the first two benchmarks, guidance information is available as a set of points that are used for each experiment concerning these benchmarks.

To evaluate the quality of a registration outcome, we need to consider some additional factors due to the fact that deformable image registration is a real-world application. Firstly, for none of the benchmarks the optimal Pareto front is known, which prohibits the use of the $D_{\mathbf{P}_f \to \mathbf{S}}$ metric [10]. Instead, the hypervolume metric will mostly be used to evaluate the quality of a registration outcome. Secondly, the clinical quality of a registration outcome can differ somewhat from the perceived quality based on the optimization functions. Moreover, registration outcomes with the highest similarity are often not the most clinically desirable. For this reason, as an additional metric to evaluate the quality of a registration outcome, the target registration error (TRE) is used. This metric requires a set of professionally annotated landmark locations on the source image and for each location a corresponding location on the target image. For one landmark the TRE is then defined as the distance between the transformed source landmark and the target landmark. As the final metric, the mean TRE is used, which is very similar to the guidance objective, but, in contrast to this objective, the TRE is never used to support optimization. Instead, it is strictly used as a post-processing method to evaluate the quality of a registration outcome.

5.2. Set-up

All benchmarks introduced in this chapter are experimented with in a black-box and a grey-box setting. The grey-box setting allows partial evaluations to be performed, which are counted as a fraction k/ℓ of an evaluation, where k is the number of modified variables. In contrast, any evaluation of a solution is counted as a complete evaluation when a black-box setting is used. For the sphere, Rosenbrock, Michalewicz, and SoREB functions, partial evaluations were implemented to speed up evaluations in a grey-box setting. However, partially evaluating the Griewank function would require problem specific adaptation of the EA to individually keep track of the sum and product factors that constitute the objective function. Any partial evaluation of the Griewank function is therefore performed identical to a full evaluation, but it is only counted as a fraction k/ℓ of an evaluation in a grey-box setting. This means that the time reported by the black-box Griewank function will be identical to that of the grey-box Griewank function. For this reason, for all required times reported for the Griewank function, a black-box setting was used.

All experiments are performed on one of four computers, two pairs of two identical machines. The first pair of computers uses an Intel® Core™ i7-2600 CPU @ 3.40GHz, and the second pair of computers uses an Intel® Core™ i7-3770S CPU @ 3.10GHz. To account for the disparity between these computers, we scale the resulting number of seconds of experiments performed on the fastest computer down by a constant factor. This factor is determined by performing 1000 runs of univariate GOMEA on the sphere problem with 10^4 variables using guideline parameters. The fastest of the two computers required on average 3.72 seconds, compared to 4.06 seconds for the slower computer, resulting in a scaling factor of 1.09. All times resulting from experiments performed on the faster set of computers are divided by this factor 1.09.

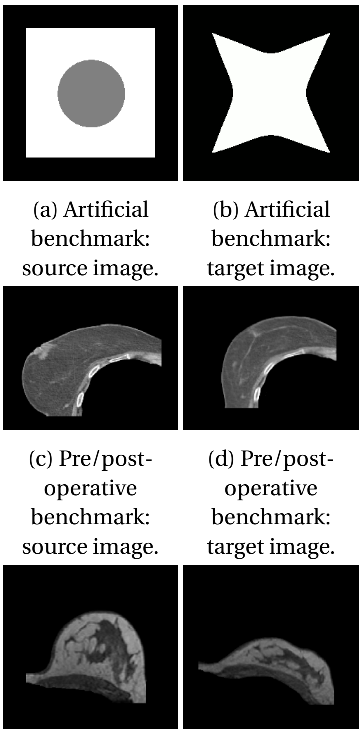

(a) Artificial benchmark: source image.
(b) Artificial benchmark: target image.
(c) Pre/post-operative benchmark: source image.
(d) Pre/post-operative benchmark: target image.
(e) Prone/supine benchmark: source image.
(f) Prone/supine benchmark: source image.

Figure 5.11: All images used as benchmarks for the deformable image registration problem.

5.3. Results

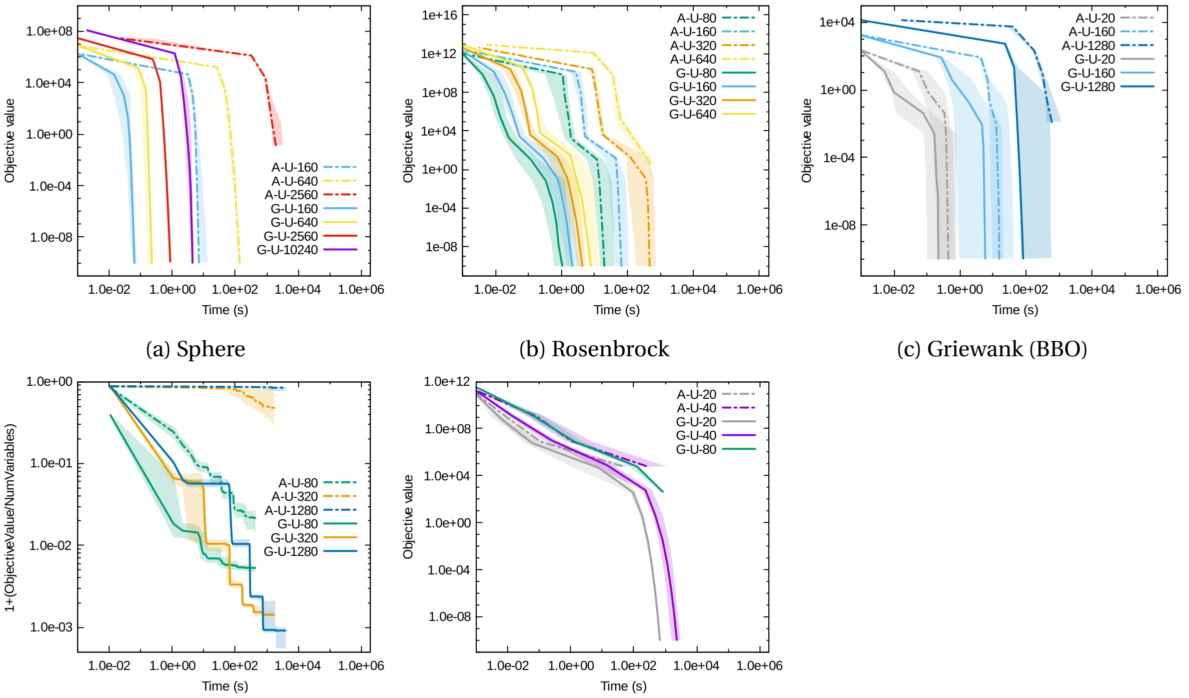
In this subsection, the results of all experiments are reported. The results for the single-objective and multi-objective GOMEA, as well as those for the deformable image registration problem, are each reported in a separate subsection. To increase clarity and readability, all graphs in these subsections only show a limited number of problem sizes. Tables reporting the final results of all experiments on all problem sizes are included in Appendix A and B. When an algorithm was unable to reach the target objective value of the problem in any run, no experiment was performed on that problem of any larger dimensionality. Cells are shaded if a success rate lower than 100% was achieved, with the success rate defined as the percentage of runs that achieved the VTR within the predefined time limit. We use the following shadings for the following ranges of the success rate p^{suc}. No results are shown for $p^{\text{suc}} < 10\%$, we use red shading for $10\% \le p^{\text{suc}} < 50\%$, orange shading for $50\% \le p^{\text{suc}} < 90\%$, yellow shading for $90\% \le p^{\text{suc}} < 100\%$, and no shading for $p^{\text{suc}} = 100\%$. Note that experiments without a predefined VTR cannot have failed runs.

5.3.1. Single-Objective Optimization

For each experiment reported in this section, we performed 30 independent runs. For all grey-box experiments, we choose to plot the time against the objective value, because the time required for evaluations is often relatively little compared to the total time required. Only reporting the number of evaluations would not include the model-building time required, which is often substantial compared to the time required for the evaluations. Conversely, in black-box optimization evaluations are often the most time-consuming and are therefore reported. Graphs display a line describing the linearly interpolated median time of these 30 runs, and a marked area describing the interdecile range, i.e., the range of values between the 10th and 90th percentile. The displayed median was always an interpolation of all 30 runs. If in at least one of the 30 runs a certain objective value was not reached, no line is plotted from this objective value onward. The legend is formatted as $Algorithm - FOS - \ell$. GOMEA is abbreviated to G and AMaLGaM is abbreviated to A. The univariate FOS structure is indicated by U and the full FOS structure by F.

All single-objective benchmark problems that are considered, were introduced in Section 5.1.1. The experiments on the sphere, Rosenbrock, Griewank, and SoREB problems used a value-to-reach (VTR) of 10^{-10}. Any run is terminated when an objective value smaller than the VTR is reached by any individual, after which the elapsed time and total number of evaluations are reported. Additionally, a time limit of one hour was used, meaning that any run is terminated an hour after starting, regardless of its result. If an algorithm was unable to reach the VTR within the time limit of a certain run, this run is deemed unsuccessful. In contrast to the previously stated benchmark problems, the optimal objective value of the Michalewicz function is not equal to 0. Instead, it is a value slightly above $-\ell$. A VTR for the Michalewicz function would have to be dependent on ℓ and would require us to calculate the optimal objective value, which can be quite time consuming. Therefore, the experiments on the Michalewicz function do not use a VTR. They are only subject to a time limit of 5ℓ seconds, up to a maximum of 1 hour. Additionally, we do not directly report an algorithm's objective value achieved on the Michalewicz function, because the range of achieved objective values depends on ℓ. Instead, we report the fraction $1 + (f_{\text{Michalewicz}}/\ell)$, indicating the distance towards $-\ell$, the lower bound on the optimal objective value. Lower values of $1 + (f_{\text{Michalewicz}}/\ell)$ are superior to higher values. A base population size of 10 was used, because using a base population size of 5 would lead to a selection set of size 1. The remaining parameters are set according to their guidelines, i.e., an initialization range between -115 and -100, a selection percentile of $\tau = 0.35$, a distribution multiplier decreaser of $\eta^{\text{DEC}} = 0.9$, a standard deviation ratio threshold of $\theta^{\text{SDR}} = 1$, a maximum no improvement stretch of $25 + \ell$, and a fitness variance tolerance level of 10^{-10}.

Firstly, we report the results of all experiments with algorithms using a univariate FOS structure. These results for the sphere, Rosenbrock, Griewank, Michalewicz, and SoREB problems are displayed in Figure 5.12.



(a) Sphere

(b) Rosenbrock

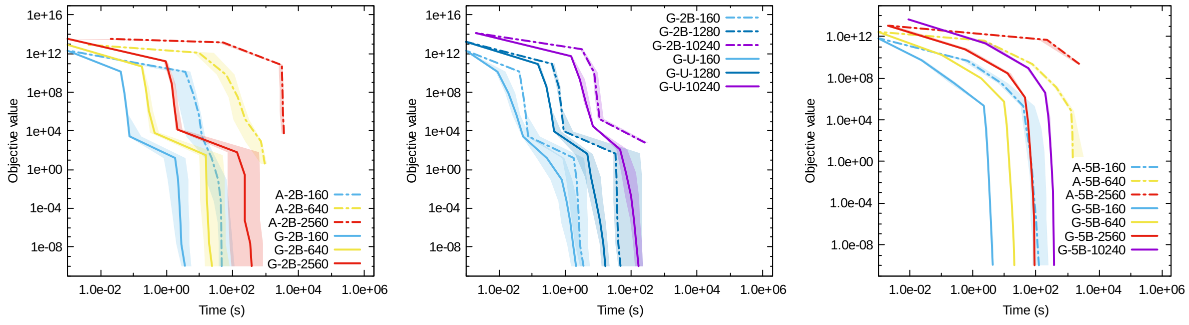
(c) Griewank (BBO)

(d) Michalewicz

(e) SoREB

Figure 5.12: Experiments measuring time of GOMEA and AMaLGaM using a univariate FOS structure, for all considered benchmarks, in a grey-box setting. Only for the Griewank function a black-box setting was used.

Because some problems contain dependencies between variables, it can be beneficial to model these variables in the FOS. Of the problems that were experimented with, only the Rosenbrock and SoREB problems contain dependencies. The Rosenbrock function was well solvable by GOMEA and AMaLGaM using a univariate FOS structure, but the SoREB function was not, as displayed in Figure 5.12e. This figure shows that both GOMEA and AMaLGaM were unable to solve the SoREB function with more than 40 variables within the time limit of one hour, using a univariate FOS structure. Therefore we perform the same experiments on the Rosenbrock and SoREB functions using FOS structures specifically tuned for these problems. For the Rosenbrock function a so-called 2-block FOS is used, which is is abbreviated to 2B in the legend. It models dependencies within disjoint blocks of 2 variables each, i.e., $\boldsymbol{F}_i = \{2i, 2i+1\}$. This FOS does not model each dependency present in the Rosenbrock function, because the Rosenbrock function has overlapping dependencies between every pair of consecutive variables. However, only modeling a subset of dependencies could still prove to be beneficial. Results of this experiment are displayed in Figure 5.13a. We also compare GOMEA's performance with the 2-block structure to its performance with the univariate structure, which is displayed in Figure 5.13b. For the SoREB function we use a so-called 5-block FOS, which is abbreviated to 5B in the legend. It models dependencies within disjoint blocks of 5 variables each, i.e., $\boldsymbol{F}_i = \{5i, 5i+1, 5i+2, 5i+3, 5i+4\}$. This FOS exactly models each dependency present in the SoREB problem. The results of this experiment are displayed in Figure 5.13c.

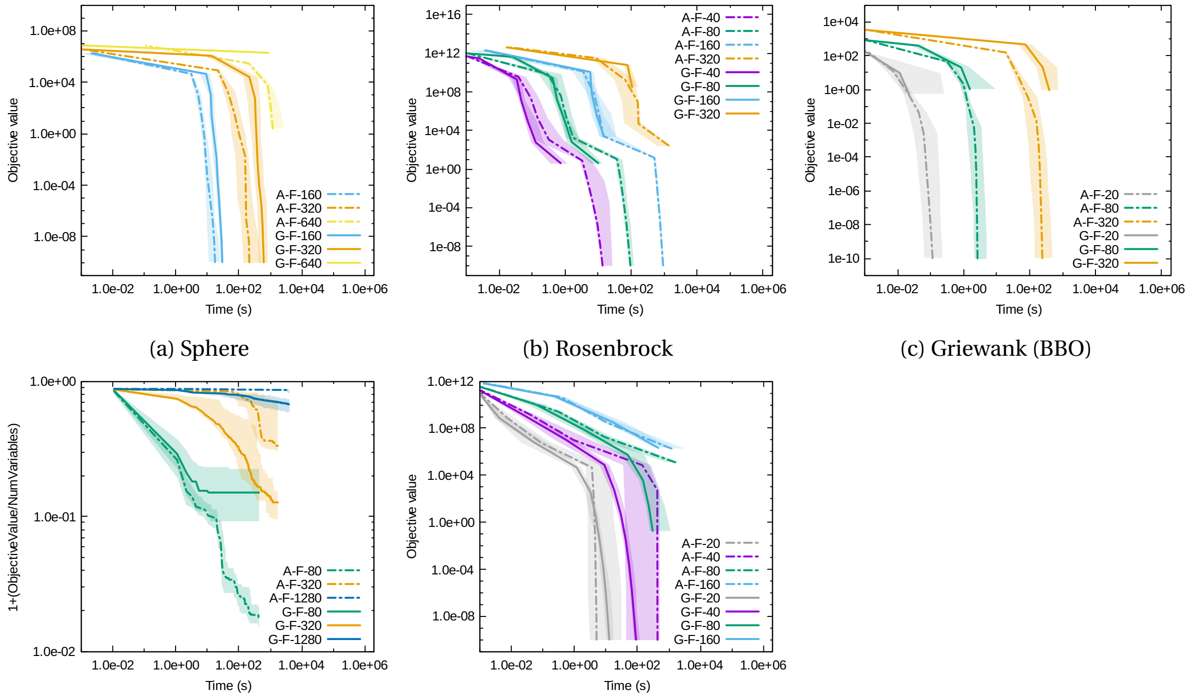


(a) 2-Block Rosenbrock

(b) Univariate and 2-Block Rosenbrock for GOMEA

(c) 5-Block SoREB

Figure 5.13: Experiments measuring time of GOMEA and AMaLGaM using a k-block FOS structure, for the Rosenbrock and SoREB functions, in a grey-box setting.



