Mixed-integer evolution strategies for parameter optimization and their applications to medical image analysis

Li, R.

Mixed-Integer Evolution Strategies for Parameter Optimization and Their Applications to Medical Image Analysis

Rui Li

Mixed-Integer Evolution Strategies for Parameter Optimization and Their Applications to Medical Image Analysis

Proefschrift

ter verkrijging van
de graad van Doctor aan de Universiteit Leiden,
op gezag van de Rector Magnificus prof. mr. P.F. van der Heijden,
volgens besluit van het College voor Promoties
te verdedigen op dinsdag 6 October 2009
klokke 13.45 uur

door

Rui Li
geboren te Urumqi, Xinjiang, China 1978

Promotiecommissie

Promotor:	Prof. Dr. T.H.W. Bäck	
Co-promotor:	Dr. M.T.M. Emmerich	
Referent:	Dr. S. Cagnoni	(University of Parma)
Overige leden:	Prof. Dr. G. Rozenberg	
	Prof. Dr. F. Arbab	
	Prof. Dr. J.N. Kok	
	Prof. Dr. H.A.G. Wijshoff	
	Prof. J.H.C. Reiber	(Leiden Universiteit Medical Center)
	Dr. J. Eggermont	(Leiden Universiteit Medical Center)

The work in this thesis has been carried out under the auspices of the research school IPA (Institute for Programming research and Algorithmics).

This research was financed by the Netherlands Organisation for Scientific Research (NWO) under project 612.066.408 “SAVAGE”.

Cover design: Yi Zhao.

ISBN 978-90-9024665-9

This thesis is dedicated to my wife Bin.

Contents

Chapter 1

Introduction

Searching for an optimal state is one of the most important phenomena in natural systems. For instance, atoms try to form optimal bonds thereby obtaining energy minimal states, real ants are capable of adapting to the changing environment and finding shortest path from the nest to the food source, and the aggregate motion of a flock of birds increases the success rate of their vigilance. These amazing solutions from nature have always been a source of inspiration for scientists and engineers to tackle various challenging applications in our world. Natural computing is a field of research that works with computational techniques inspired by nature and develops algorithms for solving complex real-world problems [68]. In general, natural computing consists of mainly three branches, in which each has its own representative techniques (Table 1.1): Among these aforementioned

Natural Computing Branches	**Representative Techniques**
Computing inspired by natural systems	evolutionary computation, neural networks, swarm intelligence, etc.
Simulation and emulation of nature	lindenmayer systems and artificial life.
Computing with natural materials	DNA computing and quantum computing.

Table 1.1: Different natural computing branches and its typical techniques.

techniques, we focus our attention on evolutionary computation, which nowadays is one of the most active research fields of computer science with a huge amount of successful applications to real-world problems and for some techniques a highly developed theoretical foundation. Rather than emulating features of a single biological organism, evolutionary computation draws its inspiration from the dynamics of an entire population of organisms. It uses the concepts of *mutation*, *recombination*, and *selection* to mimic the process of *"organic evolution"*, in which *survival of the fittest* and *phenotypic variation* [28] principles play an important role and lead to better adaptation of a population of individuals to a given

evolutionary environment, that is, individuals with the higher fitness[1] have better chances of survival and multiplying. The whole collection of algorithms, which are derived from this "*organic evolution*" process, are normally termed evolutionary algorithms (EAs) in literature.

The original idea of our work is to extend the canonical Evolution Strategies (ES) - which is one of three computation paradigms[2] of EAs - from traditional real-valued parameter optimization domain to mixed-integer parameter optimization domain. This is necessary because there exist numerous practical optimization problems from industry in which the set of decision variables simultaneously involves continuous, integer and discrete variables. Furthermore, objective functions of this type of problems could be based on large-scale simulation models or the structure of the objective functions may be too complex to be modeled. From this perspective, optimization problems of this kind are classified into the *black-box* optimization category. For them, classic optimization techniques, which come from Mathematical Programming (MP) research field, can not be easily applied, since they are based on the assumption that the search space can always be efficiently explored using a *divide-and-conquer* scheme. While our new proposed algorithm, the so-called Mixed-Integer Evolution Strategies (MIES), by contrast, is capable of yielding good solutions to these challenging *black-box* optimization problems by using specialized variation operators tailored for mixed-integer parameter classes.

In this work not only did we introduce MIES and study it intensively from a theoretical point of view, but also we develop the framework for applying MIES to the real-world optimization problem in the medical field. More specifically, we apply MIES to the optimization of control parameters of a semi-automatic image analysis system for Intravascular Ultrasound (IVUS) images, real-time high resolution tomographic images which show the inside of coronary or other arteries. IVUS images are difficult to interpret which causes manual segmentation to be highly sensitive to intra- and inter-observer variability [66]. Thus, the development of feature detection systems for IVUS images has received much attention in medical and computer science research. However, the performance of most systems depend on a large number of control parameters that are hard to optimize manually and may differ for different interpretation contexts. Moreover, these parameters are subject to change when something changes in the image acquisition process. Compared to other approaches, with MIES the system developer can search for optimized parameter settings automatically and likely will obtain parameter settings that lead to significant higher accuracy of the feature detectors.

1.1 Overview of Thesis

The contents of this dissertation consist of three major parts: (1) the introduction and theoretical study of the newly proposed optimization algorithm; (2) Its ap-

[1] It is determined by the given environment .

[2] Another two computation paradigms are Evolutionary Programming (EP) and Genetic Algorithms (GAs).

plication to the real-world application, that is, parameter optimization of medical image analysis; (3) advanced topics, such as *Niching techniques*. More specifically, in the theoretical part the state-of-the-art MIES algorithms are introduced, and then they are tested on several carefully designed artificial landscapes, for instance, generalized Nk landscapes. The real-world application part mainly focuses on parameter optimization problems from medical research field. Our proposed MIES algorithms are applied to optimize a multi-agent system, which was developed for medical image feature detection. And some important experimental observations will be presented. In the third part, some advanced techniques, which can be used in combination with MIES, are investigated to further improve the performance of our algorithms, for example, *Metamodel-Assisted Optimization*, *Niching Techniques* and *Bayesian Learning*.

The more detailed structure of this thesis can be summarized as follows:

Chapter 2 first provides a brief overview of the essential terminology of global optimization, and the mixed-integer parameter optimization problem is introduced specifically. Several classic algorithms from the traditional Mathematical Programming (MP) research field, such as Branch-and-Bound (BB) and Outer Approximation (OA) methods, are reviewed after. As opposed to this *white-box* optimization methodology, the framework for mixed-integer parameter optimization in the *black-box* scenario is discussed in very detail. Two representative real-world applications - *optical filter design* and *chemical plant optimization* - are also presented as motivating examples.

In Chapter 3 we first introduce the general framework of EAs. Next, we explain the fundamentals of the canonical ES explicitly, which serves as the algorithmic kernels of our proposed methodology - MIES for mixed-integer parameter optimization. Then the design philosophy of MIES and several important properties are discussed in detail.

In Chapter 4, we propose two innovative synthetic test problems - *Barrier Functions* and *Mixed-Integer NK landscapes (MINKL)*. Barrier functions are created by a multi-modal problem generator that produces integer optimization problems with a scalable degree of ruggedness but no interaction between variables. MINKL are an extension of standard NK Landscapes (NKL), which are stochastically generated pseudo-boolean functions with N bits (genes) and K interactions between genes. These two artificial test problems are carefully designed and experimental results show that they are particular useful to understand the dynamics of evolutionary search within the mixed-integer space.

MIES for parameter optimization of IVUS image analysis are presented in chapter 5. An advanced multi-agent system for IVUS image features detection, especially for lumen feature detection, is introduced and the framework for optimizing this system using MIES is proposed as well as some promising experimental results.

In Chapter 6 we investigate the use of fitness based partitioning to find groups of Computed Tomographic Angiography (CTA) images that require a similar parameter setting for the segmentation algorithm while at the same time evolving

optimal parameter settings for these groups.

Chapter 7 discusses how to use metamodels, in particular radial basis function networks (RBFN), to assist MIES when applied to optimization tasks with time consuming evaluation functions, like IVUS image analysis.

Chapter 8 presents a dynamic niching technique for MIES, based upon on an existing ES niching approach, which was developed recently and successfully applied to continuous landscapes. The new method is based on the heterogeneous distance measure that addresses search space similarity in a way consistent with the mutation operators of the MIES.

Chapter 9 introduces a new estimation of distribution algorithm that extends the Bayesian optimization algorithm (with fixed network structure) from binary optimization problems to mixed-integer optimization problems. Experimental results show that a-priori knowledge on dependencies between decision variables can be exploited by this proposed algorithm in order to improve convergence speed and reliability. In this algorithm, MIES serves as a sub-algorithm in the self organized clustering process.

1.2 Overview of Publications

Here we give an overview of the way in which parts of this thesis have been published.

Chapter 3: Mixed Integer Evolution Strategies

The content of this chapter is partly based on research, which was accepted for publication as a chapter contribution in a book on Evolutionary Image Analysis and Signal Processing of Springer "Studies in Computational Intelligence" series [79].

Chapter 4: Synthetic Mixed-Integer Landscapes

A major portion of this chapter is published in the Proceedings of the Ninth International Conference on Parallel Problem Solving from Nature (PPSN IX, 2006) [78] and an extended abstract in the Proceedings of the 18th Belgium-Netherlands Conference on Artificial Intelligence (BNAIC'06) [32].

Chapter 5: Parameter Optimization for Medical Image Analysis

Major parts of this chapter are published in the Proceedings of the 1st International Workshop on Computer Vision for Intravascular and Intracardiac Imaging (MICCAI 2006) [18], Proceedings of Genetic and Evolutionary Computation Conference (GECCO'06) [77], Proceedings of Sixth European Workshop on Evolutionary Computation in Image Analysis and Signal Processing (EvoIASP'06) [75], Proceedings of Adaptive Computing in Design and Manufacture (ACDM'06) [76]

and an extended abstract in the Proceedings of the 18th Belgium-Netherlands Conference on Artificial Intelligence (BNAIC'06) [39].

Chapter 6: Dynamic Fitness Based Partitioning

This chapter is published in the Proceedings of the Seventh European Workshop on Evolutionary Computation in Image Analysis and Signal Processing (EvoIASP'07) [72], an extended abstract in the Proceedings of the 19th Belgium-Netherlands Conference on Artificial Intelligence (BNAIC'07) [73] and Proceedings of Eighth European Workshop on Evolutionary Computation in Image Analysis and Signal Processing (EvoIASP'08) [33].

Chapter 7: Meta-Model Assisted Mixed Integer Evolution Strategies

The research results in this chapter are published in the Proceedings of the IEEE Congress on Evolutionary Computation (IEEE CEC 2008) [80].

Chapter 8: Mixed-Integer Evolution Strategies with Dynamic Niching

This chapter is based on publication in the Proceedings of 10th International Conference Parallel Problem Solving from Nature (PPSN X, 2008) [74].

Chapter 9: Mixed-Integer Evolution Strategies with Bayesian Learning

Parts of this chapter are published as a full paper contribution in the Proceedings of the 20th Belgium-Netherlands Conference on Artificial Intelligence (BNAIC'08) [40].

Part I

Mixed-Integer Evolution Strategies

The expectations of life depend upon diligence; the mechanic that would perfect his work must first sharpen his tools.

Confucius

Chapter 2

Mixed-Integer Parameter Optimization

To start our journey, good preparation is always required. This chapter lays the groundwork for our study in this dissertation. Like we declared in chapter 1, the original goal of our work is to develop efficient and robust methods to deal with mixed-integer parameter optimization problems particularly in *black-box* optimization scenario. Therefore, it is important for us to first introduce the elementary terminology of the global optimization, especially the mixed-integer parameter optimization. As a traditional approach of formulating optimization problems, Mathematical Programming (MP) as well as its major subfields, such as Linear Programming (LP) and Mixed-Integer Programming (MIP), will also be covered explicitly. Next, two well-established techniques - Branch-and-Bound (BB) and Outer Approximation (OA) - will be reviewed thoroughly, because they are widely used for solving Mixed-Integer Nonlinear Programming (MINLP) problems in practice. As opposed to these *white-box* based optimization problems, we will address *black-box* optimization in mixed-integer parameter search space. At last, two selected optimization applications from industrial field will be presented.