(a) Sphere

(b) Rosenbrock

(c) Griewank (BBO)

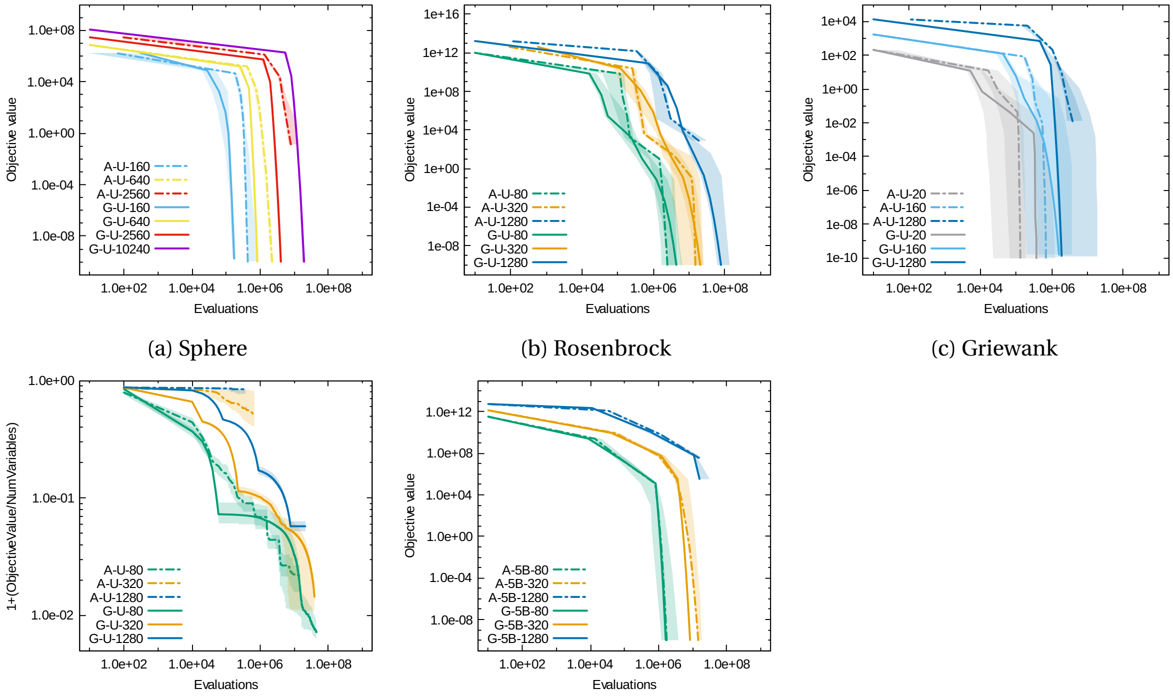
(d) Michalewicz

(e) SoREB

Figure 5.14: Experiments measuring time of GOMEA and AMaLGaM using a full FOS structure, for all considered benchmarks, in a grey-box setting. Only for the Griewank function a black-box setting was used.

We also repeat the experiments on all problems using a so-called full FOS. Such a FOS consists of one element that contains each variable. The results of these experiments are displayed in Figure 5.14.

We now test the efficiency of GOMEA for black-box optimization. This means that no knowledge about the problem is assumed to be available and partial evaluations cannot be performed. Where GOMEA would otherwise use a partial evaluation to efficiently compute a newly achieved objective value, it must now completely reevaluate the individual, regardless of the number of variables that were changed. Therefore, the evaluation of any individual is counted as a single, full evaluation. In terms of time, the difference between black-box and grey-box GOMEA is quite minor, because little of the computation time is consumed by evaluations in our case. Instead, the number of evaluations required by black-box GOMEA is almost ℓ times larger than that of grey-box GOMEA. Therefore, for these experiments we report the number of evaluations instead of the time, because this leads to a more realistic comparison. For each benchmark problem, we only consider the most appropriate FOS structure, i.e., the univariate structure for the sphere, Rosenbrock, Griewank and Michalewicz functions, and the 5-block structure for the SoREB function. The results of these experiments are displayed in Figure 5.15.



(a) Sphere

(b) Rosenbrock

(c) Griewank

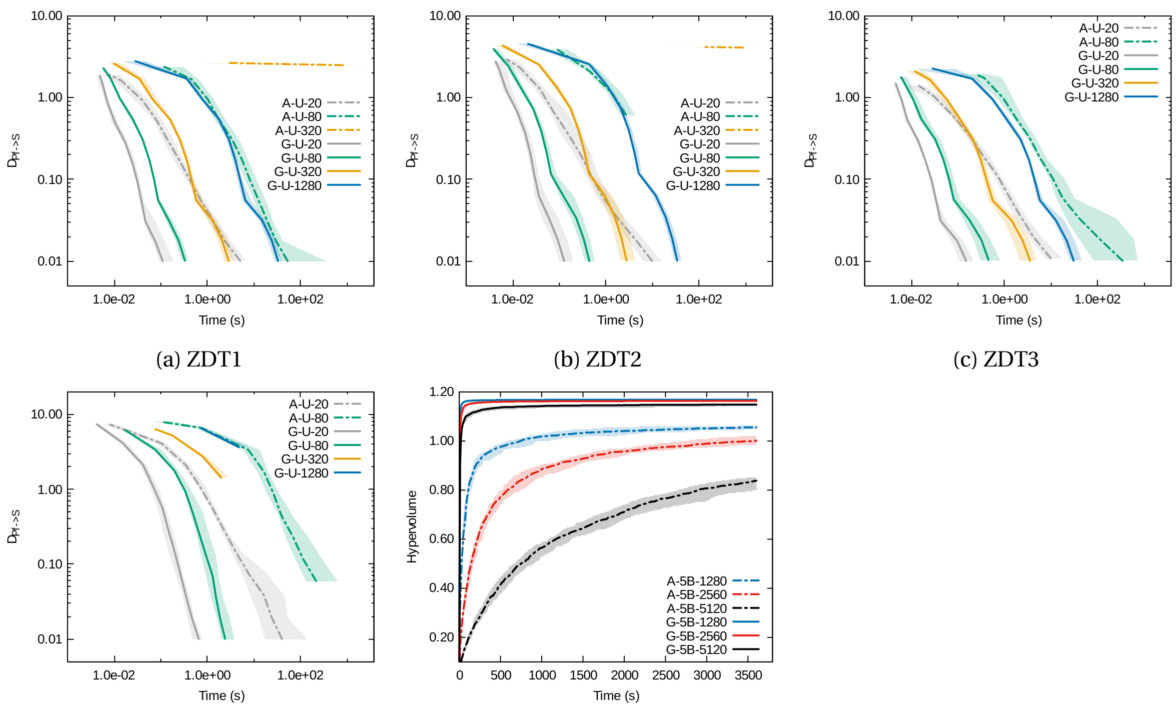
(d) Michalewicz

(e) SoREB

Figure 5.15: Experiments measuring the number of evaluations of GOMEA and AMaLGaM for each of the optimization problems in a black-box setting, using the FOS structure previously found to be the most successful.

5.3.2. Multi-Objective Optimization

For each experiment reported in this section, 30 independent runs were performed with a time limit of one hour. Graphs display a line describing the linearly interpolated median of these 30 runs, and a marked area describing the interdecile range, i.e., the range of values between the 10th and 90th percentile. The displayed median was always an interpolation of all 30 runs. If at least one of the 30 runs was unable to reach a certain performance indicator value, no line is plotted from this value onward. In Section 5.1.2 we introduced four benchmark problems for which the optimal Pareto front is known, the ZDT1, ZDT2, ZDT3, and ZDT6 functions, and one for which it is not, the Sphere-SoREB function For each of the ZDT functions, we report the $D_{\mathbf{P}_f \to \mathbf{S}}$ over time, and employ a VTR of 0.01 for the $D_{\mathbf{P}_f \to \mathbf{S}}$. For the Sphere-SoREB function, the hypervolume was measured over time, without using a VTR. All benchmarks were only tested in a black-box setting, because problem specific adaptations would be required to perform efficient partial evaluations. Only the most appropriate FOS structure was used, because extensive experiments on different FOS structures were already performed for the single-objective benchmarks. For each of the ZDT problems, a univariate FOS was used, and for the Sphere-SoREB problem, a 5-block FOS was used. A base number of clusters of 3 was used, because this is the minimum number of clusters required to have at least one non-single-objective cluster in our case of having 2 objectives. A base population size of 30 was used, leading to a cluster size of 10, which is identical to the base population size used for the single-objective experiments. The remaining parameters are identical to those used in Section 5.3.1, in addition to an elitist archive target size of 1000. Graphs describing the results are displayed in Figure 5.16.



(a) ZDT1 (b) ZDT2 (c) ZDT3

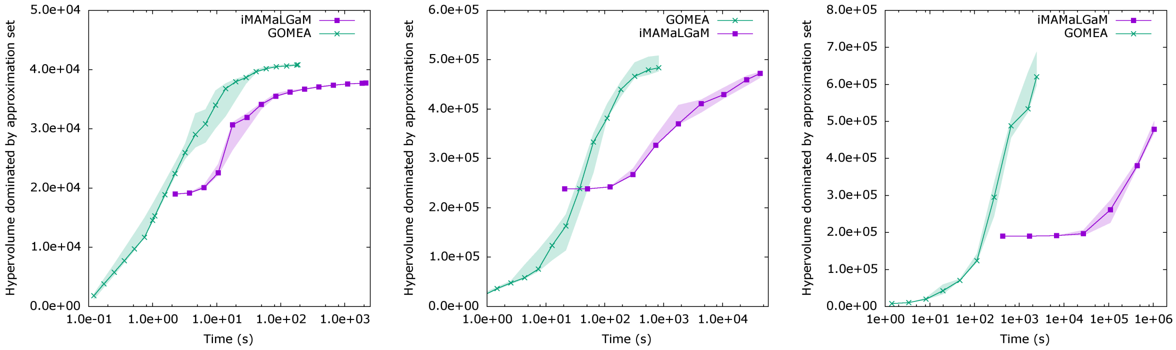
(d) ZDT6 (e) Sphere-SoREB

Figure 5.16: Experiments measuring time of GOMEA and AMaLGaM using a univariate FOS structure, for all considered benchmarks, in a black-box setting.

5.3.3. Deformable Image Registration

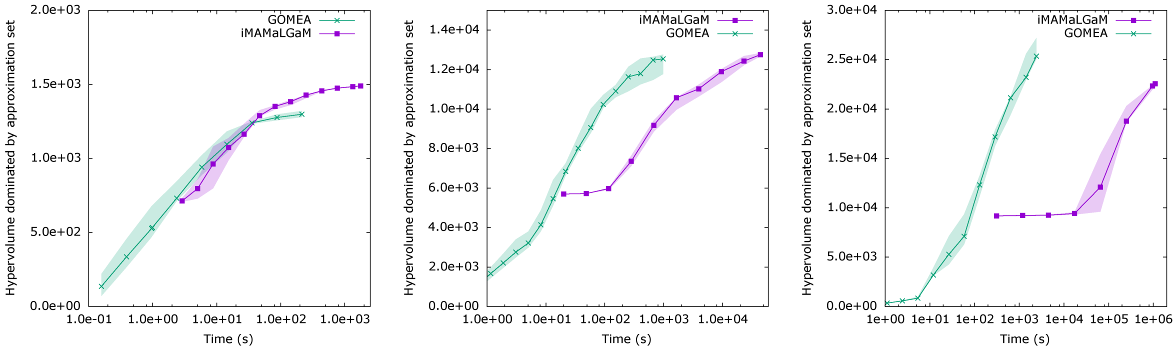
The experiments in this section are aimed at evaluating the impact that GOMEA will have when it is applied in practice to DIR. We are mostly interested in the speed-up it can achieve and whether it is capable of obtaining results of similar quality. Due to the practical application, these experiments, however, have a different goal than the experiments on single-objective and multi-objective benchmarks, described in Sections 5.3.1 and 5.3.2. These previous experiments were strictly aimed at evaluating the impact of GOMEA's core principles, such as the optimal mixing procedure. In contrast, experiments in this section will compare GOMEA to the most recently published implementation of iMAMaLGaM, because this most accurately depicts the speed-up of GOMEA compared to the previously used multi-objective EA. The findings of these experiments were also submitted as a 4-page abstract to the SPIE medical imaging conference of 2017. If accepted for publication, which is yet unknown, this abstract will be extended to a full paper. The 4-page abstract is included in Appendix C

The first experiment aims to define the speed-up that is achieved by GOMEA for different grid resolutions. Because a higher resolution grid requires a larger number of variables, GOMEA should obtain a larger speed-up, due to its beneficial scaling mostly as a result of partial evaluations. For the purpose of comparison, only single-resolution grids are used. A multi-resolution grid is more efficient, but is too dependent on its initial population. Using a fixed time or number of evaluations for each resolution will lead to either of the algorithms having an initial population of higher quality in the subsequent experiment using a higher resolution. Therefore, comparing two different algorithms using a multi-resolution grid will inevitably lead to a bias towards one of the algorithms. We therefore perform this experiment on the single-resolution grids with resolution 6 × 6, 11 × 11, and 21 × 21. The first two benchmark problems introduced in Section 5.1.3, namely an artificial benchmark and pair of pre- and post-operative breast CT scans, are used. Both benchmark problems involve a disappearing structure and include guidance information. Each run was subject to a maximum of $10^4\ell$ evaluations, respectively being $144\cdot10^4$, $484\cdot10^4$, and $1764\cdot10^4$ evaluations for each of the resolutions. For this purpose, each of GOMEAs partial evaluations was counted as a full evaluation. Each experiment was performed 4 times. The results of the artificial benchmark are displayed in Figure 5.17, and the results of the pre- and post-operative benchmark are displayed in Figure 5.18. The plotted lines in Figures 5.17 and 5.18 display the results of the second-worst run, and the marked area displays the range between the results of the best and the worst run. With the 21 × 21 resolution grid iMAMaLGaM did not reach the maximum number of evaluations, as it was terminated after running for approximately 12 days.



(a) 6 × 6 (b) 11 × 11 (c) 21 × 21

Figure 5.17: Hypervolumes of GOMEA and iMAMaLGaM on the artificial benchmark problem, using the three listed single-resolution grids.



(a) 6 × 6 (b) 11 × 11 (c) 21 × 21

Figure 5.18: Hypervolumes of GOMEA and iMAMaLGaM on the pre- and post-operative benchmark problem, using the three listed single-resolution grids.

The second experiment is aimed at evaluating the quality of registration outcomes of GOMEA. It uses a multi-resolution approach with grid resolutions of 6 × 6, 11 × 11, and 21 × 21. After a run at resolution k, the resulting approximation set is used as the initial population of the run with resolution $2k-1$. The Pareto front resulting from each resolution is reported in Figure 5.19, alongside the best mean TRE of any solution in this front. For each Pareto front, the solution with the best mean TRE is encircled in red, green, or blue, respectively for increasing resolution. These solutions achieved a mean TRE of 2.0mm, 1.6mm, and 2.3mm, respectively. Deformations of the solutions with minimal mean TRE are visualized in Figures 5.20a to 5.20c by applying them to a uniform grid. A time limit equal to the number of variables in seconds, respectively 144, 484, and 1764 seconds, was used for each of these resolutions, totaling close to 40 minutes. The benchmark that was used was the pair of 2D-slices of MRI breast scans, acquired in prone and supine position.