2.1 Global Optimization

The global optimization problem can be generalized in terms of finding the combination of parameters which optimize a given quantity depending on these parameters, possibly subject to some restrictions on the allowed parameter ranges. The quantity to be optimized is called the *objective function*[1]; the parameters which may be changed in the quest for the optimum are called *control* or *decision variables*; the restrictions on allowed parameters values are known as *constraints*.

[1] Also called performance measure, loss function, or fitness function in some context.

It is customary to write the global optimization problem as follows:

$$\begin{array}{rrcll} \text{minimize} & f(\mathbf{x}) & \in & \mathbb{R} & \\ \text{subject to} & g_i(\mathbf{x}) \in \mathbb{R} & \leq & 0 & i \in I_g \\ & h_j(\mathbf{x}) \in \mathbb{R} & = & 0 & j \in I_h \\ & \mathbf{x} & \in & \Theta, \Theta \neq \varnothing & \end{array} \tag{2.1}$$

where $f(\mathbf{x}) : \mathbf{x} \in \mathbb{R}^n$ is the objective function and its value is called the objective value of this function. $g_i(\mathbf{x})$ and $h_j(\mathbf{x})$ are the set of constraint functions. Constraint equations of the form $g(\mathbf{x}) \leq 0$ denote *inequality* constraints, and those of the form $h(\mathbf{x}) = 0$ denote *equality* constraints. I_g represents the inequalities index set, and I_h indicates the index set of equalities. Θ represents non-empty set of allowable values for $\mathbf{x}$ and is defined as:

$$\Theta = \{\mathbf{x} \in \mathbb{R}^n \,|\, g_i(\mathbf{x}) \leq 0 \ \wedge \ h_j(\mathbf{x}) = 0\} \tag{2.2}$$

Consequently, the optimal solution set can be described as follows [114]:

$$\Theta^* \equiv \arg \min_{\mathbf{x} \in \Theta} f(\mathbf{x}) = \{\mathbf{x}^* \in \Theta : f(\mathbf{x}^*) \leq f(\mathbf{x}) \text{ for all } \mathbf{x} \in \Theta\} \tag{2.3}$$

where "arg $\min_{\mathbf{x} \in \Theta}$" can be read as: Θ^* is the set of values $\mathbf{x} = \mathbf{x}^*$ that minimize (maximize in the case of "$\geq$") $f(\mathbf{x})$ subject to $\mathbf{x}^*$ satisfying the constraints represented in functions $g_i(\mathbf{x})$ and $h_j(\mathbf{x})$. In general, when the Θ of the problem is not *convex*[2], there may be several local minima and maxima, where a local minimum $\hat{\mathbf{x}}$ is defined as a point for which there exists some $\delta > 0$ so that for all $\mathbf{x}$ such that $\|\mathbf{x} - \hat{\mathbf{x}}\| \leq \delta$ the expression $f(\hat{\mathbf{x}}) \leq f(\mathbf{x})$ holds.

2.2 Mathematical Programming

Traditionally, to apply optimization concepts and tools optimization problems are frequently modeled by using Mathematical Programming (MP). MP is concerned mainly with optimization problems whose objective(s) and constraints can be clearly described by using algebraic mathematical expressions. MP is the branch of applied mathematics and numerical analysis that focuses on reaching "*best*" solutions (or decisions) by means of mathematical optimization models. It is a subbranch of operations research[3] (OR) and there exists a rich body of knowledge surrounding these optimization techniques. Many different subfields can be defined based on what kind of mathematical model is to be used to describe the optimization problem at hand [123]. In the following parts, we will review major subfields of MP, especially Mixed-Integer Nonlinear Programming (MINLP) problems and corresponding techniques.

[2] Even if Θ is convex there may be multiple local optima, as a result of non-convexity of $f(x)$.
[3] Also known as management science (MS).

2.2.1 Linear vs. Nonlinear

A linear programming (LP) problem is an optimization problem which satisfies the following requirements: (1) Objective function f is a *linear* function; (2) Both *inequality* constraint functions $g_i(i \in I_g)$ and *equality* constraint functions $h_j(j \in I_h)$ are *linear* functions. In LP problems, the linear constraints result in a convex feasible solution space. Some algorithms are developed based on this characteristic, for example the *Simplex* algorithm [27], which is very efficient in practice: its worst-case complexity is exponential in the number of problem variables.

As opposed to LP problems, there are also a large number of optimization problems in which their objective and constraints functions are nonlinear in decision variables $\mathbf{x}$. Problems in this category are called Nonlinear Programming (NLP) problems. Because of the nonlinearity of constraint functions or the objective function, the convexity of the solution space can not be guaranteed anymore. As a consequence of nonconvexity, NLP problems may have many different local optima compared to LP problems, and choosing the best one is an extremely hard task. Several nonlinear programming algorithms have been developed to obtain the convex solution space by linearising the constraints firstly, and, as a second step, employ some LP methods to find an optimal feasible solution.

2.2.2 Integer vs. Mixed-Integer

Divisibility is a common assumption in many optimization methods. It requires that each decision variable x_i is allowed to assume fractional values. A LP problem in which some or all of the variables must be non-negative integers is called an Integer Programming (ILP) problem. IP problems can be further classified into pure Integer Programming and Mixed-Integer Programming (MILP). An Integer Programming (IP) problem in which all decision variables need to be integers is called a pure integer programming problem. An integer programming problem in which only some of the variables are required to be integers is called Mixed-Integer Programming (MILP) problem.

2.2.3 Mixed-Integer Nonlinear Programming

In real world, many optimization applications are not only complex and challenging because their decision variables are combinations of real and integer variables, but also their objective function and constraint functions are nonlinear. For example, problems in the optimization of process flowsheets, portfolio selection, batch processing in chemical engineering, and optimal design of gas or water transmission networks [48, 63]. The Mixed-Integer Nonlinear Programming (MINLP) is a natural approach of formulating these kind of problems where it is necessary to simultaneously optimize the system structure (discrete) and parameters (continuous) [23, 42].

MINLP problems are very hard to solve in practice, because they combine all the difficulties of their subclasses: the combinatorial nature of Mixed-Integer Linear Programming (MILP) and the difficulty of solving Nonlinear Programming (NLP). In general, these two subclasses problems can be classified into the class of NP-hard problems. Although they are very hard to solve, the component structure of MILP and NLP within MINLP provides a collection of natural algorithmic approaches, exploiting the structure of each of the subcomponents. Analogous to Equation 2.1, we now state the general MINLP problem as follows:

$$\begin{array}{rrcll} \text{minimize} & f(\mathbf{x},\mathbf{y}) & \in & \mathbb{R} & \\ \text{subject to} & g_i(\mathbf{x},\mathbf{y}) \in \mathbb{R} & \leq & 0 & i \in I_g \\ & h_j(\mathbf{x},\mathbf{y}) \in \mathbb{R} & = & 0 & j \in I_h \\ & \mathbf{lb_x} \leq & \mathbf{x} & \leq \mathbf{ub_x} & \\ & \mathbf{lb_y} \leq & \mathbf{y} & \leq \mathbf{ub_y} & \\ & & \mathbf{x} & \in \mathbb{R}^n & n \geq 0 \\ & & \mathbf{y} & \in \mathbb{Z}^m & m \geq 0 \end{array} \tag{2.4}$$

where $f : \mathbb{R}^n \times \mathbb{Z}^m \rightarrow \mathbb{R}$ is called the objective function. The members of the solution space are bounded from above and below by **ub** and **lb** respectively. $\mathbf{x}$ is a real valued vector in $\mathbb{R}^n$ and $\mathbf{y}$ is an integer (normally *binary*) valued vector in $\mathbb{Z}^m$. Please note that the objective function f and constraint functions g_i, h_i are nonlinear in this situation.

There are several techniques employed to solve MINLP problems. They differ in complexity and running time as well as solution principle and scope of application. Branch-and-Bound (BB) and Outer Approximation (OA) are two widely used methods.

Branch-and-Bound

Branch-and-Bound (BB) is an intelligently structured search for all the feasible solutions [71]. It is non-heuristic, in the sense that it maintain a provable upper and lower bound on the (globally) optimal objective value [20] and after termination will obtain the optimal solution. The space of all feasible solutions is repeatedly partitioned into smaller and smaller subsets, and a lower bound[4] is calculated within each subset. Subsets with a bound that exceeds the cost of a known feasible solution are excluded from all further steps. The partitioning procedure continues until a feasible solution is found such that its cost is no greater than the lower bound for any other subset.

Using pseudocode to explain Branch-and-Bound (BB) method, one can use the following definitions [14, 49, 81]: a list $\mathcal{L}$ of unsolved subproblems[5] $\mathcal{S}_i$, which were obtained by relaxing some or all of the integer requirements; *ub*, an upper bound

[4] in the case of minimization.

[5] Or node, denote the problem associated with a certain portion of the feasible region of MINLP

on the value of objective function f; $lb_{\mathcal{S}_i}$, a lower bound on the value that f can have in subproblem $\mathcal{S}_i$; *active set*, the list of subproblems that must still be solved; $(\mathbf{x}^*, \mathbf{y}^*)$, a record of the best integer solution (or incumbent solution) which has been found by the algorithm so far. The basic branch-and-bound method can be generalized as algorithm 1.

Algorithm 1 The Branch-and-Bound Algorithm

1: **Initialize:**
$\mathcal{L}$, $(\mathbf{x}^*, \mathbf{y}^*)$, and ub.
2: **Select:**
Choose an unsolved subproblem $\mathcal{S}_i$ from $\mathcal{L}$. Stop if $\mathcal{L} = \varnothing$. If there is an incumbent solution, then that is an optimal solution. Otherwise, the MINLP is infeasible.
3: **Solve:**
Solve the nonlinear programming relaxation of $\mathcal{S}_i$. A solution $(\hat{\mathbf{x}}, \hat{\mathbf{y}})_{\mathcal{S}_i}$ and $lb_{\mathcal{S}_i}$ are obtained on the optimal value of this subproblem.
4: **Prune:**
If the relaxed subproblem $\mathcal{S}_i$ was infeasible, then $\mathcal{S}_i$ will not provide a better solution to MINLP than the known incumbent solution. The same as $lb_{\mathcal{S}_i} \geq ub$. Delete such $\mathcal{S}_i$ from $\mathcal{L}$ and return to the **Select** step.
5: **Integer Solution:**
If $\hat{\mathbf{y}}$ is integer, then a new incumbent integer solution has been obtained. $(\mathbf{x}^*, \mathbf{y}^*) = (\hat{\mathbf{x}}, \hat{\mathbf{y}})$ and ub is set to the optimal value of $\mathcal{S}_i$
6: **Branch:**
If there exist at least one y_k is fractional value in the solution on the $\mathcal{S}_i$, then consider splitting the $\mathcal{S}_i$. Create a new subproblem $\mathcal{S}_{i_1}$ by adding the constraint $y_k \leq \lfloor \hat{y_k} \rfloor$. Create another subproblem $\mathcal{S}_{i_2}$ by adding the constraint $y_k \geq \lceil \hat{y_k} \rceil$.Remove $\mathcal{S}_i$ and add problems $\mathcal{S}_{i_1}$ and $\mathcal{S}_{i_2}$to $\mathcal{L}$. Return to **Select** step.