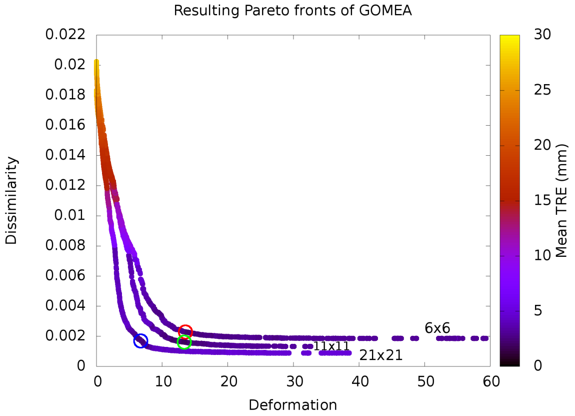
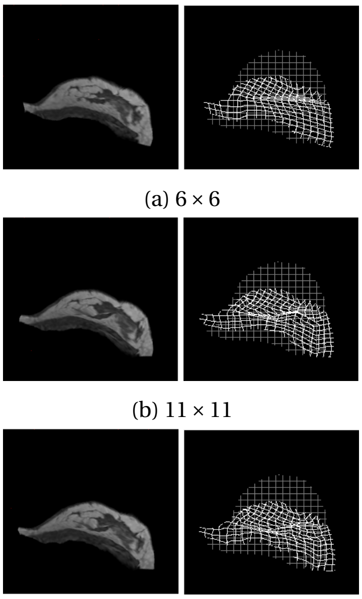


Figure 5.19: Pareto fronts of GOMEA after the three different stages of the multi-resolution approach using resolutions of 6 × 6, 11 × 11, and 21 × 21. Solutions with the best mean TREs are encircle in red, green, and blue for each of these resolutions, respectively.


(a) 6 × 6

(b) 11 × 11

(c) 21 × 21

Figure 5.20: Resulting image (left) and deformations (right) of the encircled solutions in Figure 5.19. Deformations are visualized as deformations applied to a uniform grid.

6
Discussion

After reporting the results of all conducted experiments in Chapter 5, we will now discuss these results and their implications in this chapter.

6.1. Single-Objective Optimization

First and foremost, we have observed that the FOS structure used by GOMEA potentially has a large impact on its performance. A FOS that is too small is unable to model all dependencies present in the problem, causing GOMEA to be unable to solve problems with a moderate number of variables in a reasonable amount of time. This is clearly visible in Figure 5.12e, where it can be seen that GOMEA with a univariate FOS can only solve the SoREB function for up to 40 variables within the predefined time limit of one hour. However, it still outperformed the univariate AMaLGaM, which was not even able to reach the VTR for the problem with 20 variables.

When an adequate FOS structure is used, however, GOMEA achieves a substantial increase in efficiency compared to AMaLGaM, because GOMEA can exploit the decomposability of a problem. When a FOS structure correctly models groups of dependent variables, the most promising search direction can be found relatively easily, because the acceptation procedure of a modified individual is not influenced by noise caused by the modification of other independent variables. This means that the optimization of variables in different FOS elements is largely independent. This independence is reinforced by the fact that each FOS element has its own $c^{\text{Multiplier}}$, which is updated exclusively based on the improvements caused by the respective FOS element, and by the fact that each FOS element also performs a round of AMS independently. Each FOS element therefore virtually optimizes its own subproblem, leading to a substantially smaller required population size compared to EAs that do not exploit the decomposability of optimization problems. This leads to a substantial increase in efficiency of GOMEA compared to AMaLGaM, because fewer evaluations are required per generation. The efficiency of GOMEA is improved even further when efficient partial evaluations are possible.

In contrast to the SoREB problem, all other single-objective benchmark problems can be solved efficiently using a univariate FOS structure. On these problems, GOMEA obtains a substantial speed-up compared to AMaLGaM, which can be seen in Figures 5.12a to 5.12d. To put things into perspective, GOMEA solved the sphere problem of up to 10240 variables in fewer than 10 seconds, whereas AMaLGaM was unable to solve the sphere problem with 2560 variables in one hour. A very similar scenario occurs for the Rosenbrock function, where GOMEA solves the problem with 640 variables within 10 seconds, while AMaLGaM is again unable to solve it in one hour. The times required by GOMEA and AMaLGaM are very similar for the Griewank problem, which is likely caused by the inability to calculate partial evaluations. Allowing GOMEA to use partial evaluations on the Griewank problem requires problem-specific adaptations, but would likely increase the performance significantly. Finally, for each problem size, GOMEA was able to come very close to the lower bound of $-\ell$ on the Michalewicz problem, because GOMEA exploits the decomposability of the Michalewicz problem. In contrast, AMaLGaM only performed relatively well for problem sizes up to 80, as displayed in Figure 5.12d. Because AMaLGaM is unable to solve the Michalewicz problem efficiently, GOMEA increased the set of efficiently solvable problems.

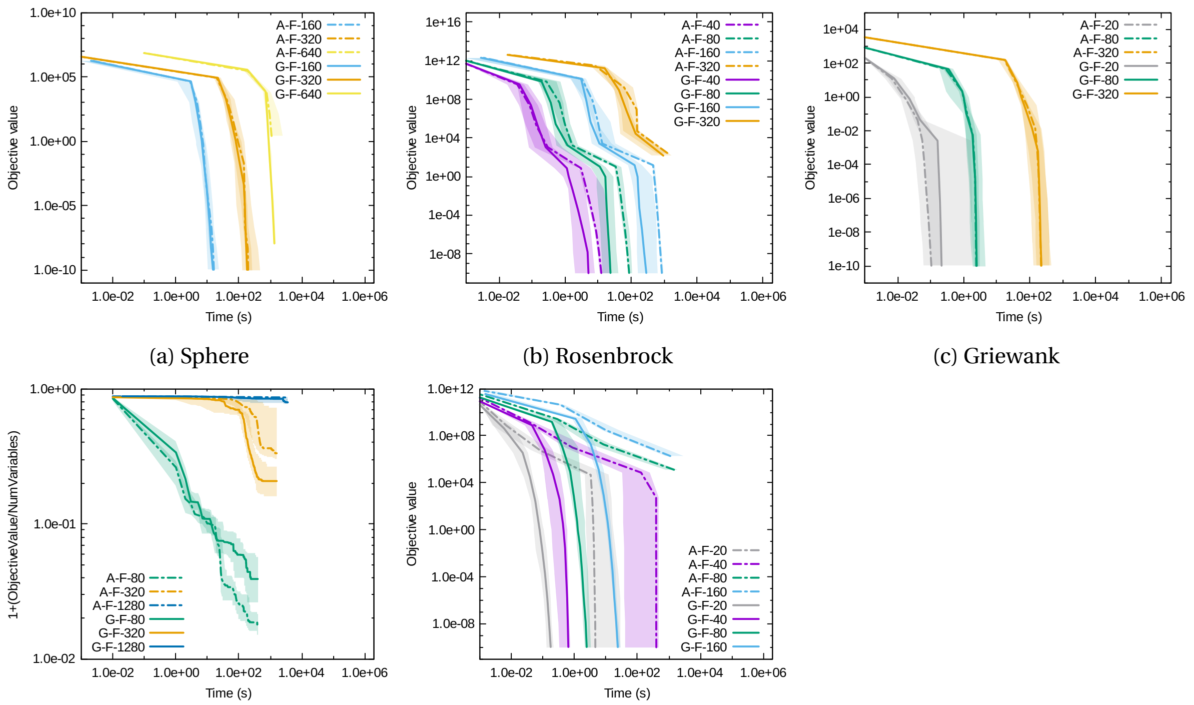
In contrast to the univariate GOMEA, which performed very poorly on the SoREB function, the 5-block

GOMEA performed very well, beating AMaLGaM by a wide margin. This is displayed in Figure 5.13. Even for a problem with 160 variables, GOMEA was more than an order of magnitude faster, with this factor even increasing for larger problem sizes. However, using a 2-block structure did not increase performance on the Rosenbrock problem, compared to the univariate structure. Even though 2-block GOMEA outperformed AMaLGaM, it was unable to achieve better performance than the univariate GOMEA. This is likely caused by the incomplete dependency structure, which only models a subset of the problem's dependencies. Adding these observations to those of the univariate experiments, this leads us to believe that GOMEA's speed-up compared to AMaLGaM is mostly enhanced when its dependency structure matches the problem dependencies.

The results of using a full dependency structure, one that models dependencies between all variables, are displayed in Figure 5.14. In this case, GOMEA was only slower than AMaLGaM on the sphere function, but had a number of prematurely converged runs on the Rosenbrock and Griewank problems, as is displayed by the lines that do not reach the VTR of 10^{-10}. On the Michalewicz problem, AMaLGaM struggles to find good results for increasingly large problem sizes. For a problem size of 80, all runs of GOMEA are stuck in a range of local minima, while AMaLGaM seems to have found a relatively good solution. However, for the problem sizes 320 and 1280, GOMEA clearly performed better than AMaLGaM, even though its performance is substantially worse than that of GOMEA using a univariate structure. GOMEA still seems to scale slightly better than AMaLGaM on the SoREB problem, but these results are inconclusive. Finally, on the sphere, Rosenbrock and Griewank functions, GOMEA with the full FOS clearly performed worse than AMaLGaM. Mostly the results of the Rosenbrock and Griewank functions are a cause for concern, because GOMEA converged prematurely for problem sizes as small as 40 and 20. Because GOMEA is expected to be able to solve these problems even using a full FOS structure, we studied the premature convergence and found that it was caused by the forced improvement technique. The forced improvement technique is applied to an individual when it has not improved during a single generation, and overwrites this individual with the elitist solution if the forced improvements are not successful. When using a full FOS structure, improvements seem to occur less frequently, causing forced improvements to be applied to some individuals that can still be improved through the regular sampling technique. Moreover, because there is only one FOS element, the forced improvements procedure is very brief and forces all variable values to immediately change to those of the elitist solution. This leads to a large number of copies of the elitist solution in the population, causing premature convergence due to diversity loss. To test whether indeed the forced improvements procedure is causing these unsuccessful runs, all experiments of GOMEA using a full FOS were repeated without the use of forced improvements, of which the results are displayed in Figure 6.1. These results show that GOMEA now performs better than AMaLGaM on the Rosenbrock function, and very similar on the Sphere and Griewank problem, clearly improving previous results for which forced improvements were used. Performance on the SoREB function even greatly improved by disabling forced improvements, achieving a speed-up of approximately a factor 100 on the 40-dimensional problem. Unfortunately, the results on the Michalewicz problem became worse than before, because GOMEA barely performed better than AMaLGaM on the problem with 1280 variables. In its current form within the real-valued GOMEA, as proposed in this thesis, there seem to be advantages and disadvantages to forced improvements. Adapting the forced improvement method to only be performed on individuals that have not improved for a certain number of consecutive generations is a potential resolution, but testing and tuning this adaptation requires additional research.

We then studied the efficiency of GOMEA in the case of black-box optimization, for which the results are shown in Figure 5.15. FOS structures previously found to be able to solve these problems efficiently were selected for these experiments. For the sphere, Rosenbrock, Griewank and Michalewicz problems, a univariate FOS structure was used. This means that, due to the black-box setting, GOMEA only modified one variable per evaluation, whereas AMaLGaM modified ℓ variables per evaluation. Two exceptions to this are the evaluation of the initial population, and evaluations of an individual after the final application of AMS. In both these cases, all variables are required to be reevaluated, requiring a single full evaluation. A 5-block FOS structure was used for the SoREB problem, meaning that GOMEA modified a block of 5 variables per evaluation.

In a black-box setting, when considering the number of evaluations, GOMEA was clearly able to perform better than AMaLGaM on the sphere, and Michalewicz problem, and achieved very similar results to AMaLGaM on the Rosenbrock, Griewank, and SoREB problems. In terms of time, GOMEA outperformed AMaLGaM on all considered benchmark problems. Even though the number of evaluations required by GOMEA and AMaLGaM are quite similar, GOMEA achieved a better performance in terms of time, because



(a) Sphere

(b) Rosenbrock

(c) Griewank

(d) Michalewicz

(e) SoREB

Figure 6.1: Experiments measuring time of GOMEA with no forced improvement method and AMaLGaM, both using a full FOS structure, for all considered benchmarks.

GOMEA performs fewer sampling operations per evaluation. In a black-box setting, when GOMEA and AMaLGaM are both using a univariate FOS structure, GOMEA performs $O(n\ell)$ evaluations per generation, whereas AMaLGaM performs $O(n)$ evaluations per generation. This means that GOMEA performed a lower number of sampling operations, which are relatively time-consuming compared to the evaluations of the benchmark problems we consider.

On the Michalewicz problem a substantial improvement of GOMEA over AMaLGaM can be observed. This is according to expectations, because the problem structure illustrated in Figure 5.1d shows that optimization in a single dimension can more easily guide the algorithm towards the optimum. In contrast to GOMEA, AMaLGaM barely even progressed towards the optimum for a problem size of 1280. For smaller problem sizes, it was able to get closer to the optimum, but it was still outperformed by GOMEA. The remarkable arc-like shapes of GOMEAs graphs are caused by the linear interpolation of too few data points, because a linear interpolation method is used to interpolate a relation that is clearly not linear. This effect is not notable on a small scale, but does become apparent when too few data points are collected. On the remaining benchmark problems, GOMEA and AMaLGaM performed similarly. GOMEA only shows slightly better scaling on the Griewank problem, but the large overlapping interdecile ranges leave us inconclusive. Considering that GOMEA had a greatly improved performance in the grey-box setting and still performs better or equally good in a black-box setting, we find that GOMEA is a very promising algorithm.

Finally, on the Rosenbrock problem, there is a remarkable difference in the success rates of GOMEA using a 2-block FOS in the black-box and the grey-box settings. The only difference between these settings is that small numerical errors in the objective value can be introduced by partial evaluations. This was indeed the cause of the reduced success rate, but only when the forced improvements procedure was used. Due to the numerical errors in the objective value of individuals, an individual can be believed to be of a higher quality than it actually is. Subsequently, different individuals will adopt the values of the variables of such an individual when the forced improvements procedure fails to improve their objective value. Re-evaluation of the population to remove numerical errors had no effect, even when the population was re-evaluated every 10 generations, because of an already substantial loss in diversity.

6.2. Multi-Objective Optimization

Results for each of the multi-objective benchmarks are displayed in Figure 5.16. These display either the $D_{\boldsymbol{P}_f \to \boldsymbol{S}}$ or the hypervolume achieved by GOMEA and MAMaLGaM within the one hour time limit. In terms of time, GOMEA clearly performed better than MAMaLGaM on each of the considered benchmark problems, obtaining an increasingly growing speed-up for problems with an increasing number of variables. GOMEA did have one prematurely converged run for the ZDT6 problem with 320 variables, and two prematurely converged runs for the ZDT6 problem with 1280 variables. Further research would have to determine the cause of this. GOMEA did not outperform AMaLGaM in terms of evaluations. However, considering that these experiments were performed in a black-box setting, GOMEA is very likely to require a lower number of evaluations on each of the benchmark problems in a grey-box setting.

6.3. Deformable Image Registration

The speed-up obtained with GOMEA was measured through the experiments displayed in Figures 5.17 and 5.18. A speed-up factor was calculated by checking the time required by the worst algorithm to achieve its maximum hypervolume, and comparing this to the time required by the other algorithm to reach the same hypervolume. This leads us to observe speed-ups of factors of 114, 102, and 1641 for the experiments on the artificial benchmark problem with increasing resolution, displayed in Figure 5.17. On the pre- and post-operative benchmark problem, displayed in Figure 5.18, we observe speed-ups of factors 0.25, 31, and 940, for runs with increasing resolution. The inferior performance of GOMEA on pre- and post-operative benchmark with resolution 6×6 could be caused by a difference in grid triangulations. In our new model used in GOMEA, the outer points of the grid are constrained to the border of the grid, while in the previous model used in AMaLGaM, these points are free to be moved further inwards. This effect is only notable on relatively small resolution grids, because the fraction of points constrained to the border decreases as the grid resolution increases.

In the experiment displayed in Figures 5.19 and 5.20, we evaluate the quality of deformation outcomes of GOMEA. This is done through the previously introduced TRE metric. Considering the Pareto fronts displayed in Figure 5.19 and their respective solutions with the minimal mean TRE, we confirm that solutions with a high deformation magnitude are often overfitted and describe unnatural deformations. Solutions with a slightly lower deformation magnitude are reported to have a better mean TRE. Additionally, the mean TRE of the best solution on each Pareto front is arguably within medically acceptable limits. Using higher resolution grids does not improve the minimal mean TRE much, but does increase the smoothness of the deformation. Moreover, a run that takes roughly 2.5 minutes achieves results that are very similar to the results of the full 40 minute run, and are also of a very similar quality as previously obtained results by iMAMaLGaM [4]. Especially considering that an entire Pareto front is generated, this is comparable to the runtime of state-of-the-art software for deformable image registration.

7 Conclusions and Future Work

We have studied the Gene-pool Optimal Mixing Evolutionary Algorithm (GOMEA) and developed three different adaptations of it, allowing GOMEA to be used, for the first time, on real-valued single-objective problems, multi-objective problems, and the deformable image registration problem while using a real-valued solution representation. Based on the results of the performed experiments, in this section, we draw a number of conclusions and introduce some topics that could be the subject of future research.

7.1. Conclusions

In a grey-box setting, GOMEA has been found to achieve a significant increase in performance on all single-objective and multi-objective benchmark problems that we considered, in terms of time and number of evaluations. According to expectations, a univariate FOS dependency structure was found to achieve the best performance on most considered benchmark problems, except for the SoREB and Sphere-SoREB problems, where the k-block marginal product FOS achieved the best performance.

Initially, in some experiments where GOMEA used a sub-optimal FOS, it was found to perform worse than AMaLGaM using the same FOS, because GOMEA often converged prematurely. However, after disabling GOMEA's forced improvements procedure and repeating these experiments, the performance of GOMEA was better than or very similar to AMaLGaM. The forced improvements method causes this decrease in efficiency, because an individual is less likely to improve if a sub-optimal FOS is used. This causes forced improvements to be applied to a relatively high number of individuals. Considering that the forced improvements phase is frequently ineffective, many copies of the elitist solution are copied into the population, causing the population to converge towards the elitist solution. The ineffectiveness of the forced improvements phase can be caused by the fact that this phase is very short when a FOS with few elements is used, or due to the linear combination of solutions leading to new solutions of relatively low quality.

In a black-box setting, GOMEA performed similarly to AMaLGaM on most problems, but clearly performed better than AMaLGaM on the sphere and Michalewicz problems. In terms of time, GOMEA in a black-box setting outperformed AMaLGaM on each single-objective and multi-objective benchmark problem. In terms of total number of evaluations required, GOMEA clearly outperformed AMaLGaM on the sphere and Michalewicz function, while achieving very similar results on the Rosenbrock, Griewank, and SoREB functions. The improved performance for the sphere and Michalewicz functions is caused by the fact that GOMEA can still exploit the decomposability of problems in a black-box setting, which seems to outweigh the disadvantage of expensive evaluations for problems that are highly decomposable. For the considered multi-objective benchmark problems in a black-box setting, GOMEA produced results comparable to that of AMaLGaM. This means that GOMEA is clearly worth considering for black-box optimization.

With regard to deformable image registration, we found that our problem-specific implementation of GOMEA is capable of obtaining a speed-up up to a factor of ~1600, while maintaining high quality solutions, on frequently used deformable image registration benchmark problems. This speed-up can mostly be attributed to the fact that deformable image registration is a decomposable problem, allowing GOMEA to efficiently and independently optimize independent subproblems. Using dual-dynamic transformation grids to support large deformations, we observed a similar quality of results to the previously used algorithm iMAMaLGaM, which was previously found to produce clinically desirable results [3, 4].

We have observed that, for the best performance of GOMEA, an adequate FOS dependency structure is required to capture the problem dependencies. A FOS structure that accurately models existing dependencies between variables allows GOMEA to mostly optimize these subsets of dependent variables separately. This mostly independent optimization of subsets of variables combined with an increased selection pressure leads to a smaller required population size, which substantially increases GOMEAs efficiency. We can therefore conclude that GOMEA is a very powerful algorithm for decomposable optimization problems, but the selection of an appropriate FOS for the problem at hand is of great importance. For this reason, using GOMEA should be an excellent choice for the optimization of any problem where a dependency structure can be roughly estimated, but where the structure of the optimization function is unknown.

7.2. Future Work

A key topic for future work is the extension of GOMEA to perform 3D deformable image registration. The very significant speed-up that is achieved by GOMEA paves the road for this to be possible. Solving 3D instances of deformable image registration inevitably requires an increasingly large number of variables compared to 2D, which would greatly decrease the performance of currently used algorithms.

Further problem-specific enhancements for deformable image registration may be possible. Firstly, calculations of the objective values could sample a selection of points instead of iterating over all pixels. In order to correctly calculate a normalized guidance objective, this approach would however require more problem specific adaptations. This is caused by the variable number of pixels that would be sampled on the guidance contours, which has to be accounted for. Alternatively, a method to efficiently iterate over all pixels of a guidance contour inside a certain triangle, would overcome this issue. Secondly, GOMEA's procedure of creating population clusters, i.e., assigning every individual in the population to a cluster, can possibly be improved. The currently used method has been observed to work well, but little research has been performed on alternative methods. Currently, each cluster is assigned a number of solutions in order to achieve a minimum size, after which remaining solutions are assigned to their closest cluster. Observations have shown that some solutions can be relatively far away from their cluster means, which can possible be improved. Finally, an additional substantial speed-up could be achieved by the application of smart grid initialization [8].

Future fundamental work on real-valued GOMEA could include the addition of the linkage tree model, which would remove the requirement of selecting a FOS by hand. With this model, GOMEA could be applied to a wide range of optimization problems in a black-box setting. The linkage tree model could also be useful for certain problems in a grey-box setting, mostly when partial evaluations are possible, but the exact dependency structure of the problem is unknown, e.g., the Max-Cut problem. The performance of GOMEA could be improved further by introducing a parallel version. To further expand the range of problems to which GOMEA can be applied, a mixed-integer version of GOMEA could be developed based on the work in [39]. Finally, the introduction of the real-valued GOMEA opens the door to the application of GOMEA to a potentially limitless supply of real-world problems dealing with real-valued variables.

Bibliography

[1] David Ackley. *A connectionist machine for genetic hillclimbing*, volume 28. Springer Science & Business Media, 2012.

[2] Tanja Alderliesten, Jan-Jakob Sonke, and Peter AN Bosman. Multi-objective optimization for deformable image registration: Proof of concept. In *SPIE Medical Imaging*, pages 831420–831420. International Society for Optics and Photonics, 2012.

[3] Tanja Alderliesten, Jan-Jakob Sonke, and Peter AN Bosman. Deformable image registration by multi-objective optimization using a dual-dynamic transformation model to account for large anatomical differences. In *SPIE Medical Imaging*, pages 866910–866910. International Society for Optics and Photonics, 2013.

[4] Tanja Alderliesten, Peter AN Bosman, and Arjan Bel. Getting the most out of additional guidance information in deformable image registration by leveraging multi-objective optimization. In *SPIE Medical Imaging*, pages 94131R–94131R. International Society for Optics and Photonics, 2015.

[5] Peter AN Bosman. On empirical memory design, faster selection of Bayesian factorizations and parameter-free Gaussian EDAs. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 389–396. ACM, 2009.

[6] Peter AN Bosman. The anticipated mean shift and cluster registration in mixture-based EDAs for multi-objective optimization. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 351–358. ACM, 2010.

[7] Peter AN Bosman and Tanja Alderliesten. Incremental Gaussian model-building in multi-objective EDAs with an application to deformable image registration. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation*, pages 241–248. ACM, 2012.

[8] Peter AN Bosman and Tanja Alderliesten. Smart grid initialization reduces the computational complexity of multi-objective image registration based on a dual-dynamic transformation model to account for large anatomical differences. In *SPIE Medical Imaging*, pages 978447–978447. International Society for Optics and Photonics, 2016.

[9] Peter AN Bosman and Dirk Thierens. Expanding from discrete to continuous estimation of distribution algorithms: The IDEA. In *Parallel Problem Solving from Nature PPSN VI*, pages 767–776. Springer, 2000.

[10] Peter AN Bosman and Dirk Thierens. The balance between proximity and diversity in multiobjective evolutionary algorithms. *IEEE transactions on evolutionary computation*, 7(2):174–188, 2003.

[11] Peter AN Bosman and Dirk Thierens. Linkage neighbors, optimal mixing and forced improvements in genetic algorithms. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation*, pages 585–592. ACM, 2012.

[12] Peter AN Bosman, Jörn Grahl, and Dirk Thierens. Adapted maximum-likelihood Gaussian models for numerical optimization with continuous EDAs. *Software Engineering [SEN]*, (E0704):1–20, 2007.

[13] Peter AN Bosman, Jörn Grahl, and Dirk Thierens. AMaLGaM IDEAs in noiseless black-box optimization benchmarking. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, pages 2247–2254. ACM, 2009.

[14] Peter AN Bosman, Jörn Grahl, and Dirk Thierens. Benchmarking parameter-free AMaLGaM on functions with and without noise. *Evolutionary computation*, 21(3):445–469, 2013.

[15] Carlos A Coello Coello, David A Van Veldhuizen, and Gary B Lamont. *Evolutionary algorithms for solving multi-objective problems*, volume 242. Springer, 2002.

[16] Mark Foskey, Brad Davis, Lav Goyal, Sha Chang, Ed Chaney, Nathalie Strehl, Sandrine Tomei, Julian Rosenman, and Sarang Joshi. Large deformation three-dimensional image registration in image-guided radiation therapy. *Physics in Medicine and Biology*, 50(24):5869, 2005.

[17] David E Goldberg. Simple genetic algorithms and the minimal, deceptive problem. *Genetic algorithms and simulated annealing*, 74:88, 1987.

[18] David E Goldberg and Jon Richardson. Genetic algorithms with sharing for multimodal function optimization. In *Genetic algorithms and their applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 41–49. Hillsdale, NJ: Lawrence Erlbaum, 1987.

[19] Jörn Grahl, Peter AN Bosman, and Franz Rothlauf. The correlation-triggered adaptive variance scaling IDEA. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 397–404. ACM, 2006.

[20] Nikolaus Hansen and Andreas Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*, pages 312–317. IEEE, 1996.

[21] Nikolaus Hansen and Andreas Ostermeier. Convergence properties of evolution strategies with the derandomized covariance matrix adaptation: The $(\mu/\mu_I, \lambda)$-CMA-ES. *Eufit*, 97:650–654, 1997.

[22] Georges Harik. Linkage learning via probabilistic modeling in the ecga. *Urbana*, 51(61):801, 1999.

[23] Georges R Harik and Fernando G Lobo. A parameter-less genetic algorithm. In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 1*, pages 258–265. Morgan Kaufmann Publishers Inc., 1999.

[24] John H Holland. Genetic algorithms. *Scientific american*, 267(1):66–72, 1992.

[25] Stefan Klein, Marius Staring, Keelin Murphy, Max A Viergever, and Josien PW Pluim. `elastix`: A toolbox for intensity-based medical image registration. *IEEE transactions on medical imaging*, 29(1):196–205, 2010.

[26] Joshua Knowles and David Corne. On metrics for comparing nondominated sets. In *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*, volume 1, pages 711–716. IEEE, 2002.

[27] Pedro Larrañaga, Ramón Etxeberria, Jose A Lozano, and José M Peña. Optimization in continuous domains by learning and simulation of Gaussian networks. 2000.

[28] Ngoc Hoang Luong, Han La Poutré, and Peter AN Bosman. Multi-objective gene-pool optimal mixing evolutionary algorithms. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, pages 357–364. ACM, 2014.

[29] Heinz Mühlenbein. The equation for response to selection and its use for prediction. *Evolutionary Computation*, 5(3):303–346, 1997.

[30] Heinz Mühlenbein and Gerhard Paass. From recombination of genes to the estimation of distributions I. binary parameters. In *International Conference on Parallel Problem Solving from Nature*, pages 178–187. Springer, 1996.

[31] Martin Pelikan. Bayesian optimization algorithm. In *Hierarchical Bayesian optimization algorithm*, pages 31–48. Springer, 2005.

[32] Martin Pelikan. Hierarchical Bayesian optimization algorithm. In *Hierarchical Bayesian Optimization Algorithm*, pages 105–129. Springer, 2005.

[33] Martin Pelikan and Heinz Mühlenbein. The bivariate marginal distribution algorithm. In *Advances in Soft Computing*, pages 521–535. Springer, 1999.

[34] Martin Pelikan, David E Goldberg, and Erick Cantu-Paz. Linkage problem, distribution estimation, and Bayesian networks. *Evolutionary computation*, 8(3):311–340, 2000.

[35] José C Pereira and Fernando G Lobo. A java implementation of parameter-less evolutionary algorithms. *arXiv preprint arXiv:1506.08694*, 2015.

[36] Kleopatra Pirpinia, Peter AN Bosman, Jan-Jakob Sonke, Marcel van Herk, and Tanja Alderliesten. A first step toward uncovering the truth about weight tuning in deformable image registration. In *SPIE Medical Imaging*, pages 978445–978445. International Society for Optics and Photonics, 2016.

[37] Sílvio Rodrigues, Pavol Bauer, and Peter AN Bosman. A novel population-based multi-objective CMA-ES and the impact of different constraint handling techniques. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, pages 991–998. ACM, 2014.

[38] Daniel Rueckert, Luke I Sonoda, Carmel Hayes, Derek LG Hill, Martin O Leach, and David J Hawkes. Nonrigid registration using free-form deformations: Application to breast MR images. *IEEE transactions on medical imaging*, 18(8):712–721, 1999.

[39] Krzysztof L Sadowski, Peter AN Bosman, and Dirk Thierens. A clustering-based model-building EA for optimization problems with binary and real-valued variables. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 911–918. ACM, 2015.

[40] Nidamarthi Srinivas and Kalyanmoy Deb. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary computation*, 2(3):221–248, 1994.

[41] Dirk Thierens. Scalability problems of simple genetic algorithms. *Evolutionary computation*, 7(4):331–352, 1999.

[42] Dirk Thierens and Peter AN Bosman. Optimal mixing evolutionary algorithms. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 617–624. ACM, 2011.

[43] Medha V Wyawahare, Pradeep M Patil, Hemant K Abhyankar, et al. Image registration techniques: An overview. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 2(3): 11–28, 2009.

Appendices

A

Tables of Single-Objective Benchmarking Experiments

		GOMEA GBO						GOMEA BBO						AMaLGaM-FOS					
		Univariate			Full			Univariate			Full			Univariate			Full		
		90p	med	10p	90p	med	10p	90p	med	10p	90p	med	10p	90p	med	10p	90p	med	10p
Sphere	20	6.00e-03	7.50e-03	9.00e-03	5.10e-02	5.75e-02	6.50e-02	3.00e-03	4.00e-03	4.00e-03	2.60e-02	3.05e-02	3.80e-02	1.25e-01	1.56e-01	2.65e-01	2.72e-02	4.63e-02	5.56e-02
	40	1.40e-02	1.60e-02	1.90e-02	2.28e-01	3.43e-01	7.70e+00	8.00e-03	9.00e-03	1.00e-02	2.02e-01	2.29e-01	2.64e-01	5.69e-01	6.58e-01	9.00e-01	2.78e-01	3.15e-01	4.79e-01
	80	2.70e-02	3.35e-02	3.80e-02	2.56e+00	3.15e+00	4.27e+02	1.90e-02	2.20e-02	2.60e-02	2.68e+00	3.20e+00	3.73e+00	2.51e+00	2.60e+00	2.82e+00	1.69e+00	1.85e+00	2.45e+00
	160	5.60e-02	6.35e-02	7.30e-02	2.73e+01	2.97e+01	3.37e+01	5.30e-02	6.00e-02	6.70e-02	2.86e+01	3.45e+01	3.89e+01	6.75e+00	7.10e+00	1.34e+01	1.11e+01	1.76e+01	2.51e+01
	320	1.08e-01	1.34e-01	1.43e-01	4.51e+02	6.31e+02	9.58e+02	1.72e-01	1.82e-01	2.04e-01	5.43e+02	8.59e+02	1.05e+03	2.50e+01	2.70e+01	2.98e+01	1.40e+02	2.15e+02	4.97e+02
	640	1.69e-01	2.27e-01	2.43e-01	-	-	-	5.78e-01	6.17e-01	6.92e-01	-	-	-	1.22e+02	1.39e+02	1.48e+02	1.51e+03	1.91e+03	2.37e+03
	1280	3.63e-01	3.98e-01	4.59e-01	-	-	-	2.24e+00	2.38e+00	2.57e+00	-	-	-	6.44e+02	6.96e+02	7.55e+02	-	-	-
	2560	7.93e-01	8.83e-01	9.66e-01	-	-	-	8.76e+00	9.61e+00	1.05e+01	-	-	-	1.84e+03	3.08e+03	3.18e+03	-	-	-
	5120	1.85e+00	1.99e+00	2.10e+00	-	-	-	3.82e+01	4.10e+01	4.49e+01	-	-	-	-	-	-	-	-	-
	10240	4.19e+00	4.44e+00	4.80e+00	-	-	-	1.65e+02	1.78e+02	1.92e+02	-	-	-	-	-	-	-	-	-

Table A.1: Time until the VTR of 10^{-10} for the sphere function, for $\ell \in [20, 40, \ldots, 10240]$. Results are marked in orange for a success rate p^{succ} within the range $50\% \leq p^{\text{suc}} < 90\%$, marked in red for a success rate within the range $10\% \leq p^{\text{suc}} < 50\%$, and not displayed for $p^{\text{suc}} < 10\%$.