There are various choices to be made during the course of algorithm 1, such as the choice of the subproblem to evaluate, and the way to divide the feasible region. An advantage of this algorithm is the clear decoupling of the continuous and discrete optimizers. Any usable continuous optimizer maybe used for solving the relaxed problem while the Branch-and-Bound (BB) method searches through the discrete space for the optimal solution. However, a major disadvantage is the speed issue. In the worst case the algorithm requires effort that grows exponentially with problem size. For instance, in the binary case, each $\mathcal{S}_i$ creates at most two new subproblems $\mathcal{S}_{i_1}$ and $\mathcal{S}_{i_2}$, whose set can be represented as a binary tree. As a result, there are total 2^m subproblems to be solved, where m is the number of discrete variables which is defined in 2.4.

Outer Approximation

The outer approximation scheme is another common technique for solving a class of MINLP problems. The outer approximation method approximates the nonlinear space utilizing linear constraints. Supporting linear hyperplanes are calculated at each iteration of the algorithm. Since we have efficient methods of solving linear programming problems, we may utilize these to solve for the MINLP problem. These linearizations overestimates the feasible region while at the same time the optimal solution is underestimated. Because many constraints are introduced, the problem may become intractable. For more detailed explanation of Outer Approximation methods, we recommend the following references [31, 41].

2.3 Black-Box Optimization

As we can see, the methodology behind these aforementioned MP techniques for solving optimization problems have often followed a pattern: Given a very specific class of problems with some known properties, design an algorithm to solve them. However, the applicability of these optimization algorithm is very restricted, because they work strictly based on assumptions about the properties of the objective functions. For example, the Branch-and-Bound (BB) method is especially designed for tackling mixed-integer nonlinear optimization problems. Unfortunately, these *divide-and-conquer* based techniques may fail when optimization problems possess the following properties:

(1) Only little knowledge about the objective function is available, such as optimization tasks which are mainly based on large-scale simulation models and the details of which often are inaccessible.

(2) The objective function is very complex, for instance multimodal, high dimensional and non-differentiable. As a consequence, the associated computational burden for this kind of optimization problems easily can become excessive for some classic MP methods.

From this perspective, these problems would fall into the class of *black-box* optimization problems, in which only little assumption about the objective function can be made or the objective function is too complex to be modeled. In this model of optimization, the objective function is often available for the optimizer as a *black-box* without assuming any local or global information [60]. Next, we will give a formal definition of *black-box* optimization based on [60].

Let us denote the finite input and output spaces by $\mathcal{X}$ and $\mathcal{Y}$, respectively. The general *black-box* optimization problem can be formally defined as the following equation system: For a given input decision parameters vector $\mathbf{x}$ in the feasible domain $\mathcal{X}$, after evaluation through the *black-box* function (e.g. simulator) a value $y = \Phi(\mathbf{x}) \in \mathcal{Y}$ is returned.

$$\Phi : \quad \mathcal{X} \to \mathcal{Y}, \quad x \mapsto \Phi(x) =: y \tag{2.5}$$

In the case of minimization, a *black-box* optimization problem is to find optimal $x^* \in \mathcal{X}$ such that $\Phi(x^*) \leq \Phi(x)$ for all $x \in \mathcal{X}$. The performance of the optimization algorithm used in this scenario, such as EAs, depends on the information collected by sampling different areas of the search space. More concretely, we explain this *black-box* optimization model by using Figure 2.1 below - a sample optimization problem with a simulator involved. In general, the combination of simulator and optimizer typically involves technical problems such as extracting the relevant simulator output data and aggregating the output data into a meaningful objective function. As one can see, the objective function is defined in regard to the output (Y) of the simulator. The optimizer then uses these values to search for optimal solution(s) $\mathbf{x}^*$.

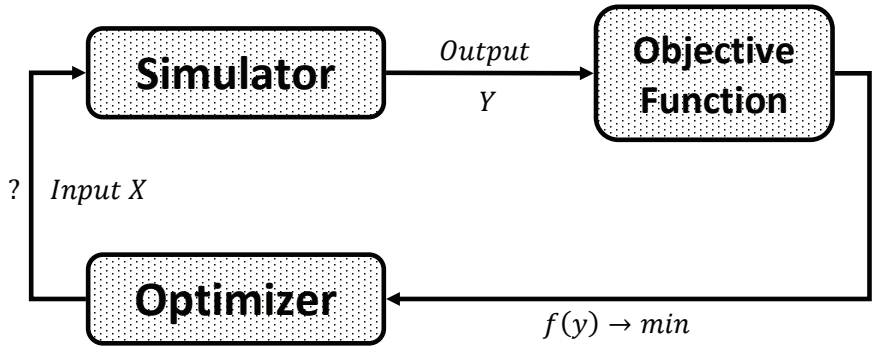


Figure 2.1: Outline of the general principle of coupling simulation and optimization.

2.3.1 Mixed-Integer Black-Box Optimization

If an input decision variables vector $\mathbf{x}$ is comprised of different types of variables - continuous, ordinal discrete (integer) and nominal discrete variables[6], the corresponding optimization problem is called mixed-integer *black-box* optimization problems. As we addressed in chapter 1, this kind of optimization problems will be the main focus of our research and will be studied in detail.

2.3.2 Related Works

Frequently, *black-box* optimization algorithms are classified based on whether they are deterministic or non-deterministic. More specifically, a deterministic method is to enumerate candidate solutions of the optimization task. Grid search and pattern search are two representatives among these deterministic methods. However,

[6]This is different from mixed-integer variables definition in MINLP problems, more detailed explanation are available in chapter 3

for most of real-world optimization problems, it becomes practically impossible because of the exponential growth with the number of dimensions. This is often referred to as the "*curse of dimensionality*" [10]. Contrary to these deterministic algorithms, stochastic algorithms (often heuristic) try to solve the problem by introducing some random choices in the search and this makes them more suitable for practical applications. In practice, there are a large number of stochastic algorithms available, such as simulated annealing, bayesian learning and clustering methods [60]. And our proposed MIES also belongs to this category.

2.4 Selected Applications

To illustrate the point of mixed-integer *black-box* optimization, let us have a look on two representative real-world optimization tasks - the *optimization of multilayer optical coatings* [7, 4] and the optimization of a chemical engineering plant [38]. For these two selected real-world applications, either its objective function is very complex or its expensive evaluation goes through a simulation software, the detail of which are inaccessible. In both cases, classical Mixed-Integer Nonlinear Programming (MINLP) techniques can not be easily applied. That is why it is highly desirable to develop new strategies to tackle problems of such a kind. We would like to mention some important characteristics, which were summarized in [4], of these practical applications as follows:

- Practical considerations require to find a robust optimum, i.e., an optimum that is insensitive with respect to small variations of the parameter values.
- The objective function is multimodal, high-dimensional, and non-differentiable, with a feasible region of the search space that is characterized by nonlinear constraints.
- In some cases, the objective function evaluation requires a run of a simulation model representing the real system to be optimized.
- Because of different parameter types, a standard representation such as binary strings or real-valued vectors is difficult to apply to these problems.

2.4.1 Optimization of Multilayer Optical Coatings

Problem Definition

The objective of the multilayer optical coatings (MOCs) design is to find a sequence of layers of certain materials and certain thicknesses (Figure 2.2), such that all unwanted frequencies are cut off, while the wanted frequencies pass without any reflection.

The *matrix method*, which is based on the Maxwell equations, is used to model MOCs as follows: the reflectance R for a given wavelength λ that depends on a

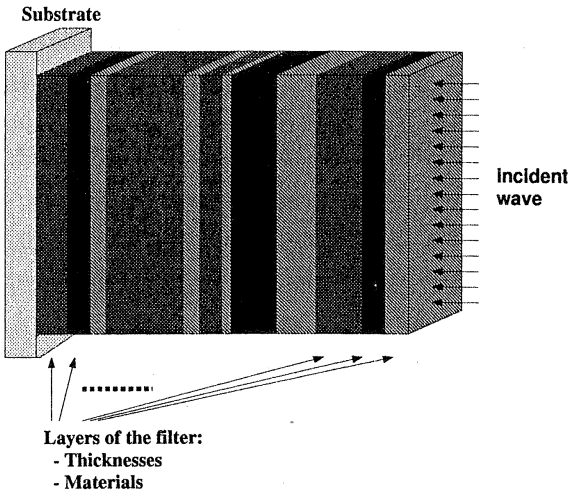


Figure 2.2: Multilayer optical coating. Figure courtesy of Bäck [4]

vector $\vec{d}$ of the geometric thicknesses of the layers and the refractive indices $\vec{\eta}$ of the materials of the corresponding layers:

$$R(\vec{d}, \vec{\eta}, \vec{\lambda}) = \frac{4\eta_a \eta_s}{|\eta_a B(\vec{d}, \vec{\eta}, \vec{\lambda}) + C(\vec{d}, \vec{\eta}, \vec{\lambda})|^2} \tag{2.6}$$

where η_a and η_s describe the refractive index of the adjacent medium and the substrate. B and C are non-linear terms of $\vec{d}, \vec{\eta}$ and $\vec{\lambda}$. The objective function f can be obtained by calculating the mean squared difference between the target wavelength profile and the profile of the give design sampled at m equidistant wavelengths λ_i in the range of interest.

Objective Function

The quality of a design can now be obtained by calculating the mean squared difference between the target wavelength profile and the profile of the given design sampled at m equidistant wavelengths λ_i in the range of interest. The objective function is defined as follows:

$$f(\vec{d}, \vec{\eta}) = \sqrt{\frac{1}{m} \sum_{i=1}^{m} R(\vec{d}, \vec{\eta}, \vec{\lambda})^2} \rightarrow \min \tag{2.7}$$

The fitness landscape of the objective function defined by equation 2.7, a three-dimensional plot of RMS-values for a two-layer filter with $\eta_1 = 2.2$, $\eta_2 = 4.2$, and d_1, d_2 varying in the range $0 - 20\,\mu m$ is shown in figure 2.3. The landscape

is characterized by parallel "waves" separated by valleys of increasing depth and decreasing width. As a consequence, optimization algorithms may be trapped within a local optimal valley.

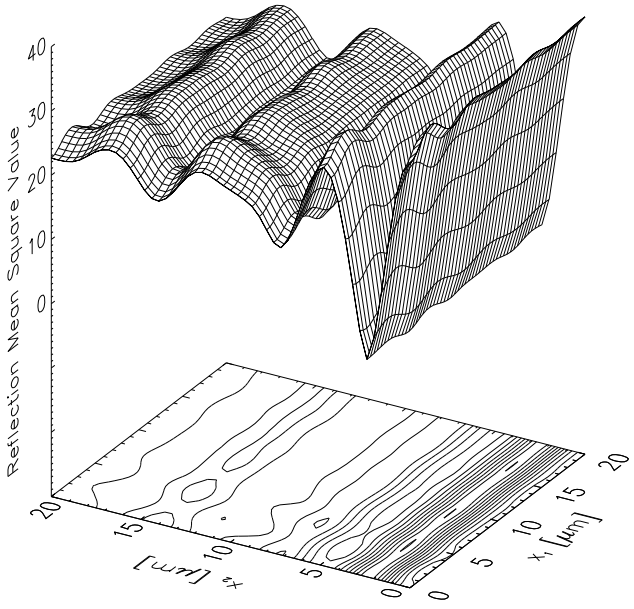


Figure 2.3: Topology of the RMS merit function in case of a fixed two-layer filter structure with $\eta_1 = 2.2$ and $\eta_2 = 4.2$. Figure courtesy of Bäck, et al. [7]

Optimization for MOCs design is a very difficult task because: (1) It involves real-valued thickness and integer-valued refractive indices variables; (2) Dimensionalities of decision variables are very high; (3) Equations which are used to compute objective values are very complex; (4) The number of dimensions is variable in the most general formulation of this problem.

2.4.2 Optimization of Chemical Engineering Plants

Problem Definition

The optimization of chemical engineering plants is another challenging application. The goal is to search for an optimal parameter configuration for a specific chemical engineering plant. A possible flowsheet for the Hydrodealkylation (HDA) process is displayed in Figure 2.4. The aim of the HDA process is the production of benzene from toluene. The annual profit is to be maximized.