		GOMEA GBO						GOMEA BBO						AMaLGaM-FOS					
		Univariate			Full			Univariate			Full			Univariate			Full		
		90p	med	10p	90p	med	10p	90p	med	10p	90p	med	10p	90p	med	10p	90p	med	10p
Sphere	20	7.94e+02	9.52e+02	1.05e+03	2.22e+04	2.50e+04	2.88e+04	1.32e+04	1.59e+04	1.70e+04	2.20e+04	2.56e+04	3.22e+04	4.90e+04	6.11e+04	8.22e+04	1.54e+04	2.50e+04	3.00e+04
	40	9.58e+02	1.08e+03	1.20e+03	7.89e+04	8.92e+04	4.97e+06	3.16e+04	3.51e+04	4.04e+04	7.51e+04	8.49e+04	9.81e+04	9.88e+04	1.43e+05	1.54e+05	6.02e+04	6.94e+04	1.03e+05
	80	1.10e+03	1.23e+03	1.38e+03	3.26e+05	4.03e+05	2.93e+07	7.10e+04	8.03e+04	9.50e+04	3.35e+05	3.95e+05	4.64e+05	2.89e+05	3.02e+05	3.28e+05	1.21e+05	1.34e+05	1.79e+05
	160	1.25e+03	1.43e+03	1.60e+03	9.63e+05	1.04e+06	1.18e+06	1.56e+05	1.75e+05	1.97e+05	9.46e+05	1.14e+06	1.24e+06	4.11e+05	4.33e+05	7.74e+05	2.42e+05	3.73e+05	5.37e+05
	320	1.48e+03	1.60e+03	1.98e+03	4.12e+06	5.91e+06	7.25e+06	3.68e+05	3.90e+05	4.35e+05	4.23e+06	1.03e+07	1.30e+07	7.99e+05	8.58e+05	9.34e+05	8.66e+05	1.25e+06	2.91e+06
	640	1.71e+03	2.04e+03	2.64e+03	-	-	-	7.88e+05	8.39e+05	9.40e+05	-	-	-	1.99e+06	2.25e+06	2.40e+06	2.13e+06	2.58e+06	3.27e+06
	1280	1.88e+03	2.60e+03	3.84e+03	-	-	-	1.76e+06	1.87e+06	2.02e+06	-	-	-	5.37e+06	5.73e+06	6.05e+06	-	-	-
	2560	1.90e+03	2.95e+03	5.20e+03	-	-	-	3.72e+06	4.09e+06	4.47e+06	-	-	-	6.72e+06	1.27e+07	1.29e+07	-	-	-
	5120	2.70e+03	4.32e+03	1.10e+04	-	-	-	8.47e+06	9.07e+06	9.95e+06	-	-	-	-	-	-	-	-	-
	10240	2.76e+03	7.16e+03	1.83e+04	-	-	-	1.80e+07	1.96e+07	2.09e+07	-	-	-	-	-	-	-	-	-

Table A.2: Number of evaluations until the VTR of 10^{-10} for the sphere function, for $\ell \in [20, 40, \ldots, 10240]$. Results are marked in orange for a success rate p^{succ} within the range $50\% \leq p^{\text{suc}} < 90\%$, marked in red for a success rate within the range $10\% \leq p^{\text{suc}} < 50\%$, and not displayed for $p^{\text{suc}} < 10\%$.

		GOMEA GBO						GOMEA BBO						AMaLGaM-FOS					
		Univariate			Full			Univariate			Full			Univariate			Full		
		90p	med	10p	90p	med	10p	90p	med	10p	90p	med	10p	90p	med	10p	90p	med	10p
Rosenbrock	20	1.88e-01	2.77e-01	4.34e-01	2.04e-01	3.03e-01	3.72e-01	1.34e-01	2.71e-01	5.16e-01	2.12e-01	2.82e-01	4.19e-01	1.82e+00	4.55e+00	5.82e+00	1.38e+00	2.36e+00	3.84e+00
	40	2.24e-01	4.39e-01	6.15e-01	1.33e+00	1.57e+00	1.91e+00	3.02e-01	4.76e-01	8.88e-01	1.44e+00	1.69e+00	2.28e+00	4.23e+00	4.92e+00	1.25e+01	1.24e+01	1.37e+01	3.47e+01
	80	5.71e-01	1.06e+00	2.30e+00	1.25e+01	1.63e+01	5.20e+01	8.69e-01	1.50e+00	2.14e+00	1.43e+01	1.82e+01	5.72e+01	1.44e+01	2.01e+01	4.01e+01	8.75e+01	9.59e+01	1.21e+02
	160	1.50e+00	2.14e+00	3.72e+00	1.42e+02	2.44e+02	5.72e+02	3.04e+00	4.91e+00	1.06e+01	1.92e+02	2.73e+02	1.16e+03	4.27e+01	6.57e+01	1.09e+02	9.03e+02	9.35e+02	1.05e+03
	320	3.10e+00	4.29e+00	5.72e+00	-	-	-	1.14e+01	1.60e+01	2.03e+01	-	-	-	1.81e+02	4.74e+02	7.51e+02	-	-	-
	640	6.35e+00	7.73e+00	1.35e+01	-	-	-	4.28e+01	5.19e+01	1.97e+02	-	-	-	6.35e+02	7.44e+02	2.00e+03	-	-	-
	1280	1.27e+01	1.67e+01	2.22e+01	-	-	-	1.58e+02	2.01e+02	3.29e+02	-	-	-	2.66e+03	2.96e+03	3.56e+03	-	-	-
	2560	2.53e+01	3.79e+01	5.29e+01	-	-	-	5.77e+02	7.67e+02	1.30e+03	-	-	-	-	-	-	-	-	-
	5120	6.21e+01	8.17e+01	3.25e+02	-	-	-	2.42e+03	2.84e+03	3.56e+03	-	-	-	-	-	-	-	-	-
	10240	1.34e+02	1.69e+02	2.32e+02	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Table A.3: Time until the VTR of 10^{-10} for the Rosenbrock function, for $\ell \in [20, 40, \ldots, 10240]$. Results are marked in orange for a success rate p^{succ} within the range $50\% \leq p^{\text{suc}} < 90\%$, marked in red for a success rate within the range $10\% \leq p^{\text{suc}} < 50\%$, and not displayed for $p^{\text{suc}} < 10\%$.

		GOMEA GBO						GOMEA BBO						AMaLGaM-FOS					
		Univariate			Full			Univariate			Full			Univariate			Full		
		90p	med	10p	90p	med	10p	90p	med	10p	90p	med	10p	90p	med	10p	90p	med	10p
Rosenbrock	20	5.04e+04	7.50e+04	1.34e+05	1.57e+05	2.30e+05	2.68e+05	5.93e+05	1.12e+06	2.88e+06	1.62e+05	2.19e+05	2.93e+05	8.19e+05	1.93e+06	2.03e+06	7.38e+05	1.36e+06	2.36e+06
	40	3.31e+04	5.89e+04	8.73e+04	4.60e+05	5.40e+05	6.58e+05	1.16e+06	1.75e+06	3.25e+06	5.00e+05	5.60e+05	6.79e+05	1.01e+06	1.15e+06	2.91e+06	3.12e+06	3.37e+06	9.00e+06
	80	4.39e+04	6.78e+04	2.18e+05	1.51e+06	1.97e+06	1.03e+07	2.64e+06	4.44e+06	6.36e+06	1.68e+06	2.11e+06	1.26e+07	1.75e+06	2.49e+06	4.64e+06	7.58e+06	8.13e+06	1.06e+07
	160	5.03e+04	7.16e+04	1.38e+05	4.85e+06	1.16e+07	3.20e+07	6.36e+06	9.62e+06	1.37e+07	6.04e+06	1.27e+07	8.16e+07	2.48e+06	4.10e+06	6.88e+06	2.25e+07	2.32e+07	2.47e+07
	320	5.34e+04	7.25e+04	9.59e+04	-	-	-	1.51e+07	2.09e+07	2.69e+07	-	-	-	5.93e+06	1.52e+07	2.46e+07	-	-	-
	640	5.56e+04	6.62e+04	1.17e+05	-	-	-	3.28e+07	3.93e+07	1.59e+08	-	-	-	1.11e+07	1.30e+07	3.22e+07	-	-	-
	1280	5.67e+04	7.26e+04	9.52e+04	-	-	-	6.32e+07	7.92e+07	1.37e+08	-	-	-	2.48e+07	2.79e+07	3.34e+07	-	-	-
	2560	5.43e+04	8.20e+04	1.11e+05	-	-	-	1.17e+08	1.59e+08	2.51e+08	-	-	-	-	-	-	-	-	-
	5120	6.63e+04	8.74e+04	4.85e+05	-	-	-	2.65e+08	3.13e+08	3.58e+08	-	-	-	-	-	-	-	-	-
	10240	7.27e+04	9.11e+04	1.23e+05	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Table A.4: Number of evaluations until the VTR of 10^{-10} for the Rosenbrock function, for $\ell \in [20, 40, \ldots, 10240]$. Results are marked in orange for a success rate p^{succ} within the range $50\% \leq p^{\text{suc}} < 90\%$, marked in red for a success rate within the range $10\% \leq p^{\text{suc}} < 50\%$, and not displayed for $p^{\text{suc}} < 10\%$.

		GOMEA BBO						AMaLGaM-FOS					
		Univariate			Full			Univariate			Full		
		90p	med	10p	90p	med	10p	90p	med	10p	90p	med	10p
Griewank	20	2.00e-02	2.06e-01	5.59e-01	-	-	-	2.09e-01	4.41e-01	7.00e-01	6.76e-02	1.14e-01	2.19e-01
	40	7.70e-02	3.95e-01	1.47e+00	3.55e-01	4.19e-01	6.52e-01	1.11e+00	1.21e+00	2.47e+00	4.12e-01	6.48e-01	7.37e-01
	80	3.47e-01	1.26e+00	5.27e+00	2.82e+00	3.53e+00	5.90e+00	3.45e+00	6.03e+00	1.25e+01	2.16e+00	2.63e+00	4.93e+00
	160	1.05e+00	5.14e+00	1.06e+01	-	-	-	9.48e+00	1.55e+01	4.13e+01	1.88e+01	2.83e+01	4.52e+01
	320	4.24e+00	2.14e+01	6.70e+01	4.52e+02	4.96e+02	2.26e+03	3.51e+01	4.95e+01	1.74e+02	1.46e+02	2.43e+02	4.95e+02
	640	1.82e+01	7.16e+01	1.26e+02	-	-	-	1.89e+02	2.74e+02	4.44e+02	1.84e+03	1.91e+03	2.62e+03
	1280	7.22e+01	7.99e+01	4.70e+02	-	-	-	6.70e+02	9.01e+02	2.03e+03	-	-	-
	2560	2.79e+02	3.39e+02	1.98e+03	-	-	-	-	-	-	-	-	-
	5120	1.28e+03	1.38e+03	1.51e+03	-	-	-	-	-	-	-	-	-
	10240	-	-	-	-	-	-	-	-	-	-	-	-

Table A.5: Time until the VTR of 10^{-10} for the Griewank function, for $\ell \in [20, 40, \ldots, 10240]$. Results are marked in yellow for a success rate p^{succ} within the range $90\% \leq p^{\text{suc}} < 100\%$, marked in orange for a success rate within the range $50\% \leq p^{\text{suc}} < 90\%$, marked in red for a success rate within the range $10\% \leq p^{\text{suc}} < 50\%$, and not displayed for $p^{\text{suc}} < 10\%$.

		GOMEA GBO						GOMEA BBO						AMaLGaM-FOS					
		Univariate			Full			Univariate			Full			Univariate			Full		
		90p	med	10p	90p	med	10p	90p	med	10p	90p	med	10p	90p	med	10p	90p	med	10p
Griewank	20	1.00e+04	2.41e+04	8.77e+04	2.44e+04	2.28e+05	7.00e+05	2.43e+04	3.74e+05	1.25e+06	-	-	-	6.68e+04	1.33e+05	1.85e+05	2.48e+04	4.31e+04	9.14e+04
	40	1.45e+03	1.02e+04	2.20e+04	7.03e+04	1.20e+05	7.72e+07	5.38e+04	4.38e+05	1.72e+06	9.07e+04	1.01e+05	1.64e+05	1.88e+05	2.07e+05	4.30e+05	6.15e+04	1.03e+05	1.23e+05
	80	2.26e+03	1.17e+04	4.18e+04	2.00e+06	2.26e+06	2.97e+06	1.43e+05	7.90e+05	3.11e+06	2.89e+05	3.59e+05	6.21e+05	3.01e+05	5.34e+05	1.08e+06	1.29e+05	1.59e+05	3.30e+05
	160	1.36e+03	1.16e+04	5.02e+04	8.17e+05	1.16e+06	1.81e+07	2.03e+05	1.67e+06	3.67e+06	-	-	-	4.12e+05	6.88e+05	1.81e+06	3.63e+05	5.55e+05	9.13e+05
	320	1.52e+03	1.23e+04	3.80e+04	6.22e+06	9.80e+06	1.49e+07	4.03e+05	3.36e+06	1.08e+07	3.88e+06	4.25e+06	3.27e+07	7.81e+05	1.12e+06	3.79e+06	8.49e+05	1.35e+06	2.85e+06
	640	1.68e+03	1.12e+04	1.64e+04	-	-	-	8.87e+05	5.69e+06	1.07e+07	-	-	-	2.17e+06	2.53e+06	4.73e+06	2.39e+06	2.50e+06	3.64e+06
	1280	1.85e+03	3.01e+03	2.43e+04	-	-	-	1.75e+06	1.98e+06	1.93e+07	-	-	-	4.25e+06	5.92e+06	1.29e+07	-	-	-
	2560	1.95e+03	3.11e+03	1.71e+04	-	-	-	3.52e+06	4.12e+06	4.56e+07	-	-	-	-	-	-	-	-	-
	5120	2.64e+03	5.70e+03	9.36e+03	-	-	-	8.13e+06	8.60e+06	9.49e+06	-	-	-	-	-	-	-	-	-
	10240	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Table A.6: Number of evaluations until the VTR of 10^{-10} for the Griewank function, for $\ell \in [20, 40, \ldots, 10240]$. Results are marked in yellow for a success rate p^{succ} within the range $90\% \leq p^{\text{suc}} < 100\%$, marked in orange for a success rate within the range $50\% \leq p^{\text{suc}} < 90\%$, marked in red for a success rate within the range $10\% \leq p^{\text{suc}} < 50\%$, and not displayed for $p^{\text{suc}} < 10\%$.

		GOMEA GBO						GOMEA BBO						AMaLGaM-FOS					
		Univariate			Full			Univariate			Full			Univariate			Full		
		90p	med	10p	90p	med	10p	90p	med	10p	90p	med	10p	90p	med	10p	90p	med	10p
Michalewicz	20	-1.964e+01	-1.963e+01	-1.963e+01	-1.746e+01	-1.672e+01	-1.549e+01	-1.964e+01	-1.964e+01	-1.963e+01	-1.800e+01	-1.670e+01	-1.513e+01	-1.948e+01	-1.939e+01	-1.933e+01	-1.949e+01	-1.938e+01	-1.930e+01
	40	-3.963e+01	-3.962e+01	-3.958e+01	-3.538e+01	-3.382e+01	-3.103e+01	-3.962e+01	-3.960e+01	-3.957e+01	-3.568e+01	-3.294e+01	-3.069e+01	-3.923e+01	-3.911e+01	-3.889e+01	-3.934e+01	-3.914e+01	-3.894e+01
	80	-7.962e+01	-7.958e+01	-7.957e+01	-7.266e+01	-6.799e+01	-6.208e+01	-7.953e+01	-7.943e+01	-7.931e+01	-7.168e+01	-6.767e+01	-6.414e+01	-7.880e+01	-7.827e+01	-7.787e+01	-7.880e+01	-7.862e+01	-7.831e+01
	160	-1.596e+02	-1.596e+02	-1.595e+02	-1.421e+02	-1.384e+02	-1.324e+02	-1.588e+02	-1.583e+02	-1.579e+02	-1.435e+02	-1.381e+02	-1.266e+02	-1.175e+02	-1.028e+02	-8.640e+01	-1.224e+02	-1.167e+02	-1.080e+02
	320	-3.196e+02	-3.195e+02	-3.195e+02	-2.894e+02	-2.794e+02	-2.710e+02	-3.173e+02	-3.167e+02	-3.162e+02	-2.890e+02	-2.802e+02	-2.718e+02	-2.250e+02	-1.670e+02	-5.662e+01	-2.234e+02	-2.135e+02	-8.933e+01
	640	-6.395e+02	-6.394e+02	-6.392e+02	-4.934e+02	-3.835e+02	-2.564e+02	-6.329e+02	-6.001e+02	-5.952e+02	-5.175e+02	-4.276e+02	-2.391e+02	-3.475e+02	-1.095e+02	-9.950e+01	-3.503e+02	-1.414e+02	-9.709e+01
	1280	-1.279e+03	-1.279e+03	-1.279e+03	-5.205e+02	-4.153e+02	-3.375e+02	-1.213e+03	-1.207e+03	-1.199e+03	-5.025e+02	-4.132e+02	-3.460e+02	-2.932e+02	-1.999e+02	-1.813e+02	-2.462e+02	-1.772e+02	-1.746e+02
	2560	-2.555e+03	-2.554e+03	-2.554e+03	-7.637e+02	-5.527e+02	-5.159e+02	-2.439e+03	-2.422e+03	-2.410e+03	-9.044e+02	-5.664e+02	-5.195e+02	-4.293e+02	-3.422e+02	-3.363e+02	-3.341e+02	-3.303e+02	-3.269e+02
	5120	-5.110e+03	-5.109e+03	-5.108e+03	-1.125e+03	-9.667e+02	-9.346e+02	-4.164e+03	-4.124e+03	-4.074e+03	-1.086e+03	-9.869e+02	-9.363e+02	-6.519e+02	-6.451e+02	-6.395e+02	-6.351e+02	-6.285e+02	-6.227e+02
	10240	-1.015e+04	-1.015e+04	-1.014e+04	-1.916e+03	-1.833e+03	-1.789e+03	-7.034e+03	-6.970e+03	-6.936e+03	-1.972e+03	-1.810e+03	-1.779e+03	-1.255e+03	-1.246e+03	-1.240e+03	-1.229e+03	-1.215e+03	-1.207e+03

Table A.7: Objective values reached after min$(5\ell, 3600)$ seconds for the Michalewicz function, for $\ell \in [20, 40, \ldots, 10240]$.

		GOMEA GBO						GOMEA BBO						AMaLGaM-FOS					
		Univariate			Full			Univariate			Full			Univariate			Full		
		90p	med	10p	90p	med	10p	90p	med	10p	90p	med	10p	90p	med	10p	90p	med	10p
SoREB	20	6.37e+02	6.73e+02	7.02e+02	1.07e+01	1.28e+01	1.70e+01	1.14e+03	1.27e+03	1.38e+03	1.11e+01	1.21e+01	1.65e+01	1.31e+03	1.55e+03	2.82e+03	3.01e+00	5.06e+00	3.18e+01
	40	1.54e+03	2.32e+03	2.95e+03	7.65e+01	9.03e+01	1.17e+02	-	-	-	8.12e+01	9.55e+01	7.26e+02	-	-	-	4.45e+01	4.34e+02	5.11e+02
	80	-	-	-	5.55e+02	6.20e+02	1.61e+03	-	-	-	5.78e+02	6.66e+02	2.08e+03	-	-	-	2.72e+02	1.82e+03	2.46e+03
	160	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	320	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Table A.8: Time until the VTR of 10^{-10} for the SoREB function, for $\ell \in [20, 40, \ldots, 320]$. For larger problem instances, all algorithms were unable to reach the VTR within one hour. Results are marked in orange for a success rate p^{succ} within the range $50\% \leq p^{\text{suc}} < 90\%$, marked in red for a success rate within the range $10\% \leq p^{\text{suc}} < 50\%$, and not displayed for $p^{\text{suc}} < 10\%$.

		GOMEA GBO						GOMEA BBO						AMaLGaM-FOS					
		Univariate			Full			Univariate			Full			Univariate			Full		
		90p	med	10p	90p	med	10p	90p	med	10p	90p	med	10p	90p	med	10p	90p	med	10p
SoREB	20	1.67e+08	1.75e+08	1.79e+08	3.57e+06	4.26e+06	4.71e+06	6.15e+08	6.74e+08	7.00e+08	3.51e+06	4.07e+06	4.60e+06	3.45e+08	4.34e+08	7.94e+08	7.40e+05	1.26e+06	6.76e+06
	40	2.08e+08	3.15e+08	3.84e+08	1.19e+07	1.39e+07	1.62e+07	-	-	-	1.26e+07	1.41e+07	5.46e+07	-	-	-	5.43e+06	5.45e+07	5.67e+07
	80	-	-	-	3.83e+07	4.11e+07	1.13e+08	-	-	-	3.96e+07	4.32e+07	1.43e+08	-	-	-	1.12e+07	9.19e+07	1.14e+08
	160	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	320	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Table A.9: Number of evaluations until the VTR of 10^{-10} for the SoREB function, for $\ell \in [20, 40, \ldots, 320]$. For larger problem instances, all algorithms were unable to reach the VTR within one hour. Results are marked in orange for a success rate p^{succ} within the range $50\% \leq p^{\text{suc}} < 90\%$, marked in red for a success rate within the range $10\% \leq p^{\text{suc}} < 50\%$, and not displayed for $p^{\text{suc}} < 10\%$.

		GOMEA GBO			GOMEA BBO			AMaLGaM-FOS		
		2-Block			2-Block			2-Block		
		90p	med	10p	90p	med	10p	90p	med	10p
Rosenbrock	20	1.88e-01	2.87e-01	4.71e-01	2.01e-01	3.32e-01	3.65e+00	4.16e-01	1.38e+00	2.17e+00
	40	3.93e-01	5.97e-01	1.28e+00	5.70e-01	1.07e+00	3.22e+00	1.79e+00	3.64e+00	6.71e+00
	80	8.77e-01	1.46e+00	2.73e+00	1.50e+00	2.89e+00	1.50e+01	6.84e+00	1.15e+01	2.83e+01
	160	2.83e+00	3.61e+00	5.79e+00	4.65e+00	6.54e+00	1.82e+01	3.86e+01	4.74e+01	1.18e+02
	320	6.11e+00	8.19e+00	6.63e+01	1.78e+01	2.88e+01	1.77e+02	1.28e+02	1.87e+02	4.64e+02
	640	1.49e+01	2.41e+01	8.22e+01	7.98e+01	1.14e+02	5.32e+02	5.49e+02	1.38e+03	1.99e+03
	1280	4.20e+01	4.84e+01	2.32e+02	3.15e+02	4.34e+02	2.21e+03	-	-	-
	2560	9.63e+01	3.84e+02	8.19e+02	1.21e+03	1.47e+03	1.79e+03	-	-	-
	5120	2.04e+02	2.65e+02	2.38e+03	-	-	-	-	-	-
	10240	3.87e+02	4.75e+02	3.21e+03	-	-	-	-	-	-

Table A.10: Time until the VTR of 10^{-10} for the Rosenbrock function, for $\ell \in [20, 40, \ldots, 10240]$. Results are marked in yellow for a success rate p^{succ} within the range $90\% \leq p^{\text{suc}} < 100\%$, marked in orange for a success rate within the range $50\% \leq p^{\text{suc}} < 90\%$, marked in red for a success rate within the range $10\% \leq p^{\text{suc}} < 50\%$, and not displayed for $p^{\text{suc}} < 10\%$.

		GOMEA GBO			GOMEA BBO			AMaLGaM-FOS		
		2-Block			2-Block			2-Block		
		90p	med	10p	90p	med	10p	90p	med	10p
Rosenbrock	20	7.49e+04	1.21e+05	2.44e+05	6.31e+05	1.28e+06	1.89e+07	2.45e+05	8.01e+05	1.24e+06
	40	9.10e+04	1.28e+05	3.10e+05	1.78e+06	3.24e+06	1.54e+07	5.82e+05	1.13e+06	2.04e+06
	80	9.26e+04	1.58e+05	4.01e+05	3.74e+06	7.35e+06	5.43e+07	1.19e+06	1.86e+06	4.05e+06
	160	1.46e+05	1.93e+05	4.22e+05	8.89e+06	1.24e+07	4.21e+07	3.36e+06	4.04e+06	9.05e+06
	320	1.61e+05	2.14e+05	2.98e+06	2.18e+07	3.60e+07	2.52e+08	5.69e+06	8.38e+06	1.96e+07
	640	1.94e+05	3.09e+05	1.52e+06	5.81e+07	8.34e+07	4.14e+08	1.26e+07	3.01e+07	4.06e+07
	1280	2.64e+05	3.09e+05	1.83e+06	1.30e+08	1.82e+08	9.02e+08	-	-	-
	2560	3.01e+05	1.31e+06	2.23e+06	2.83e+08	3.44e+08	4.17e+08	-	-	-
	5120	3.18e+05	4.02e+05	3.73e+06	-	-	-	-	-	-
	10240	3.11e+05	3.76e+05	3.04e+06	-	-	-	-	-	-

Table A.11: Number of evaluations until the VTR of 10^{-10} for the Rosenbrock function, for $\ell \in [20, 40, \ldots, 10240]$. Results are marked in yellow for a success rate p^{succ} within the range $90\% \leq p^{\text{suc}} < 100\%$, marked in orange for a success rate within the range $50\% \leq p^{\text{suc}} < 90\%$, marked in red for a success rate within the range $10\% \leq p^{\text{suc}} < 50\%$, and not displayed for $p^{\text{suc}} < 10\%$.

		GOMEA GBO			GOMEA BBO			AMaLGaM-FOS		
		5-Block			5-Block			5-Block		
		90p	med	10p	90p	med	10p	90p	med	10p
SoREB	20	1.60e-01	2.33e-01	2.86e-01	3.04e-01	4.34e-01	4.95e-01	2.14e-01	5.17e-01	1.36e+00
	40	4.82e-01	7.54e-01	1.05e+00	1.40e+00	2.23e+00	2.92e+00	2.42e+00	3.09e+00	6.73e+00
	80	1.78e+00	1.88e+00	4.34e+00	9.61e+00	1.03e+01	2.40e+01	1.62e+01	2.34e+01	2.66e+01
	160	4.13e+00	4.25e+00	4.54e+00	4.30e+01	4.60e+01	4.90e+01	7.80e+01	1.24e+02	2.42e+02
	320	9.25e+00	9.72e+00	2.63e+01	1.83e+02	1.95e+02	2.02e+02	6.16e+02	6.64e+02	8.56e+02
	640	2.02e+01	2.06e+01	2.13e+01	8.05e+02	8.17e+02	8.38e+02	1.47e+03	1.58e+03	1.81e+03
	1280	4.17e+01	4.28e+01	4.48e+01	3.26e+03	3.37e+03	3.54e+03	-	-	-
	2560	8.92e+01	9.06e+01	9.25e+01	-	-	-	-	-	-
	5120	1.80e+02	1.83e+02	1.86e+02	-	-	-	-	-	-
	10240	3.68e+02	3.76e+02	3.82e+02	-	-	-	-	-	-

Table A.12: Time until the VTR of 10^{-10} for the SoREB function, for $\ell \in [20, 40, \ldots, 10240]$. Results are marked in orange for a success rate p^{suc} within the range $50\% \leq p^{\text{suc}} < 90\%$, and not displayed for $p^{\text{suc}} < 10\%$.

		GOMEA GBO			GOMEA BBO			AMaLGaM-FOS		
		5-Block			5-Block			5-Block		
		90p	med	10p	90p	med	10p	90p	med	10p
SoREB	20	4.07e+04	5.67e+04	6.99e+04	1.56e+05	2.20e+05	2.48e+05	6.98e+04	1.62e+05	4.40e+05
	40	5.96e+04	9.37e+04	1.25e+05	4.18e+05	6.66e+05	8.66e+05	3.90e+05	4.63e+05	9.85e+05
	80	1.12e+05	1.18e+05	2.66e+05	1.55e+06	1.64e+06	3.83e+06	1.21e+06	1.74e+06	1.81e+06
	160	1.32e+05	1.36e+05	1.47e+05	3.63e+06	3.85e+06	4.12e+06	2.86e+06	4.43e+06	8.78e+06
	320	1.51e+05	1.55e+05	4.14e+05	8.11e+06	8.45e+06	8.85e+06	1.39e+07	1.46e+07	1.90e+07
	640	1.64e+05	1.67e+05	1.68e+05	1.77e+07	1.82e+07	1.84e+07	1.66e+07	1.70e+07	1.85e+07
	1280	1.69e+05	1.70e+05	1.72e+05	3.69e+07	3.73e+07	3.75e+07	-	-	-
	2560	1.72e+05	1.74e+05	1.76e+05	-	-	-	-	-	-
	5120	1.75e+05	1.76e+05	1.79e+05	-	-	-	-	-	-
	10240	1.78e+05	1.80e+05	1.84e+05	-	-	-	-	-	-

Table A.13: Number of evaluations until the VTR of 10^{-10} for the SoREB function, for $\ell \in [20, 40, \ldots, 10240]$. Results are marked in orange for a success rate p^{suc} within the range $50\% \leq p^{\text{suc}} < 90\%$, and not displayed for $p^{\text{suc}} < 10\%$.

B

Tables of Multi-Objective Benchmarking Experiments

		GOMEA BBO			MAMaLGaM-FOS		
		Univariate			Univariate		
		90p	med	10p	90p	med	10p
ZDT1	20	9.81e-02	1.14e-01	1.98e-01	3.68e+00	5.13e+00	6.93e+00
	40	1.48e-01	1.80e-01	2.26e-01	1.13e+01	1.55e+01	2.11e+01
	80	2.89e-01	3.38e-01	4.05e-01	3.81e+01	5.27e+01	3.62e+02
	160	7.62e-01	9.94e-01	1.47e+00	1.36e+03	2.84e+03	3.48e+03
	320	2.33e+00	2.95e+00	3.87e+00	-	-	-
	640	8.06e+00	1.06e+01	1.33e+01	-	-	-
	1280	2.73e+01	3.32e+01	3.94e+01	-	-	-
	2560	1.10e+02	1.20e+02	1.46e+02	-	-	-
	5120	4.75e+02	5.53e+02	6.78e+02	-	-	-
	10240	2.40e+03	3.26e+03	3.69e+03	-	-	-

Table B.1: Time until the VTR of 0.01 for the ZDT1 function, for $\ell \in [20, 40, \ldots, 10240]$. Results are marked in orange for a success rate p^{succ} within the range $50\% \leq p^{\text{suc}} < 90\%$, marked in red for a success rate within the range $10\% \leq p^{\text{suc}} < 50\%$, and not displayed for $p^{\text{suc}} < 10\%$.