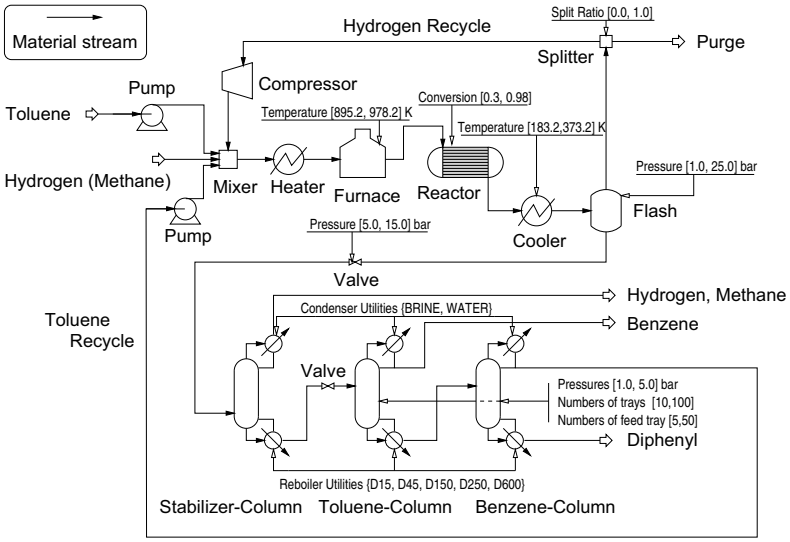


Figure 2.4: Flowsheet of the HDA process network with different chemical devices (unit operations) connected by material streams. The intervals and discrete sets indicate the domain of decision parameters to be optimized. Figure courtesy of Emmerich, et al. [38]

Objective Function

By definition, there are three types of decision parameters involved into the optimization procedure and they are indicated in Figure 2.4. The evaluation is carried out through one rigorous simulation model and this is presented in Figure 2.5 (cf. Figure 2.1). This optimization problem is also difficult because: (1) there exist different types of decision parameters; (2) fitness evaluation is based on a commercial simulation software and we have no access to details of its implementation. The classical techniques, such as BB and OA, are not applicable in this case.

2.5 Summary

In this chapter, different types of optimization problems are presented and the special attention is paid to mixed integer nonlinear programming problems, which occur a lot in real-world applications and are extremely hard to handle in practice. To tackle these very hard problems, some promising methods which come either from classical mathematical programming or heuristic domains are discussed in detail.

As we emphasized at the very beginning of this thesis, compared to "white-box" optimization problems, we are more interested in problems from "black-

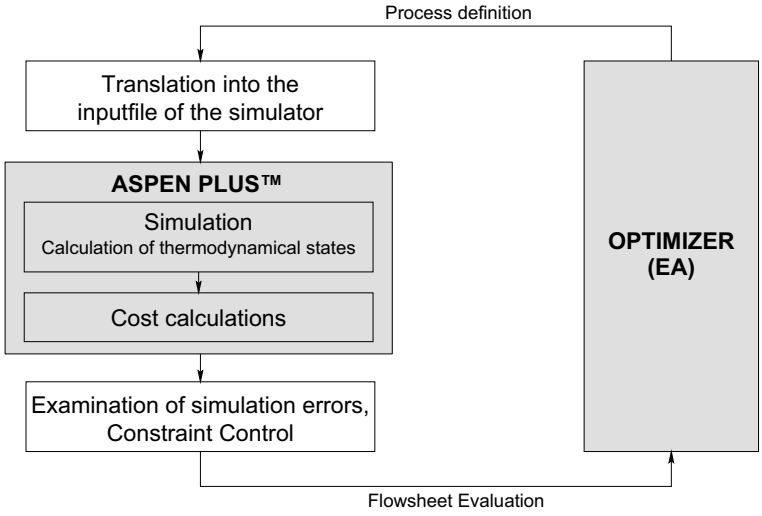


Figure 2.5: The interaction between the flowsheet simulator ASPEN PLUS^{TM} and the optimizer. Figures courtesy of Emmerich, et al. [38]

box" scenarios. Unclear objective function structure and high dimensionality make these *black-box* optimization problems more challenging and it is difficult to apply methods from the traditional mathematical programming research field, like Branch-and-Bound (BB) algorithm. By contrast, some heuristic methods provide possibilities to establish a connection between candidate solutions and the corresponding problem domain and lead us to a global/local optimum in an intelligent way.

Among many well studied heuristic methods, techniques from Evolutionary Computation domain, especially Evolution Strategies (ES), will be further investigated in this work.

Chapter 3

Mixed-Integer Evolution Strategies

In chapter 2 we introduced mixed-integer parameter optimization, especially in a *black-box* optimization scenario. Now, we will propose one promising algorithm, the so-called Mixed-Integer Evolution Strategies (MIES), which are capable to deal with the aforementioned *black-box* mixed-integer parameter optimization problems. This chapter is organized as follows: firstly, Evolutionary Algorithms (EAs) will be reviewed. Some important characteristics of each EAs model will be presented next. Then, Evolution Strategies (ES) will be discussed briefly, especially some important components of canonical $(\mu \overset{+}{,} \lambda)$- ES, such as *individuals structure*, *mutation*, *recombination* and *selection*. Finally, a more general framework for mixed-integer parameter optimization by using the MIES will be proposed in the rest of the chapter.

3.1 Evolutionary Algorithms

ES is one important branch of EAs, and other two branches are Genetic Algorithms (GAs) and Genetic Programming (GP). As we addressed, EAs derive from Darwin's theory of the survival of the fittest and mimic the process of organic evolution by using operators "*population*", "*mutation*", "*recombination*" and "*selection*" [52]. The better an individual performs under certain conditions the greater its chance to live for a longer and generate offspring, which in turn inherit the parental genetic information. Over the course of evolution, this leads to a penetration of the population with the genetic information of individuals of above-average fitness [5].

A high level abstraction of all essential components of standard implementations of evolutionary algorithms is given in Algorithm 2, for more detailed information about different evolutionary computation models (Genetic Algorithms, Evo-

lution Strategies and Genetic Programming) we recommend books [34, 58]. Based

Algorithm 2 General schema of an evolutionary algorithm

```
1: t := 0
2: Initialize population with random candidate solutions
3: Evaluate each candidate solution
4: while terminate condition is not satisfied do
5:    Select parents
6:    Recombine pairs of parents
7:    Mutate the resulting offspring
8:    Evaluate new candidate solution
9:    Select individuals for the next generation
10:   t := t + 1
11: end while
```

on this algorithm description, some important features of EAs can be summed up as follows: EAs are population based, they mostly use recombination or mutation to generate new candidate solutions, and they are stochastic.

3.2 Evolution Strategies

Evolution Strategies (ES) were founded in the early 1960s by Rechenberg and Schwefel at the Technical University of Berlin (TUB). In the beginning, ES were devised for the automatic design and analysis of consecutive experiments with stepwise variable adjustments driving a suitably flexible object into its optimal state in spite of environmental noise [12]. The first dissertation in the field of ES was completed by Rechenberg [95, 96] in 1971. In his thesis, Rechenberg analyzed the (1+1)-ES with Gaussian mutations on two very different real-valued functions - hypersphere and rectangular corridor function, and was able to show its convergence velocity, the achieved order of convergence and the optimal mutation strength. Born proposed population based $(\mu + 1)$-ES [15] and proved the convergence with probability 1. By applying principles from organic evolution in more rigorous way, Schwefel extended the (1+1)-ES towards a $(\mu + \lambda)$-ES and (μ, λ)-ES and proposed an ES capable of *self-adapting* some of its strategy parameters [106, 107]. In the following sections, we will explain the components of classical $(\mu \overset{+}{,} \lambda)$-ES in detail (cf. Algorithm 3), since it is seen as laying the foundations for our proposed Mixed-Integer Evolution Strategies (MIES).

3.2.1 Individuals Structure

Canonical Evolution Strategies (ES) are typically used for continuous parameter optimization ($\mathbb{R}^n \rightarrow \mathbb{R}$). For a given optimization problem $f(\vec{x}) \rightarrow min$, an individual of the evolution strategy consists of two components:

1. A candidate solution (a set of decision variables or control parameters), which is represented as $\vec{x} \in \mathbb{R}^n$;

2. *Endogenous strategy parameters*, which can be further divided into two sets, mutation step sizes $\vec{\sigma}$ and rotation angles $\vec{\alpha}$ ($\vec{\alpha}$ are not always used). $\vec{\sigma}$ essentially encode the n-dimensional normal distribution and are to be used to control certain statistical properties of the mutation operator. The $\vec{\alpha}$ values represent interactions between the step sizes used for different variables. Endogenous strategy parameters are very special in ES and can evolve during the whole evolution process.

Putting it all together, an individual in ES can be given in a more general form through a triple: $\vec{a} = (\vec{x}, \vec{\sigma}, \vec{\alpha})$.

3.2.2 Mutation

Mutations are the primary source of genetic variation in ES and are carried out by adding Δx_i to each x_i, where the Δx_i values are randomly drawn using the given normal distribution $N(0, \sigma)$ with zero mean and strandard deviation σ. In practice, the mutation step sizes $\vec{\sigma}$ are not set by the user, rather they are co-evolving with the solutions $\vec{x}$. To achieve this it is essential to modify the σ value first, and then mutate the x_i values with the new σ. The corresponding updating procedure can be defined as:

$$x_i' = x_i + N(0, \sigma') \tag{3.1}$$

where σ' is the mutated value of σ. Next, we will describe three special cases of mutation which are often used to mutate the value of σ in ES [34].

Uncorrelated Mutation with One Step Size

In this case, the same distribution is used to mutate each x_i, as a result each individual includes only one strategy parameter σ. The mutation mechanism is specified by the following formulas:

$$\begin{aligned} \sigma' &= \sigma \cdot e^{\tau \cdot N(0,1)} \\ x_i' &= x_i + \sigma' \cdot N_i(0,1) \end{aligned} \tag{3.2}$$

where σ is mutated each time step by multiplying it by a term $e^{\tau \cdot N(0,1)}$. $N(0,1)$ denotes a draw from the standard normal distribution, while $N_i(0,1)$ denotes a separate draw from the standard normal distribution for each variable x_i. The parameter τ can be interpreted as *learning rate* and readers can refer to [3] for a more detailed explanation.

Uncorrelated Mutation with n Step Sizes

Compared to one step size uncorrelated mutation, n step sizes mutation treats dimensions differently and can learn axes-parallel mutation ellipsoids. This is because that the fitness landscape can have a different slope in one direction than in another direction. Now, the mutation mechanism can be described as follows:

$$\begin{aligned} \sigma_i' &= \sigma_i \cdot e^{\tau \cdot N(0,1) + \tau' \cdot N_i(0,1)} \\ x_i' &= x_i + \sigma_i' \cdot N_i(0,1) \end{aligned} \tag{3.3}$$

where τ and τ' are called global and local learning rate respectively. The common base mutation $e^{\tau \cdot N(0,1)}$ allows an overall change of the mutability, while the $e^{\tau' \cdot N_i(0,1)}$ provides the flexibility to use different mutation strategies in different directions.

Correlated Mutation

This version of mutation allows the ellipses to have any orientation by rotating them with a covariance matrix C. The vectors $\vec{\sigma}$ and $\vec{\alpha}$ represent the complete covariance matrix of the n-dimensional normal distribution, where the covariances are given by rotation angles α_i describing the coordinate rotations necessary to transform an uncorrelated mutation vector into a correlated one. The complete mutation mechanism is performed according to:

$$\begin{aligned} \sigma_i' &= \sigma_i \cdot e^{\tau \cdot N(0,1) + \tau' \cdot N_i(0,1)} \\ \alpha_j' &= \alpha_j + \beta \cdot N_j(0,1) \\ \vec{x}' &= \vec{x} + N(\vec{0}, \mathbf{C}(\vec{\sigma}', \vec{\alpha}')) \end{aligned} \tag{3.4}$$

where $N(\vec{0}, \mathbf{C}(\vec{\sigma}', \vec{\alpha}'))$ denotes the correlated mutation vector and $\beta \approx 0.0873$. The details of this kind of mutation can be found in the literature e.g. in [100].

To make these different types of mutation more clear to readers, we illustrate how the degrees of freedoms grow as the number of strategy parameters is increased in Figure 3.1. As one can see, each of the three figures shows a two-dimensional ($n = 2$) hypothetical objective function topology, including isolines of equal objective function value and the location of a global optimum $\vec{x}^*$. The gray-shaded circles and ellipsoids correspond to individuals and their corresponding probability distribution to produce an offspring. For $n_\sigma = 1$ (the left figure), all distributions are spherically symmetric and only the radius of the circles is individually different. For $n_\sigma = 2$ (the middle figure), step sizes along one dimension might be different from the one along other dimension, such that preference search directions can be adjusted. For $n_\sigma = 2, n_\alpha = 1$, the ellipsoids can rotate and therefore allow an adjustment of arbitrary preference directions regardless of the coordinate system.

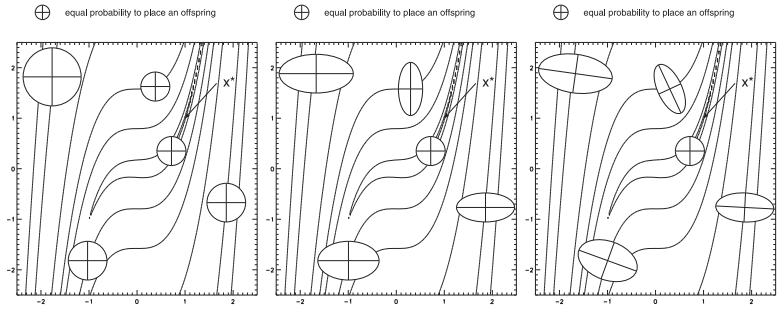


Figure 3.1: Schematic visualization of the three different types of self-adaptive mutation in Evolution Strategies. Left: $n_\sigma = 2$, middle: $n_\sigma = 2$, right: $n_\sigma = 2, n_\alpha = 1$. Figures courtesy of Thomas Bäck [6]

3.2.3 Recombination

The basic recombination scheme in Evolution Strategies (ES) involves two parents that create one offspring. According to the manner of recombining parent alleles, recombination can be classified into *discrete* and *intermediate* recombination. This can be formalized as follows:

$$z_i = \begin{cases} (x_i + y_i)/2 & \text{intermediate recombination} \\ \begin{cases} x_i & \text{if } U(0,1) > 0.5 \\ y_i & \text{otherwise} \end{cases} & \text{discrete recombination} \end{cases} \tag{3.5}$$

where $U(0,1)$ denotes a draw from one given uniform distribution. In cases where the whole population of μ individuals is used to generate one offspring, we are taking about *global* recombination. In practice, discrete recombination is recommended for the object/decision variable part and the intermediate recombination is suggested for the strategy parameters part. This scheme preserves diversity within the solution space while assuring a more cautious adaptation of strategies parameters.

3.2.4 Selection

The classical Evolution Strategies (ES) offer two different variants for selecting candidate solutions for the next iteration of the main loop of the algorithm: comma(indicated by ,) and plus (indicated by +) selection.

Comma selection

In this situation, after μ parents create $\lambda > \mu$[1] offspring by means of recombination and mutation, the best μ offspring are deterministically selected to replace

[1] The ratio of μ to λ is called selective pressure in ES, and $\mu/\lambda = 1/7$ is strongly recommended.

the parents. Using this kind of selection the best member of the population at generation $t+1$ might perform worse than the best individual at the previous generation t, Thus the strategy could escape from the local optimum and reach a better optimum. Comma selection is advantageous in the case of multimodal topologies.

Plus selection

In contrast, the $(\mu+\lambda)$ strategy selects the μ survivors from the union of μ parents and λ offspring, such that a monotonic course of evolution is guaranteed. This scheme is typically used in a steady-state setting or under circumstances where fitness deteriorations from one generation to the next are strictly unacceptable.

3.2.5 Results of Theoretical Study

The theoretical study on ES algorithms focuses mainly on the *convergence velocity* and *convergence reliability*. The former concentrates on the speed of the algorithm when a local optimum is approached, while the latter targets on proving that the algorithm is capable of finding the global optimum of the given objective function. For *convergence reliability*, so far, the convex case can be handled under strong simplifications of the objective functions that can be analyzed. The *convergence reliability* analysis yields a result for $t \to \infty$ independent of the objective function [11]. As one can see from the following sections, we will also apply the similar studies on proposed Mixed-Integer Evolution Strategies (MIES).

Based on the brief review of canonical Evolution Strategies (ES), especially $(\mu \overset{+}{,} \lambda)$-ES, we can make a short summary now. Compared to other EA models, e.g. Genetic Algorithms (GAs), Evolution Strategies (ES) are operating completely on a phenotypic level and this give them a good opportunity to utilize much more knowledge about the application. Moreover, the *self-adaptation* of strategy parameters provides a larger flexibility for ES over the complete evolution process [2, 52]. Last but not least, ES combines convergence velocity and convergence reliability in a more robust way. With respect to all these important properties, Evolution Strategies (ES) should prove to be global optimization algorithms and competitive with other global optimization methods.

3.3 Mixed-Integer Evolution Strategies

Mixed-Integer Evolution Strategies (MIES) is a special variant of an Evolution Strategies (ES) for the simultaneous optimization of continuous, integer, and nominal discrete parameters. It combines mutation operators of Evolution Strategies in the continuous domain [107], for integer programming [101], and for binary search spaces [3]. These operators have in common that they have certain desirable properties, such as symmetry, scalability, and maximal entropy, the details of which will be discussed later. The MIES was originally developed for optical

filter optimization [7, 105], and chemical engineering plant optimization [38, 47]. Recently, as discussed in this contribution, it has been used in the context of medical image analysis [75, 77]. In the latter work also its convergence behavior on various artificial landscapes was studied empirically, including a collection of single-peak landscapes in [38] and landscapes with multiple peaks in [77, 78].

3.3.1 Problem Definition

Many application problems from industry involve the simultaneous use of continuous, integer, and nominal discrete objective variables. The problem of mixed integer parameter optimization can be formalized as follows: let $r_1, \ldots, r_{n_r}$ denote a set of real-valued decision variables, $z_1, \ldots, z_{n_z}$ denote a set of integer decision variables, and $d_1, \ldots, d_{n_d}$ denote a set of nominal discrete decision variables, each of which is taken from a finite domain. The finite domains for the nominal discrete variables will be denoted with $D^{(1)}$, $\ldots$, $D^{(n_d)}$. We do not encode nominal discrete variables as integers, in order to exploit the fact that there is no meaningful a-priori ordering given for the domain of them. Furthermore, let $f : \mathbb{R}^{n_r} \times \mathbb{Z}^{n_z} \times D^{(1)} \times \cdots \times D^{(n_d)} \rightarrow \mathbb{R}$ denote an objective function to be minimized, $g_i : \mathbb{R}^{n_r} \times \mathbb{Z}^{n_z} \times D^{(1)} \times \cdots \times D^{(n_d)} \rightarrow \mathbb{R}$, $i = 1, \ldots, n_g$ and $h_j : \mathbb{R}^{n_r} \times \mathbb{Z}^{n_z} \times D^{(1)} \times \cdots \times D^{(n_d)} \rightarrow \mathbb{R}$, $j = 1, \ldots, n_h$ denote constraint functions. Then the mixed integer parameter optimization problem can be defined as:

$$\begin{aligned} f(\mathbf{r} \circ \mathbf{z} \circ \mathbf{d}) &\rightarrow \min \\ g_i(\mathbf{r} \circ \mathbf{z} \circ \mathbf{d}) &\leq 0, \quad i = 1, \ldots, m \\ h_j(\mathbf{r} \circ \mathbf{z} \circ \mathbf{d}) &= 0, \quad j = 1, \ldots, n \\ r_i &\in [r_i^{(min)}, r_i^{(max)}], \quad i = 1, \ldots, n_r \\ z_i &\in [z_i^{(min)}, z_i^{(max)}], \quad i = 1, \ldots, n_z \\ d_i &\in D^{(i)}, \quad i = 1, \ldots, n_d \end{aligned} \tag{3.6}$$

Here, the constants $r_i^{(min)}$ and $r_i^{(max)}$ define lower and upper bounds for the real variables and the constants $z_i^{(min)}$ and $z_i^{(max)}$ define lower and upper bounds for the integer variables. The symbol $\circ$ denotes tuple concatenation. In contrast with mixed integer nonlinear programming (cf. section 2.2), here three types of variables occur:

Continuous Variables, denoted with r_i, are taken from an interval $R_i \subset \mathbb{R}$ and their values are represented as floating point numbers. In the image processing field, for instance threshold parameters or a radius parameter for a geometrical shape are often represented as continuous variables.

Integer Variables, denoted with z_i, are taken from a range of integer variables $Z_i \subset \mathbb{Z}$. Important characteristics of integer variables are that their values have a smallest neighborhood (as opposed to continuous variables) and that

a linear ordering is defined on the values (as opposed to nominal discrete variables). The number of gray values in an image is a typical example of an integer variable in the image processing domain.

Nominal Discrete Variables, denoted with d_i, are variables the value of which are taken from a finite domain, denoted with D_i. Neither a metric nor an ordering is defined on this domain. An example is a variable which takes its value from a set of geometrical shapes (ellipse, square, triangle). Also binary variables (such as switches) belong to this class of variables.

As we are interested in the *black-box* scenario we assume that the structure of f, $g_i, i = 1, \ldots, n_g$ and $h_j, j = 1, \ldots, n_h$ is unknown or we only can make some very general statements about it, such as continuity assumptions based on a similarity measure defined on the search space [53]. As a result of this it becomes harder to apply standard techniques from mathematical programming - so called mixed-integer nonlinear programming methods [42] to solve them deterministically, such as outer approximation (OA) [31], branch-and-bound (BB) [14], and generalized Benders decomposition [43].

In cases where mathematical programming techniques fail, metaheuristics for mixed integer optimization can be an interesting method to heuristically search for solutions that improve the objective function value. In order to solve mixed integer optimization problems with metaheuristics two general approaches can be considered:

- ***Hierarchical Approach***: Separate the discrete problem from the continuous problem by optimizing the discrete variables in an higher level optimization problem and treating the optimization of the continuous parameters as a subproblem [119, 82, 96]
- ***Simultaneous Approach***: Optimize discrete and continuous parameters simultaneously . In this approach we consider that similarity of parameter vectors due to a appropriate metric as being positively correlated to the similarity in function values [47, 105].

The second method is worth requiring more attention and there are two reasons why we favor this approach over the hierarchical approach: Firstly, the hierarchical approach requires a sub-optimization of continuous parameters for each set of discrete parameters chosen in the outer level. This can be very time consuming. Secondly, in the hierarchical approach it is difficult to consider correlations between discrete and continuous variables, as they are strictly separated from each other. In the following, we will discuss the design philosophy of MIES in detail and present several important properties.

3.3.2 Algorithm description

The problem of designing an evolution strategy for a new type of search space breaks down into three subtasks:

(1) definition of the generational cycle,

(2) definition of the individual representation,

(3) definition of variation operators for the representation of choice.

These subtasks will be discussed next.

The chosen algorithm will be an instantiation of a $(\mu \overset{+}{,} \lambda)$-ES for mixed-integer spaces. It generalizes the more common $(\mu \overset{+}{,} \lambda)$-ES for continuous spaces, the dynamic behavior of which was subject to thorough theoretical and empirical studies. For instance, Schwefel [107] compared it to traditional direct optimization algorithms and Bäck [3] to other evolutionary algorithms. Theoretical studies of the convergence behavior of the ES were carried out for instance by Beyer [11], Oyman [88] and Rudolph [102]. A comparison to other evolutionary algorithms such as Genetic Algorithms can be found in Bäck [3, 52]. The results indicate that the ES is a robust optimization tool that can deal with a large number of practically relevant function classes, including discontinuous and multimodal functions. In addition, the ES performance scales well with the search space dimension.