		GOMEA BBO			MAMaLGaM-FOS		
		Univariate			Univariate		
		90p	med	10p	90p	med	10p
ZDT1	20	2.033e+04	2.203e+04	2.684e+04	9.779e+03	1.147e+04	1.261e+04
	40	4.413e+04	4.828e+04	5.642e+04	1.777e+04	2.040e+04	2.468e+04
	80	9.475e+04	1.024e+05	1.103e+05	3.667e+04	4.311e+04	1.310e+05
	160	2.012e+05	2.166e+05	2.351e+05	4.407e+04	1.575e+05	4.760e+05
	320	4.128e+05	4.528e+05	4.982e+05	-	-	-
	640	8.992e+05	9.384e+05	9.908e+05	-	-	-
	1280	1.912e+06	2.150e+06	2.247e+06	-	-	-
	2560	4.751e+06	4.936e+06	5.283e+06	-	-	-
	5120	1.319e+07	1.430e+07	1.538e+07	-	-	-
	10240	3.130e+07	3.711e+07	4.894e+07	-	-	-

Table B.2: Number of evaluations until the VTR of 0.01 for the ZDT1 function, for $\ell \in [20, 40, \ldots, 10240]$. Results are marked in orange for a success rate p^{succ} within the range $50\% \leq p^{\text{suc}} < 90\%$, marked in red for a success rate within the range $10\% \leq p^{\text{suc}} < 50\%$, and not displayed for $p^{\text{suc}} < 10\%$.

		GOMEA BBO			MAMaLGaM-FOS		
		Univariate			Univariate		
		90p	med	10p	90p	med	10p
ZDT2	20	1.05e-01	1.30e-01	2.04e-01	6.54e+00	9.83e+00	1.57e+01
	40	1.66e-01	2.25e-01	3.18e-01	1.45e+01	1.79e+01	2.65e+01
	80	3.53e-01	4.50e-01	5.83e-01	5.09e+01	7.84e+01	3.02e+02
	160	7.17e-01	1.11e+00	1.47e+00	1.08e+03	3.16e+03	3.73e+03
	320	2.10e+00	2.85e+00	4.41e+00	-	-	-
	640	8.93e+00	1.12e+01	1.41e+01	-	-	-
	1280	2.95e+01	3.42e+01	4.11e+01	-	-	-
	2560	1.05e+02	1.19e+02	1.40e+02	-	-	-
	5120	4.55e+02	5.13e+02	5.47e+02	-	-	-
	10240	1.97e+03	2.37e+03	2.97e+03	-	-	-

Table B.3: Time until the VTR of 0.01 for the ZDT2 function, for $\ell \in [20, 40, \ldots, 10240]$. Results are marked in yellow for a success rate p^{succ} within the range $90\% \leq p^{\text{suc}} < 100\%$, marked in red for a success rate within the range $10\% \leq p^{\text{suc}} < 50\%$, and not displayed for $p^{\text{suc}} < 10\%$.

		GOMEA BBO			MAMaLGaM-FOS		
		Univariate			Univariate		
		90p	med	10p	90p	med	10p
ZDT2	20	2.372e+04	2.659e+04	3.553e+04	1.223e+04	1.516e+04	1.779e+04
	40	5.510e+04	6.432e+04	7.556e+04	2.100e+04	2.410e+04	2.968e+04
	80	1.192e+05	1.393e+05	1.563e+05	4.683e+03	5.516e+03	7.904e+03
	160	2.462e+05	3.043e+05	3.280e+05	3.935e+04	1.720e+05	4.930e+05
	320	5.645e+05	6.066e+05	6.701e+05	-	-	-
	640	1.194e+06	1.277e+06	1.413e+06	-	-	-
	1280	2.511e+06	2.703e+06	2.832e+06	-	-	-
	2560	5.436e+06	6.076e+06	6.558e+06	-	-	-
	5120	1.420e+07	1.489e+07	1.622e+07	-	-	-
	10240	3.843e+07	4.243e+07	5.031e+07	-	-	-

Table B.4: Number of evaluations until the VTR of 0.01 for the ZDT2 function, for $\ell \in [20, 40, \ldots, 10240]$. Results are marked in yellow for a success rate p^{succ} within the range $90\% \leq p^{\text{suc}} < 100\%$, marked in red for a success rate within the range $10\% \leq p^{\text{suc}} < 50\%$, and not displayed for $p^{\text{suc}} < 10\%$.

		GOMEA BBO			MAMaLGaM-FOS		
		Univariate			Univariate		
		90p	med	10p	90p	med	10p
ZDT3	20	1.16e-01	1.53e-01	2.26e-01	7.84e+00	1.06e+01	2.02e+01
	40	1.89e-01	2.40e-01	3.06e-01	1.75e+01	4.68e+01	9.97e+01
	80	3.22e-01	4.64e-01	8.44e-01	8.59e+01	3.47e+02	7.29e+02
	160	7.37e-01	1.19e+00	1.74e+00	-	-	-
	320	2.17e+00	3.66e+00	4.97e+00	-	-	-
	640	6.47e+00	9.25e+00	1.60e+01	-	-	-
	1280	2.58e+01	3.17e+01	4.55e+01	-	-	-
	2560	8.24e+01	9.84e+01	1.33e+02	-	-	-
	5120	3.44e+02	3.85e+02	4.54e+02	-	-	-
	10240	1.56e+03	1.74e+03	2.18e+03	-	-	-

Table B.5: Time until the VTR of 0.01 for the ZDT3 function, for $\ell \in [20, 40, \ldots, 10240]$. Results are not displayed for a success rate p^{suc} of $p^{\text{suc}} < 10\%$.

		GOMEA BBO			MAMaLGaM-FOS		
		Univariate			Univariate		
		90p	med	10p	90p	med	10p
ZDT3	20	2.395e+04	2.775e+04	3.205e+04	1.906e+04	2.169e+04	2.932e+04
	40	5.173e+04	5.703e+04	7.084e+04	3.017e+04	4.493e+04	6.708e+04
	80	1.091e+05	1.224e+05	1.408e+05	6.180e+04	1.430e+05	2.990e+05
	160	2.204e+05	2.492e+05	3.062e+05	-	-	-
	320	4.704e+05	5.609e+05	6.437e+05	-	-	-
	640	9.561e+05	1.116e+06	1.197e+06	-	-	-
	1280	1.956e+06	2.312e+06	2.657e+06	-	-	-
	2560	4.275e+06	4.884e+06	5.483e+06	-	-	-
	5120	1.044e+07	1.131e+07	1.273e+07	-	-	-
	10240	2.948e+07	3.222e+07	3.976e+07	-	-	-

Table B.6: Number of evaluations until the VTR of 0.01 for the ZDT3 function, for $\ell \in [20, 40, \ldots, 10240]$. Results are not displayed for a success rate p^{suc} of $p^{\text{suc}} < 10\%$.

		GOMEA BBO			MAMaLGaM-FOS		
		Univariate			Univariate		
		90p	med	10p	90p	med	10p
ZDT6	20	4.85e-01	6.76e-01	7.68e-01	1.75e+01	3.99e+01	1.40e+02
	40	1.12e+00	1.26e+00	1.38e+00	9.65e+01	2.12e+02	5.15e+02
	80	2.09e+00	2.46e+00	4.06e+00	3.75e+02	2.14e+03	2.76e+03
	160	5.11e+00	7.10e+00	1.18e+01	-	-	-
	320	1.63e+01	3.00e+01	4.94e+01	-	-	-
	640	5.13e+01	9.57e+01	1.50e+02	-	-	-
	1280	2.52e+02	4.83e+02	1.82e+03	-	-	-
	2560	1.67e+03	2.71e+03	3.51e+03	-	-	-
	5120	-	-	-	-	-	-

Table B.7: Time until the VTR of 0.01 for the ZDT6 function, for $\ell \in [20, 40, \ldots, 5120]$. Results are marked in yellow for a success rate p^{succ} within the range $90\% \le p^{\text{suc}} < 100\%$, marked in orange for a success rate within the range $50\% \le p^{\text{suc}} < 90\%$, marked in red for a success rate within the range $10\% \le p^{\text{suc}} < 50\%$, and not displayed for $p^{\text{suc}} < 10\%$.

		GOMEA BBO			MAMaLGaM-FOS		
		Univariate			Univariate		
		90p	med	10p	90p	med	10p
ZDT6	20	4.964e+04	7.040e+04	8.494e+04	4.653e+04	6.174e+04	1.090e+05
	40	1.129e+05	1.716e+05	1.785e+05	1.023e+05	1.596e+05	2.860e+05
	80	2.667e+05	3.639e+05	5.407e+05	1.310e+05	1.680e+05	3.290e+05
	160	7.284e+05	8.364e+05	1.297e+06	-	-	-
	320	3.606e+05	4.146e+05	5.116e+05	-	-	-
	640	2.656e+06	4.524e+06	7.008e+06	-	-	-
	1280	3.648e+05	3.763e+05	6.135e+05	-	-	-
	2560	1.345e+07	2.636e+07	6.523e+07	-	-	-
	5120	-	-	-	-	-	-

Table B.8: Number of evaluations until the VTR of 0.01 for the ZDT6 function, for $\ell \in [20, 40, \ldots, 5120]$. Results are marked in yellow for a success rate p^{succ} within the range $90\% \le p^{\text{suc}} < 100\%$, marked in orange for a success rate within the range $50\% \le p^{\text{suc}} < 90\%$, marked in red for a success rate within the range $10\% \le p^{\text{suc}} < 50\%$, and not displayed for $p^{\text{suc}} < 10\%$.

		GOMEA BBO			MAMaLGaM-FOS		
		5-Block			5-Block		
		90p	med	10p	90p	med	10p
SoREB	20	1.17	1.17	1.17	1.17	1.17	1.17
	40	1.17	1.17	1.17	1.17	1.17	1.17
	80	1.17	1.17	1.17	1.16	1.16	1.15
	160	1.17	1.17	1.17	1.14	1.14	1.13
	320	1.17	1.17	1.17	1.12	1.11	1.11
	640	1.17	1.17	1.17	1.10	1.09	1.09
	1280	1.17	1.17	1.17	1.06	1.05	1.04
	2560	1.16	1.16	1.16	1.02	1.00	0.98
	5120	1.15	1.15	1.14	0.85	0.84	0.80
	10240	1.10	1.08	1.05	0.51	0.49	0.46

Table B.9: Hypervolume reached after $\min(5\ell, 3600)$ seconds for the Sphere-SoREB function, for $\ell \in [20, 40, \ldots, 10240]$.

C
SPIE medical imaging submission

A novel model-based evolutionary algorithm for multi-objective deformable image registration with content mismatch and large deformations: benchmarking efficiency and quality

P.A. Bouter, T. Alderliesten, P.A.N. Bosman

Abstract

Taking a multi-objective optimization approach to deformable image registration has recently gained attention, because such an approach removes the requirement of manually tuning the weights of all the involved objectives. Especially for problems that require large complex deformations, this is a non-trivial task. From the resulting Pareto set of solutions one can then much more insightfully select a registration outcome that is most suitable for the problem at hand. To serve as an internal optimization engine, currently used multi-objective algorithms are competent, but rather inefficient. In this paper we largely improve upon this by introducing a multi-objective real-valued adaptation of the recently introduced Gene-pool Optimal Mixing Evolutionary Algorithm (GOMEA) for discrete optimization. In this work, GOMEA is tailored specifically to the problem of deformable image registration to obtain substantially improved efficiency. This improvement is achieved by exploiting a key strength of GOMEA: iteratively improving small parts of solutions, exploiting faster, partial evaluations of the impact of such updates on the objectives at hand. We performed experiments on three registration problems. In particular, an artificial problem containing a disappearing structure, a pair of pre- and post-operative breast CT scans and a pair of breast MRI scans acquired in prone and supine position were considered. Results show that compared to the previously used evolutionary algorithm, GOMEA obtains a speed-up of up to a factor of ~1600 on the tested registration problems while achieving registration outcomes of similar quality.

INTRODUCTION

Many state-of-the-art algorithms designed for the deformable image registration (DIR) problem are very efficient and capable of producing good results for certain applications, but only produce a single registration outcome. The reason for this is that although DIR encompasses multiple objectives, e.g. similarity and deformation smoothness objectives, these objectives are often condensed into a single optimization function through a weighted sum of the objectives. Setting these weights appropriately can be very problem-specific. Moreover, manually fine-tuning them is difficult and time-consuming. Using a multi-objective approach removes the requirement of setting weights manually, because such an approach results in a set of non-dominating solutions, i.e., a set where no single solution is better in every objective than any other solution in this set. The solution that is deemed the most appropriate for the problem at hand can manually be selected from this so-called Pareto set a posteriori. This set of solutions is highly intuitive to navigate, contrary to tuning weights a priori as is current practice. Previously, a multi-objective evolutionary algorithm (EA) was used within a recently proposed multi-objective framework for DIR, because EAs are among the state-of-the-art for multi-objective optimization. Although capable of obtaining high-quality results, previously employed multi-objective EAs have been used without many problem-specific enhancements, making them very slow, potentially beyond practical use.

In the aforementioned recent framework for multi-objective DIR, a dual-dynamic grid transformation model was proposed in order to deal with large (dis)appearing structures. Instead of overlaying the source image with a fixed grid and the target image with a moving grid, this approach overlays both the source and the target image with a moving triangulated grid of the same topology. Increasing the size of a triangle while decreasing the size of the corresponding triangle in the opposing grid naturally supports large deformations, including (dis)appearing structures. Using this model does however double the number of variables to optimize (i.e., for both grids: the number of grid points times the spatial dimensionality of the image). This large number of variables impacts the efficiency of optimization, especially for the multi-objective EA that has been used in the aforementioned recent multi-objective framework for DIR: iMAMaLGaM [1]. This algorithm samples new solutions from a normal mixture distribution probability distribution that is estimated from a selection of the best solutions from the previous generation. The inefficiency is largely caused by the fact that, to ensure the most robust performance on a large variety of problems without making any assumptions on these problems, dependencies are assumed to exist between all pairs of variables. It was previously indeed demonstrated that disregarding all dependencies in DIR leads to inferior results. However, taking into account *all* dependencies is only required for very particular, often artificial, problems. In DIR, however, grid points are strongly dependent on grid points that are adjacent, because these points directly influence the mapping of pixels from the source image to the target image, but are only weakly dependent on grid points that are remote.

To improve on the results of iMAMaLGaM, we exploit this dependency structure. To do so, we introduce a real-valued adaptation of the recently introduced Gene-pool Optimal Mixing Evolutionary Algorithm (GOMEA) that uses a prescribed dependency model to incrementally improve parts of solutions. Excellent results have recently been obtained with GOMEA on a wide range of discrete optimization problems.

Purpose

Recent research on DIR has shown that multi-objective EAs are capable of obtaining excellent results comparable to algorithms that are currently used in practice. The advantage of a multi-objective approach is that it results in a set of solutions representing different trade-offs between the objectives of interest, from which a desirable registration outcome can be selected much more insightfully compared to manually tuning weights of a traditional, weighted-objectives, single-objective approach. For DIR, the computation time required by an off-the-shelf state-of-the-art multi-objective EA is however far from desirable, potentially even prohibiting practical use. Therefore, we have developed a real-valued problem-specific adaptation of the recently introduced Gene-pool Optimal Mixing Evolutionary Algorithm to greatly improve the efficiency of multi-objective DIR.

MATERIALS AND METHODS

Real-valued GOMEA

A key difference between GOMEA and iMAMaLGaM is that GOMEA has a much higher selection pressure as a result of incrementally improving parts of existing so-called parent solutions. The parts to improve are identified in a so-called dependency, or linkage, model. Instead of estimating one normal mixture probability distribution, as is done in iMAMaLGaM, GOMEA estimates such a distribution for each subset of variables in the dependency model separately. Solutions are then improved by sampling new values for these subsets of variables in the linkage model independently. Only if these newly sampled values lead to an improved solution, the newly sampled values are accepted, otherwise the parent solution is returned to its previous state. Because solutions are now only partially altered, it is not required to re-evaluate the entire solution to assess the contribution of a partial alteration. Such partial evaluations are an important reason for the practical efficiency of GOMEA.

To apply GOMEA to DIR we have studied the use of various dependency models. A good model should represent the minimum number of dependencies required to be able to solve the problem efficiently. Initial experiments have shown that a dependency structure in which each element describes all coordinates of a single triangle leads to the best results. This means that solutions are incrementally modified by sampling new coordinates for single triangles (either in the source or the target triangulation). Moreover, for DIR, efficient partial evaluations require that objective values can be updated by only re-evaluating contributions made by individual triangles (δ) that were modified. For this purpose, each objective is defined as a sum over the set of triangles Δ_s in the source grid and the triangles Δ_t in the target grid.