Generational Cycle

The main procedure of the ES is described in Algorithm 3. After a uniform random initialization and evaluation of the first population $P(0)$ of μ individuals (parameter vectors taken from an individual space I) and setting the generation counter t to zero the main loop of the algorithm starts. In a first step of the iteration the algorithm generates the set $Q(t)$ of λ new offspring individuals, each of them obtained by the following procedure:

Two individuals are randomly selected from $P(t)$ and an offspring is generated by recombining these parents and then mutating (random perturbation) the individual resulting from the recombination. In the next step of the iteration, the λ offspring individuals are evaluated using the objective function to rank the individuals (the lower the objective function value the better the rank). In case of a $(\mu + \lambda)$ selection, the μ best individuals out of the union of the λ offspring individuals and the μ parental individuals are selected. In case of a (μ, λ) selection the μ best individuals out of the λ offspring individuals are selected. The selected individuals form the new parent population $P(t+1)$. After this, the generation counter is incremented. The generational loop is repeated until the termination criterion[2] is fulfilled.

Representation

An individual in an Evolution Strategy contains the information about one solution candidate. The contents of parent individuals is inherited by offspring individuals and is subject to variation. The standard representation of a solution in

[2] In most cases a maximal number of generations is taken as termination criterion.

Algorithm 3 $(\mu \overset{+}{,} \lambda)$-Evolution Strategy

1: $t \leftarrow 0$
2: **initialize** Population $P(t) \in \mathbb{I}^{\mu}$
3: **evaluate** the μ initial individuals with objective function f
4: **while** Termination criteria not fulfilled **do**
5: **for all** $i \in \{1, \ldots, \lambda\}$ **do**
6: choose uniform randomly parents c_{i_1} and c_{i_2} from $P(t)$ (repetition is possible)
7: $x_i \leftarrow \text{mutate}(\text{recombine}(c_{i_1}, c_{i_2}))$
8: $Q(t) \leftarrow Q(t) \cup \{x_i\}$
9: **end for**
10: $P(t+1) \leftarrow \mu$ individuals with best objective function value from $P(t) \cup Q(t)$ (plus), or $Q(t)$ (comma)
11: $t \leftarrow t + 1$
12: **end while**

an ES individual is a continuous vector. In addition parameters of the probability distribution used in the mutation (such as standard-deviations or *step-sizes*) are stored in the individual. The latter parameters are referred to as strategy parameters.

To solve mixed-integer problems with an Evolution Strategy we extend the real-vector representation of individuals by introducing integer and nominal discrete variables as well as strategy parameters related to them. The domain of an individual then reads:

$$\mathbb{I} = R_1 \times \cdots \times R_{n_r} \times Z_1 \times \cdots \times Z_{n_z} \times D_1 \times \cdots \times D_{n_d} \times A_s$$

Here, A_s denotes the domain of strategy parameters and is defined as:

$$A_s = \mathbb{R}_+^{n_\sigma + n_\varsigma} \times [0,1]^{n_p}, n_\sigma \leq n_r, n_\varsigma \leq n_z, n_p \leq n_d$$

An individual of a population P(t) in generation t is denoted as:

$$\vec{a} = (r_1, \ldots, r_{n_r}, z_1, \ldots, z_{n_z}, d_1, \ldots, d_{n_d}, \sigma_1, \ldots, \sigma_{n_\sigma}, \varsigma_1, \ldots, \varsigma_{n_\varsigma}, p_1, \ldots, p_{n_p})$$

The so-called object variables $r_1, \ldots, r_{n_r}, z_1, \ldots, z_{n_z}, d_1, \ldots, d_{n_d}$ determine the objective function value and thus the fitness of the individual (cf. Equation 3.6). Here, $r_1, \ldots, r_{n_r}$ denote real valued, $z_1, \ldots, z_{n_z}$ integer valued, and $d_1, \ldots, d_{n_d}$ nominal discrete variables. The so-called strategy-variables $\sigma_1, \ldots, \sigma_{n_\sigma}$ are standard deviations used in the mutation of the real valued variables, $\varsigma_1, \ldots, \varsigma_{n_\varsigma}$ denote mean step sizes in the mutation of the integer parameters. Finally, $p_1, \ldots, p_{n_p}$ denote mutation probabilities (or rates) for the nominal discrete object parameters. All these parameters are subject to inheritance, recombination, and mutation within Algorithm 3. Object variables are initialized uniformly within their domain.

Recombination

The recombination operator can be subdivided into two steps, selection of the parents and recombination of the selected parents. Here we will focus on local recombination which works with two recombination partners. In this work we will apply local recombination which works with two recombination partners. The two recombination partners $c_1 \in I$ and $c_2 \in I$ are chosen randomly according to a uniformly distribution from the parental generation for each of the offspring individuals. The information contained in these individuals is combined in order to generate an offspring individual. In Evolution Strategies two recombination types are commonly used: dominant and intermediate recombination [107]. In a *dominant* (or) *discrete recombination* the operator chooses randomly one of the corresponding parental parameters for each offspring vector position. *Intermediate recombination* computes the arithmetic mean of both parents and thus, in general, can only be applied for continuous object variables and strategy variables. In Mixed-Integer ES, dominant recombination is used for the solution parameters while intermediate recombination is used for the strategy parameters.

Mutation

For the parameter mutation, standard mutations with maximal entropy for real, integer and discrete parameter types are combined, as described in [3, 101, 105, 107]. The choice of mutation operators was guided by the following requirements for a mutation in general search spaces (e.g. [30, 101, 11]):

- **Accessibility:** Every point of the individual search space should be accessible from any other point by means of a finite number of applications of the mutation operator.
- **Feasibility:** The mutation should produce feasible individuals. This guideline can be crucial in search spaces with a high number of infeasible solutions.
- **Symmetry:** No additional bias should be introduced by the mutation operator.
- **Similarity:** Evolution strategies are based on the assumption that a solution can be gradually improved. This means it must be possible to generate similar solutions by means of mutation.
- **Scalability:** There should be an efficient procedure, by which the strength of the impact of the mutation operator on the fitness values can be controlled.
- **Maximal Entropy:** If there is no additional knowledge about the objective function available the mutation distribution should have maximal entropy [101]. By this measure a more general applicability can be expected.

Respecting these guidelines, the following operators have been selected in [38]: The mutation of *continuous variables* is described in Algorithm 4. The new individual is obtained by adding a normal distributed random perturbation, to the old values of the vector. The corresponding standard deviations are also subject to the evolution process and are thus multiplied in each step by a logarithmic distributed random number. Schwefel [107] termed the resulting process as *self-adaptive*, because the adaptation of the mutation parameters is governed by an evolutionary process itself. The general idea behind self-adaptation is that, if a set of different individuals is generated, each with a different probability distribution, the individual with the best object variables is also likely to be the one with the best probability distribution that lead to the generation of these object variables. Thus the parameters of this probability distribution are also inherited by the offspring individual. We will now spend some remarks on the properties of the

Algorithm 4 Mutation of real valued parameters

input: $r_1, \ldots, r_{n_r}, \sigma_1, \ldots, \sigma_{n_r}$
output: $r'_1, \ldots, r'_{n_r}, \sigma'_1, \ldots, \sigma'_{n_\sigma}$
control parameters: $n_\sigma \in \{1, n_r\}$
$N_c \leftarrow N(0,1)$ {Generate and store a normally distributed random number}
$\tau \leftarrow \frac{1}{\sqrt{2n_r}}; \tau' \leftarrow \frac{1}{\sqrt{2\sqrt{n_r}}}$ {Initialize global and local learning rate}
if $n_\sigma = 1$
{Single step-size mode} **then**
  $\sigma'_1 = \sigma_1 \exp(\tau N_c)$
  for all $i \in \{1, \ldots, n_r\}$ **do**
    $r'_i \leftarrow r_i + \sigma'_1 N(0,1))$
  end for
else
  {Multiple step-size mode}
  for all $i \in \{1, \ldots, n_r\}$ **do**
    $\sigma'_i \leftarrow \sigma_i \exp(\tau N_c + \tau' N(0,1))$
    $r'_i \leftarrow r_i + \sigma'_i N(0,1))$
  end for
end if
{Interval boundary treatment}
for all $i \in \{1, \ldots, n_r\}$ **do**
  $r'_i \leftarrow T_{[r_i^{min}, r_i^{max}]}(r'_i)$
end for

normal distributions, as they are responsible for the choice of this type of distribution for mutating continuous variables. Among all continuous distributions with finite variance on $\mathbb{R}$, the normal distribution possesses the maximum entropy [67]. The multidimensional normal distribution is symmetrical to its mean value and unimodal. The step-sizes represent standard deviations of the multi-dimensional normal distribution for each real-valued variable. By variation of these standard

deviations the impact of the mutation on the variable vector in terms of similarity can be scaled.

As opposed to continuous variables, integer variables are less commonly used in evolution strategies. The mutation procedure for integer variables is borrowed from [101], where the replacement of the normal distributed random variables by the difference between two geometrical distributed variables has been suggested. Among distributions defined on integer spaces the multidimensional geometric distribution is one of the distributions of maximum entropy and finite variance, as the original geometric distribution is single-tailed, Rudolph [101] suggested to use instead the difference $Z_1 - Z_2$ of two geometrically distributed random variables. The resulting distribution is depicted for the 1-D case in Figure 3.2 and 2-D case in Figure 3.3. It is l_1-symmetrical[3] centered around its mean value, unimodal and it has an infinite support, thereby symmetry and accessibility of the mutation is obtained. Accessibility is given in a strict sense: each possible configuration can be reached with a finite probability in a single step. The strength of the mutation for the integer parameters is controlled by a set of step-size parameters which represent the mean value of the absolute variation of the integer object variables. The details of this mutation operator are found in Algorithm 5. Note that a geometrically distributed random value with mean step size parameter ς can be generated by transforming a uniformly distributed random value u, using:

$$z = \left\lfloor \frac{\ln(1-u)}{\ln(1-\psi)} \right\rfloor, \quad \psi = 1 - \varsigma \left(1 + \sqrt{1+\varsigma^2}\right)^{-1} \tag{3.7}$$

The width of the distribution can be controlled by the parameter ς, the mean value of the exponential distribution (cf. 3.2, for a derivation, see [101]). Excepting the different distribution types used, it is very similar to the real valued mutation operator in Algorithm 4. Self-adaptation is used to control the width parameter(s). The mutation of the width parameter is done as in [101] using a global learning rate τ and local learning rate τ'. Since a mean step-size below 1 is not useful for integer problems the mutated mean step-size is set back to 1, whenever its mutation results in a value less then 1.

Since we have to keep integer and continuous parameters within their feasible interval, the mutation operators need to be extended. Therefore a transformation function $T_{[a,b]}$ is applied to the mutation operators, that brings parameters beyond boundaries back into the feasible domain. For the continuous and integer parameters this is achieved by (an illustration of how the transformation function works can be found in Figure 3.4.):

$$T_{[a,b]} = a + (b-a)\frac{2}{\pi}\sin^{-1}(|\sin\left(\frac{\pi(x-a)}{2(b-a)}\right)|) \tag{3.8}$$

The transformation function can be viewed as a reflection at the interval boundaries. Given a step-size of the mutation, we may consider this to be the length

[3] The l_1-norm of a vector $\mathbf{z} \in \mathbb{Z}^n$ is defined as $\sum_{i=1}^{n} |z_i|$.

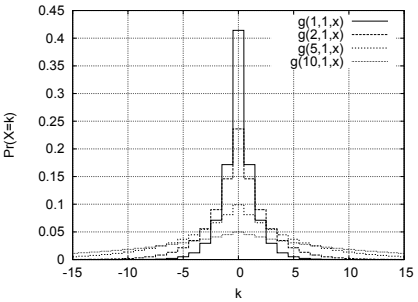


Figure 3.2: 2-D representation of the distribution obtained as the difference of two geometrical distributions for different values of ς.

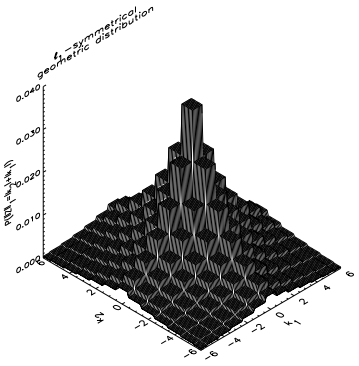


Figure 3.3: 3-D representation of the distribution obtained as the difference of two geometrical distributions. Figure courtesy of Günter Rudolph.

a particle has to travel within the interval. Starting in the direction of the original unbounded mutation, whenever it meets with an interval boundary the direction is inverted until the total length of the unbounded mutation has been covered. The method can be efficiently implemented as seen in algorithm 6. Unlike other mappings, the limiting distribution of the random walk $X_{t+1} = T_{[a,b]}(X_t + \sigma N(0,1)), t = 1, 2, \ldots$ is the uniform distribution. This means that there are no preferred regions of the search space in the long term in case of neutral selection and thus bias is avoided. In order to prevent a loss of causality, the step-size should be kept smaller than the interval width. We recommend a maximal stepsize of $0.2(b-a)$.

$y = (x - a)/(b - a)$
if $\lfloor y \rfloor \bmod 2 = 0$ **then**
  $y' = |y - \lfloor y \rfloor|$
else
  $y' = 1 - |y - \lfloor y \rfloor|$
end if
$x' = a + (b - a)y'$
return x'

Algorithm 6: Computation $T_{[a,b]}(x)$, for interval boundaries a and b.

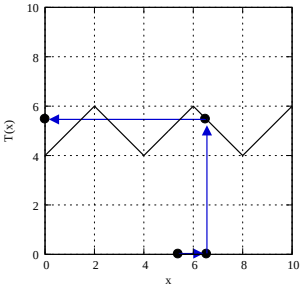


Figure 3.4: An illustration to show working mechanism of the transformation function (a=4, b=6).

Algorithm 5 Mutation of integer parameters

input: $z_1, \ldots, z_{n_z}, \varsigma_1, \ldots, \varsigma_{n_\varsigma}$
output: $z'_1, \ldots, z'_{n_z}, \varsigma'_1, \ldots, \varsigma'_{n_\varsigma}$
control parameters: $n_\varsigma \in \{1, n_z\}$
$N_c \leftarrow N(0,1)$
$\tau \leftarrow \frac{1}{\sqrt{2n_z}}; \tau' \leftarrow \frac{1}{\sqrt{2\sqrt{n_z}}}$
if $n_\varsigma = 1$ **then**
  {Single step-size mode}
  $\varsigma'_1 \leftarrow \max(1, \varsigma_1 \exp(\tau N_c))$
  for all $i \in \{1, \ldots, n_z\}$ **do**
    $u_1 \leftarrow U(0,1); u_2 \leftarrow U(0,1); \psi \leftarrow 1 - (\varsigma'_1/n_z)\left(1 + \sqrt{1 + (\frac{\varsigma'_1}{n_z})^2}\right)^{-1}$
    $G_1 \leftarrow \left\lfloor \frac{\ln(1-u_1)}{\ln(1-\psi)} \right\rfloor; G_2 \leftarrow \left\lfloor \frac{\ln(1-u_2)}{\ln(1-\psi)} \right\rfloor$
    $z'_i \leftarrow z_i + G_1 - G_2$
  end for
else
  {Multiple step-size mode}
  for all $i \in \{1, \ldots, n_z\}$ **do**
    $\varsigma'_i \leftarrow \max(1, \varsigma_i \exp(\tau N_c + \tau' N(0,1)))$
    $u_1 \leftarrow U(0,1); u_2 \leftarrow U(0,1); \psi \leftarrow 1 - (\varsigma'_i/n_z)\left(1 + \sqrt{1 + (\frac{\varsigma'_i}{n_z})^2}\right)^{-1}$
    $G_1 \leftarrow \left\lfloor \frac{\ln(1-u_1)}{\ln(1-\psi)} \right\rfloor; G_2 \leftarrow \left\lfloor \frac{\ln(1-u_2)}{\ln(1-\psi)} \right\rfloor$
    $z'_i \leftarrow z_i + G_1 - G_2$
  end for
end if
{Interval boundary treatment}
for all $i \in \{1, \ldots, n_z\}$ **do**
  $z'_i \leftarrow T_{[z_i^{min}, z_i^{max}]}(z'_i)$
end for

Finally, a mutation of the discrete parameters is carried out with a mutation probability as described in Algorithm 7. The probability is a strategy parameter for each discrete variable. Each new value is chosen randomly (uniformly distributed) out of the finite domain of values. The application of a uniform distribution is due to the principle of maximal entropy, since the assumption was made that there is no reasonable order defined between the discrete values.

To reason about requirements like symmetry and scalability we need to define a distance measure on the discrete sub-space. The assumption that there is no order, which can be defined on the finite domains of discrete values, leads to the application of the overlap distance[4] measure: $\Delta((d_1, \ldots, d_{n_d}), (d'_1, \ldots, d'_{n_d})) =$

[4]For the binary case this corresponds to the Hamming distance.

$\sum_{i=1}^{n_d} H(d_i = d'_i)$ with $H(true) = 1; H(false) = 0$ as a similarity measure for guiding the design of the mutation operator.

A self-adaptation of the mutation probability for the discrete parameters is achieved by a logistic mutation of these parameters, generating new probabilities in the feasible domain. The logistic transformation function is recommended and discussed by Schütz [105]. The basic idea of this transformation is to keep the variables within the range $[0, 1]$. Given an original mutation probability $p \in [0, 1]$, it can be mutated using the following procedure:

$$p' = \frac{1}{1 + \frac{1-p}{p} * \exp(-\tau' N(0,1))} \tag{3.9}$$

Here $N(0, 1)$ denotes a function that returns a normally distributed random number. We recommend to employ a second transformation function ($T_{p_{min},p_{max}}$) that keeps the value of p in the interval $[1/(3n_d), 0.5]$. The upper bound of 0.5 for the mutation probability is motivated by the observation that the mutation loses its causality once the probability exceeds the value of about 0.5. The lower bound is used to prevent the mutation probability from being too close to 0, in which case the MIES becomes insensitive to changes of that parameters. In case of $p = 1/(3n_d)$ a discrete mutation can be expected in every third application of the mutation operator.

Depending on the discrete subspace, it can be advantageous to use a single mutation probability instead of many individual mutation probabilities $p_1, \ldots, p_{n_d}$. In case of a single mutation probability, for each position of the discrete subvector it is decided independently, but with the same probability, whether to mutate this position or not. By adapting the mutation rate, the average number of mutations on the discrete values is adjusted to the mean step-size if the Hamming distance is considered as metric.

3.3.3 Step-size Adaptation Study

Previous work already showed that self adaptive ES are able to converge to optima of simple functions in arbitrary precision by using step size adaptation. However, it is an open question whether the self adaptation indeed is capable of helping the step size close to a optimal value that optimizes the progress rate. A theoretical analysis of the step-size adaptation is very difficult, even for simple models such as the sphere models. In this chapter we used a semi-empirical approach by approximating the local progress rate at a given distance to the optimum statistically for different step-sizes, in order to find the optimal step-size s^* that maximizes the local progress of the MIES. This computation is repeated for different stages of the evolution and each time the empirically found optimal step-size $\hat{s}^*$ is compared to the current step-size of the MIES.

Though we use the approach in this article only for the analysis on the continuous, integer, and discrete sphere model, it is applicable also for analysis on

Algorithm 7 Mutation of nominal discrete parameters

input: $d_1, \ldots, d_{n_d}, p_1, \ldots, p_{n_p}$
output: $d'_1, \ldots, d'_{n_d}, p'_1, \ldots, p'_{n_p}$
control parameters: $n_p \in \{1, n_d\}$
$N_c \leftarrow N(0,1)$
$\tau \leftarrow \frac{1}{\sqrt{2n_d}}$; $\tau' \leftarrow \frac{1}{\sqrt{2\sqrt{n_d}}}$
if $n_p = 1$ then **then**
  {Single step-size mode}
  $p' \leftarrow \frac{1}{1+\frac{1-p}{p} * \exp(-\tau * N_c)}$
  $p' = T_{[0.01,0.5]}(p')$
  for all $i \in \{1, \ldots, n_p\}$ **do**
    if $U(0,1) < p'$ **then**
      choose a new element uniform distributed out of $D_i \setminus \{d_i\}$
    end if
  end for
else
  {Multiple step-size mode}
  for all $i \in \{1, \ldots, n_p\}$ **do**
    $p'_i \leftarrow \frac{1}{1+\frac{1-p_i}{p_i} * \exp(-\tau * N_c - \tau' * N(0,1))}$
    $p'_i = T_{[1/(3n_d),0.5]}(p'_i)$
    if $U(0,1) < p'_i$ **then**
      choose a new element uniform distributed out of $D_i \setminus \{d_i\}$
    end if
  end for
end if

any other test problem for which the optimum is known and the evaluation of the objective function is fast.

In correspondence with [11, 12], the local progress rate $\phi(s, \mathbf{x})$ for a step-size s is the expectation of the distance covered towards the optimum in one mutation step [13] starting from position $\mathbf{x}$. Consider a mutation operator mut_s parameterized by the step-size, an objective function $f : \mathbb{I} \to \mathbb{R}$ with single optimum $\mathbf{x}^* \in X$, and a position $\mathbf{x}$ in the metric search space $(\mathbb{X}, d)$. Then

$$\phi(s, \mathbf{x}) = E\Big(\frac{\max\{0, R(\mathbf{x}) - R(\text{mut}_s(\mathbf{x}))\}}{R(\mathbf{x})}\Big), R(\mathbf{x}) = d(\mathbf{x}, \mathbf{x}^*) \tag{3.10}$$

In order to compute $\phi(s, \mathbf{x})$ we compute the sample mean for $M = 50000$ samples:

$$\hat{\phi}(s, \mathbf{x}) = \frac{1}{M} \sum_{i=1}^{M} \frac{\max\{0, R(\mathbf{x}) - R(\text{mut}_s^i(\mathbf{x}))\}}{R(\mathbf{x})}. \tag{3.11}$$

Depending on the parameter type, a different distance measurement is used to compute $d(\mathbf{x}, \mathbf{x}^*)$ (cf. Equation 3.12). For instance, the *Euclidean* distance is ap-

plied to continuous parameters, *Manhattan* distance is applied to integer parameters, and for discrete parameters we choose the *Overlap* distance function.

$$d(\mathbf{x}, \mathbf{x}^*) = \begin{cases} \sqrt{\sum_i^n (x_i - x_i^*)^2} & \text{if } x_i \in \mathbb{R} \quad (\textit{Euclidean Distance}) \\ \sum_i^n |x_i - x_i^*| & \text{if } x_i \in \mathbb{Z} \quad (\textit{Manhattan Distance}) \\ \sum_i^n \mathbf{I}(x_i, x_i^*) = \begin{cases} 0 & \text{if } (x_i = x_i^*) \\ 1 & \text{if } (x_i \neq x_i^*) \end{cases} & \text{if } x_i \in \mathbb{D} \quad (\textit{Overlap Distance}) \end{cases} \tag{3.12}$$

The optimal step-size s^* is approximated by means of a graphical plot. The value of $\hat{\phi}(s, \mathbf{x})$ is computed for an equidistant set of $L = 40$ points $s_1, s_2, \ldots, s_L$ in the interval $[0, \text{smax}(\mathbf{x})]$. The upper interval boundary smax is chosen as $\text{smax}(\mathbf{x}) = 2|\mathbf{x} - \mathbf{x}^*|$ and as $\text{smax} = 1$ whenever $\text{smax}(\mathbf{x})$ represents a mutation rate. It is plausible that an optimal step-size exists, as first of all the value of ϕ is always positive and for $s = 0$ it should take the value of 0. Whenever the step-size s gets too large the progress rate also approaches zero, since the probability to step beyond the region of improvement gets very high. The upper bound of 1 in cases where s represents a probability seems to be a natural choice. However, in case of a high search space dimensionality the optimal value of s might be very close to zero and a reduction of the upper bound can be considered. The research question is whether the MIES can find and keep the step-size that maximizes the local progress rate.