Similarity objective

In this paper, the similarity of the transformed source image and the target image is evaluated by computing the mean squared difference in pixel intensities, which is to be minimized. We note that, although not considered here, other notions of (dis)similarity often used in DIR can be evaluated partially also. For a single triangle δ, we iterate over the pixels inside this triangle, denoted $px(\delta)$ and compute the squared difference between a pixel's intensity in the one image and the bilinear-interpolated intensity at this pixel's corresponding position in the other image. The correspondence of positions in opposing images is straightforwardly governed by the correspondence of triangles in the triangulations defined over the opposing images. To obtain values that are independent of the image resolution, normalization is applied by dividing by the total number of pixels. Defining the (bi-linearly interpolated) intensity of a point p as $I(p)$ and its corresponding position in the opposing grid as p^c, we have:

$$F_{dissimilarity} = \frac{1}{\sum_{\delta_s \in \Delta_s} |px(\delta_s)| + \sum_{\delta_t \in \Delta_t} |px(\delta_t)|} \left[\sum_{\delta_s \in \Delta_s} \left[\sum_{p_s \in px(\delta_s)} \left(I_s(p_s) - I_t(p_s^c) \right)^2 \right] + \sum_{\delta_t \in \Delta_t} \left[\sum_{p_t \in px(\delta_t)} \left(I_t(p_t) - I_s(p_t^c) \right)^2 \right] \right]$$

Deformation magnitude objective

We employ Hooke's law based on the mean squared difference of edge lengths between edges e in the one grid and their corresponding edges e^c in the opposing grid. The final objective value of one triangle is then the squared sum of edge-length differences, normalized by dividing by three times the total number of triangles, i.e.,:

$$F_{deformation} = \frac{1}{3(|\Delta_s| + |\Delta_t|)} \left[\sum_{\delta_s \in \Delta_s} \left[\sum_{e_s \in edges(\delta_s)} (\|e_s\| - \|e_s^c\|)^2 \right] + \sum_{\delta_t \in \Delta_t} \left[\sum_{e_t \in edges(\delta_t)} (\|e_t\| - \|e_t^c\|)^2 \right] \right]$$

Guidance error objective

This objective is only used when guidance information is supplied, which is a set of tuples of contours or landmarks $G = \{(G_s, G_t)_1, \ldots, (G_s, G_t)_k\}$ that predefine corresponding points or lines in the source and target images. The guidance objective aims to minimize the distance between these pairs of contours. For all pixels on a contour within a certain triangle δ, denoted $G(\delta)$, the minimal distance to a point on the opposing contour is calculated. The objective value is the total sum of these minimal distances, normalized by the total number of pixels on the contour, again computed symmetrically, and summed over all pairs of contours, i.e.,:

$$F_{guidance} = \sum_{\delta_s \in \Delta_s} \left[\sum_{(G_s, G_t) \in G} \frac{1}{|G_s| + |G_t|} \sum_{p_s \in G_s(\delta_s)} \min_{p_t \in G_t} \{ d(p_s^c, p_t) \} \right] + \sum_{\delta_t \in \Delta_t} \left[\sum_{(G_s, G_t) \in G} \frac{1}{|G_s| + |G_t|} \sum_{p_t \in G_t(\delta_t)} \min_{p_s \in G_s} \{ d(p_t^c, p_s) \} \right]$$

Feasibility constraints

If any two edges within a triangulation intersect, this means that certain pixels are located within the perimeter of more than one triangle. A technique known as constraint domination, by which feasible solutions are always preferred over infeasible solutions, is used to deal with infeasible solutions. To ensure efficiency by exploiting partial evaluations in GOMEA, constraint-violation values also must be computable per triangle. Therefore, we check whether a grid point p is inside the polygon bounded by the edges between the neighbors of p. If p lies outside this polygon, the triangulated grid has intersecting edges and is therefore infeasible. The total constraint-violation value is the number of grid points that violate this constraint. After generating a subset of new points, each of the newly generated points and each of their neighbors have to be re-checked to update the constraint-violation value. Note that this is a linear-time calculation as opposed to the quadratic-time method previously used in iMAMaLGaM to check all pairs of edges for intersections.

Experiments

We compared the speed and quality of results of GOMEA with that of iMAMaLGaM. For a notion of Pareto front quality, we used the well-known hypervolume metric that calculates the total volume of the objective space that is dominated by a Pareto front. By considering the time it takes two algorithms to reach a certain hypervolume, we can define a notion of speed-up. We evaluate the quality of registration results based on a problem with a set of five annotated landmarks, allowing us to compute the target registration error (TRE), which is the mean distance between the deformed landmark locations and the predefined target landmark locations. The purpose of this metric is similar to that of the guidance objective, but it is not used as an objective during optimization.

In the first experiment we ran GOMEA and iMAMaLGaM on two problems with a content mismatch. In particular, we consider an artificial benchmark problem and a pair of 2D slices taken from pre- and post-operative breast CT scans, both including guidance information. Previous work demonstrated that both problems can be adequately solved by iMAMaLGaM in a multi-resolution scheme, but for the sake of comparison, in this experiment we considered the use of GOMEA and iMAMaLGaM using only single grid-resolution schemes with dimensions 6×6, 11×11, and 21×21. Each run was terminated after $\ell \cdot 10^4$ evaluations where ℓ is the number of variables, respectively being 144, 484 and 1764. For this purpose, each of GOMEA's partial evaluations was counted as a single evaluation in iMAMaLGaM.

In a second experiment we used a multi-resolution scheme in GOMEA with resolutions 6×6, 11×11, and 21×21 on two 2D slices of a breast MRI scan acquired in prone and supine position, not including guidance information. A time limit equal to the number of variables in seconds is used for each resolution, totaling close to 40 minutes.

RESULTS AND DISCUSSION

For the first experiment, the hypervolumes interpolated over 4 runs are displayed in Fig. 1 where the shaded areas demark the range between the best and worst run. The median run is displayed as the plotted line. Speed-up is then calculated by comparing the time required to reach the minimum hypervolume obtained by either iMAMaLGaM or GOMEA upon termination. For the artificial problem, speed-up factors of 114, 102, and 1641 are obtained on the respective resolutions. For the pre- and post-operative problem, A speed-up of a factor of 0.25, 31, and 940 on the respective solutions. The apparent inferior performance of GOMEA for the 6×6 grid resolution on the pre-post operative problem is caused by the fact that in GOMEA we used a slightly different definition of triangulations where the outer points of the grid (i.e., 20 out of 36 points for a 6×6 regular grid resolution) are constrained to be on the borders of the image so as to ensure that the entire image is covered by the transformation. This was not the case for the recentmost publication of iMAMaLGaM for DIR [3]. For higher grid resolutions, the impact of this difference is negligible.

Figure 2 shows the Pareto fronts after each stage of the multi-resolution scheme for GOMEA. Each solution is color-coded with regard to its mean TRE and the solution with the minimal mean TRE is encircled. After running the 6×6 grid resolution for only 144 seconds, a solution was already found with a mean TRE of 2.0 mm. For higher resolutions the mean TRE does not get much better, but the deformation does become smoother. The position of the encircled solution makes it clear that minimizing dissimilarity does not always lead to higher-quality solutions. Instead, highly deformed, over-fitted solutions with unnatural deformations are obtained. Moreover, because the mean TRE is very irregular along the Pareto front, searching by trial and error for a linear combination of weights to use in a single-objective approach that linearly weights the objectives is very likely to get stuck in a local optimum of weights with a high risk of obtaining an unsatisfactory registration outcome. This highlights the strength of a multi-objective approach.

Our results could be improved even further by enabling a previously introduced adaptive steering approach [2]. During optimization, this technique purges certain solutions from the Pareto set in order to direct the algorithm towards the most interesting part of the Pareto front, i.e., the part where each objective value of a solution is close to the best known values of this objective function. GOMEA could further be combined with the recently introduced smart grid-initialization technique that was observed to obtain an additional speed-up of a factor between 10 and 100 [3].

New or breakthrough work to be presented

We introduce a new multi-objective evolutionary algorithm for deformable image registration that obtains a speed-up of up to a factor of ~1600 compared to the previously used algorithm, scaling to even higher factors as the dimensionality of the transformation grid increases.

CONCLUSION

Multi-objective deformable image registration with a dual-dynamic transformation model to account for large anatomical differences has high potential, but using off-the-shelf state-of-the-art multi-objective EAs it is prohibitively slow for real-world practice. We showed that by using a problem-specific tailored multi-objective implementation of the recently introduced EA known as GOMEA, a large speed-up can be obtained that opens the door to applying multi-objective evolutionary algorithms to accurately solve deformable image registration problems in 3D with dual dynamic transformation grids to support large anatomical variations and content mismatches.

Whether the work is being, or has been, submitted for publication or presentation elsewhere. No.

REFERENCES

[1] S.F. Rodrigues, P. Bauer and P.A.N. Bosman. "A novel population-based multi-objective CMA-ES and the impact of different constraint handling techniques". Proc. of Annual Conf. on Genetic and Evolutionary Computation. ACM, 2014.

[2] T. Alderliesten, P.A.N. Bosman and A. Bel. "Getting the most out of additional guidance information in deformable image registration by leveraging multi-objective optimization". Proc. *SPIE Medical Imaging*, 2015.

[3] P.A.N. Bosman and T. Alderliesten. "Smart grid initialization reduces the computational complexity of multi-objective image registration based on a dual-dynamic transformation model to account for large anatomical differences". Proc. *SPIE Medical Imaging*, 2016.

SUPPORTING TABLES AND FIGURES

Figure 1: Hypervolumes of single resolution runs with 6×6, 11×11, and 21×21 grids, on an artificial problem instance with a disappearing structure (top row) and a set of pre- and post-operative breast CT scans (bottom row).

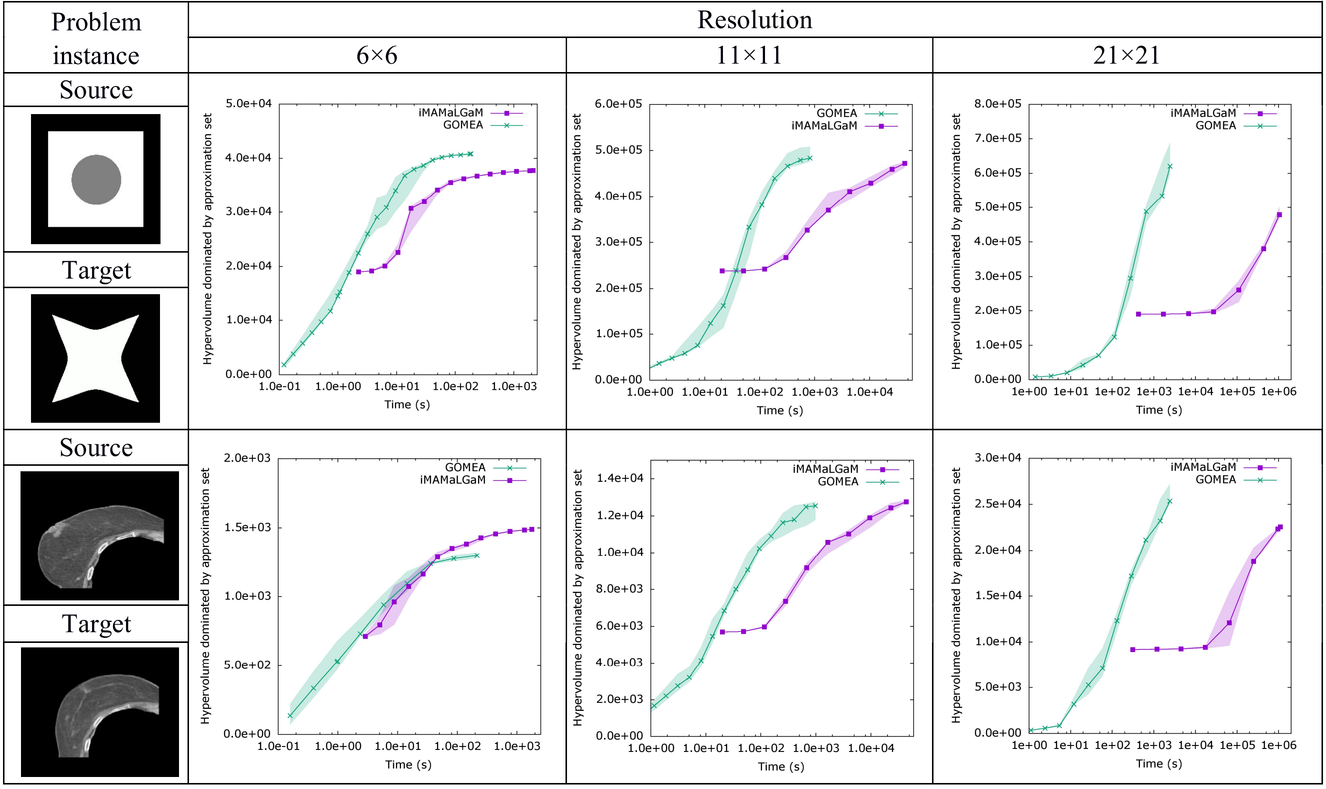
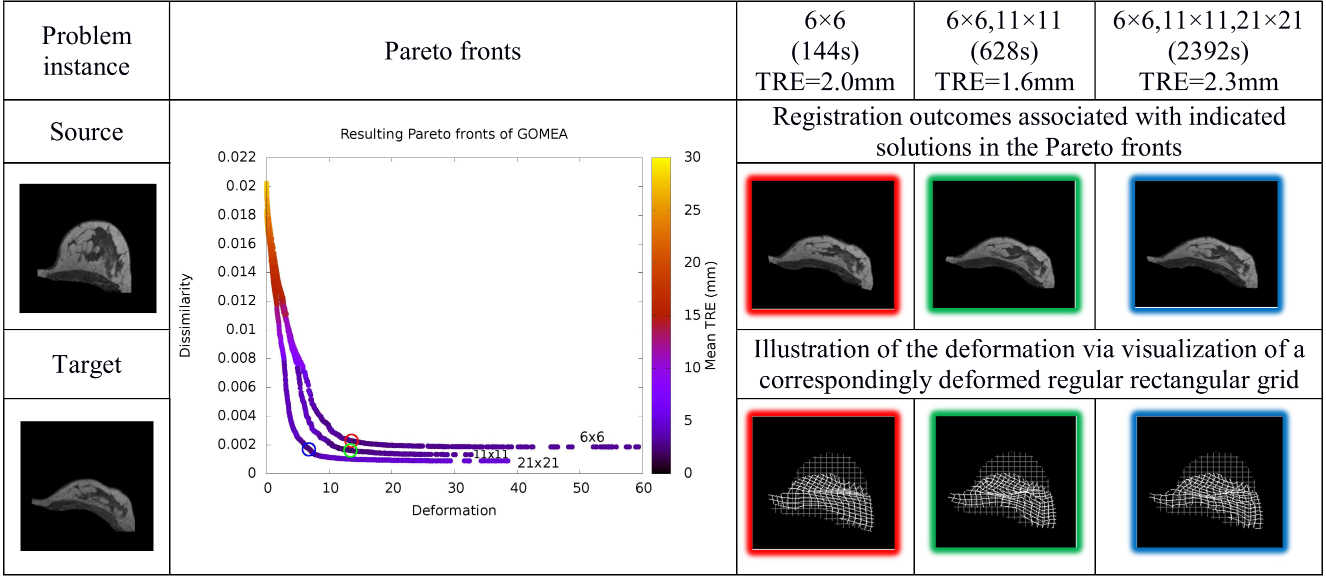


Figure 2: Results on the prone-supine breast MRI test case. The Pareto fronts are displayed for the single, double, and triple resolution runs where every run started from the 6×6 grid resolution. In each Pareto front, the solution with the minimal mean TRE is encircled and displayed on the right along with its deformation applied to a uniform grid, its mean TRE, and the run-time. Note: the actual dual-dynamic triangular grids that define the deformation field are not shown.


Problem instance
Pareto fronts
6×6 (144s) TRE=2.0mm
6×6,11×11 (628s) TRE=1.6mm
6×6,11×11,21×21 (2392s) TRE=2.3mm
Source
Target
Registration outcomes associated with indicated solutions in the Pareto fronts
Illustration of the deformation via visualization of a correspondingly deformed regular rectangular grid
Resulting Pareto fronts of GOMEA
Dissimilarity
Deformation
Mean TRE (mm)
6x6
11x11
21x21