The experimental setup is as follows: We compute the optimal step-size at different stages of the evolution. Let $\mathbf{x}^{(t)}$ denote the parent individual in the t-th generation. We then compute $\hat{\phi}(s_i, \mathbf{x}^{(1)})$ for $s_i \in [0, \text{smax}(\mathbf{x}^{(t)})], i = 1, \ldots, L$, and graphically compare the peak of the graph of $\hat{\phi}(s_i, \mathbf{x}^{(t)})$ with the step-size $s^{(t)}$ used by the MIESES at different stages of the evolution. The search space dimension is 15 and the variable range is $[-1000, 1000]$ for the integer and discrete variables, and $\{0, \ldots, 9\}$ for the discrete variables. As a test problem the minimization of the sum of squares of the variables is used. Continuous, integer and discrete spaces were studied separately.

Results for different parameter types are shown in Figure 3.5 (continuous), 3.6 (integer) and 3.7 (discrete). For all three cases the optimum of the step-size is found and tracked. This proves that, at least for relatively simple – but nevertheless high-dimensional – problems, the self-adaptation of the step-sizes works. Note, that the scale of the plots in Figures changes during the run by orders of magnitude. In order to achieve the results we used a learning rate of 0.5. It is also possible to use the recommended values for τ (cf. 3.3.2). For this setting we achieved worse results, although the right order of magnitude for the step-size was still obtained. In summary, this study shows that all distributions used for mutation can be controlled in their width by means of scaling parameters, allowing self-adaptation to be implemented. In the following part, we will present some result of theoretical study of MIES on *convergence reliability*.

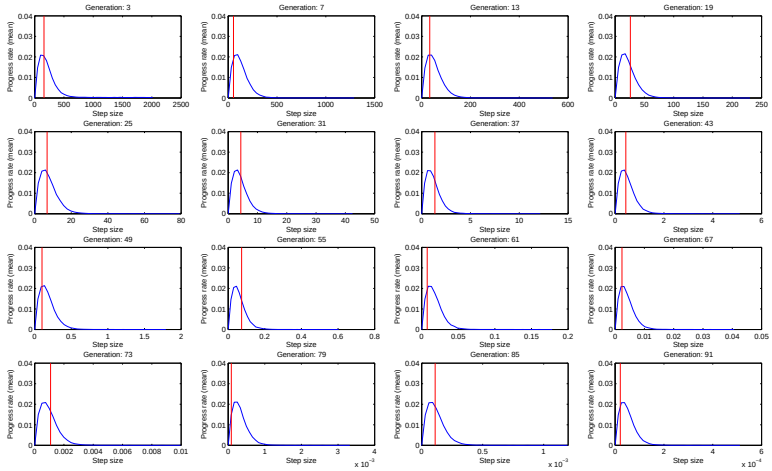


Figure 3.5: Comparison between the peak of the graph of $\hat{\phi}(s_i, x^{(t)})$ with the stepsize $s^{(t)}$ (vertical line) of continuous variables used by the MIES at that stage of the evolution. The step-size $s^{(t)}$ is found by the self-adaptation within the $(4, 28)$-MIES (i.e. without knowledge of $\hat{\phi}$).

3.3.4 Global Convergence Properties

If certain regularity requirements are met, it is possible to prove strong probabilistic convergence of the MIES for $t \to \infty$ towards the global optimum. The theorem generalizes a theorem on the ES for continuous spaces by Born [15]. Both the plus and the comma strategy are considered, and for the convergence analysis the best solution found so far, i.e. $\mathbf{x}^t_{best}$, will be considered.

Definition 1
A function $f : C \to \mathbb{R}$ is called *regular*, if:

(A) f is continuous,

(B) $C \subseteq \mathbb{R}^n$ is a closed set,

(C) $\forall \mathbf{x}' \in C : \forall \epsilon > 0 :$ the set $\{\mathbf{x} \in \mathbb{C} | \mathbf{x} \neq \mathbf{x}' \wedge f(\mathbf{x}) \leq f(\mathbf{x}') + \epsilon\}$ is non-empty.

Definition 2
Let $A' \subseteq A$ and let $g : A \to B$ a function. By $g|_{A'}$ we denote the restriction of the function g defined by $g|_{A'}(a') := g(a')$ where $a' \in A'$.

Given these technical preliminaries we can state the following theorem:

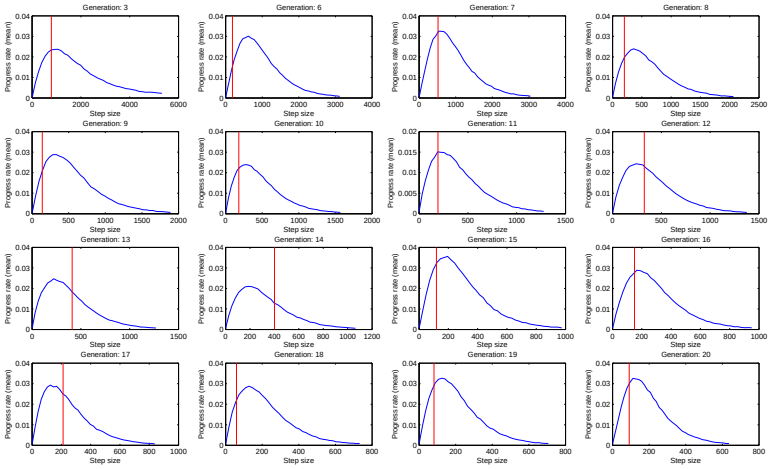


Figure 3.6: Comparison between the peak of the graph of $\hat{\phi}(s_i, x^{(t)})$ with the stepsize $s^{(t)}$ (vertical line) of integer variables used by the MIES at that stage of the evolution. The step-size $s^{(t)}$ is found by the self-adaptation within the $(4, 28)$-MIES (i.e. without knowledge of $\hat{\phi}$).

Theorem 3
Let $f : \mathbb{R}^n \times \mathbb{A} \to \mathbb{R}$ denote a mixed-integer function, and $f|_{\mathbb{R}^n \times \{a\}}$ is a regular function for at least one $a^* \in \mathbb{A}$ which is optimal. Then for a $(\mu + \lambda)$ MIES with lower limit $\sigma_{min} > 0$ for the stepsizes and mutation rates, the series $f(\mathbf{x}^t_{best})_{t=1,2,\ldots}$ converges with probability one to the global minimum of f, i.e.

$$\Pr\{\lim_{t\to\infty} \Delta_t = 0\} = 1, \text{with } \Delta_t = f(\mathbf{x}^t_{best}) - f^* \geq 0 \tag{3.13}$$

Here t represents the number of iterations, and f^* denotes the global optimum.

Proof (Proof)
From the construction of the algorithm it follows:

$$\forall t \geq 0 : \Delta_{t+1} \leq \Delta_t \tag{3.14}$$

and from the definition of a global optimum we get

$$\forall t \geq 0 : \Delta_t \geq 0 \tag{3.15}$$

With proposition 3.14 and 3.15 it follows that $\Delta_t (t = 1, 2, \ldots)$ has a limit value

$$\lim_{t\to\infty} \Delta_t = \Delta_\infty \tag{3.16}$$

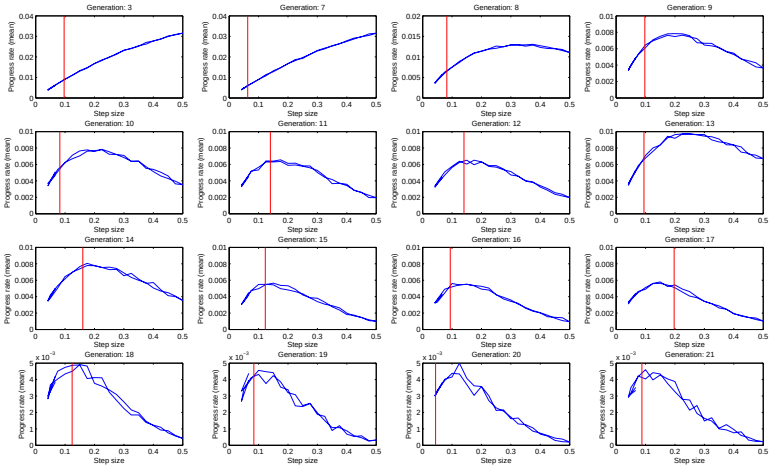


Figure 3.7: Comparison between the peak of the graph of $\hat{\phi}(s_i, x^{(t)})$ with the stepsize $s^{(t)}$ (vertical line) of discrete variables used by the MIES at that stage of the evolution. The step-size $s^{(t)}$ is found by the self-adaptation within the $(4, 28)$-MIES (i.e. without knowledge of $\hat{\phi}$). The results were also computed for probabilities that leave the feasible interval $[1/(3n_d), 0.5]$ and for which the interval transformation $T_{[a,b]}$ was applied.

Below we show that $\Delta_\infty > 0$ leads to a contradiction with proposition 3.15 and thus $\Delta_\infty = 0$ is true. Let f^∞_{best} denote the function value f^t_{best} for $t \to \infty$, the existence of which we have shown above. Then let

$$\epsilon = (f^\infty_{best} - f^*)/2 \tag{3.17}$$

Given the assumption $\Delta_\infty > 0$ and hence $\epsilon > 0$, it follows that

$$\mathbf{X}^*_\epsilon = \{(\mathbf{r}, a^*) \in \mathbb{R}^n \times \mathbb{A} \mid \ | f|_{\mathbb{R}^n \times \{a^*\}}(\mathbf{r}) - f^*| \leq \epsilon\} \tag{3.18}$$

is a nonempty set, where a^* denotes an optimal setting for $a \in \mathbb{A}$. Then $\mathbf{X}^*_\epsilon$ is non-empty because of the assumption of regularity of $f|_{\mathbb{R}^n \times \{a\}}$ for any optimal $a \in \mathbb{A}$.
Thus there exists a closed n-dimensional ball $K = \{\mathbf{r} \in \mathbb{R}^n | \, |\mathbf{r}_0 - \mathbf{r}| \leq \rho\}$ with $\rho > 0$ and center $\mathbf{r}_0 \in \mathbb{R}^n$ that $K_{a^*} = \{(\mathbf{r}, a^*) | \mathbf{r} \in K\} \subseteq \mathbf{X}^*_\epsilon$.

Now, let us compute the probability of the event that the mutation of the discrete and continuous variables yield a point in K_{a^*}, given some arbitrary parent $(\mathbf{r}', a')$, where a' denotes the discrete part of a solution. This can be computed as the joint probability for the following two independent events:

(E1) the mutation of real vector generates $\mathbf{r} \in K$

(E2) the mutation of discrete part of the solution generates a^*

This joint probability is lower bounded by:

$$p_\epsilon = \min_{a' \in \mathbb{A}} p_{a' \to a^*} \min_{\mathbf{r}_0 \in \mathbb{R}^n} \left(\frac{1}{\sqrt{2\pi}\sigma_{min}^2} \right)^n \cdot \int_{\mathbf{r} \in K} \exp\left(\frac{1}{2\sigma_{min}^2} (\mathbf{r} - \mathbf{r}_0)^T \cdot (\mathbf{r} - \mathbf{r}_0) \right) d\mathbf{r} > 0 \tag{3.19}$$

for a step-size $\sigma_{min} > 0$. Here $p_{a' \to a^*}$ is the probability to obtain a^* by one mutation of discrete parameters, which is larger than 0. Now we can derive a lower bound for the probability that K_{a^*} is hit at least once after q generations as (where $(q-1)\lambda \leq t < q\lambda$):

$$\Pr(\bigvee_{i=1}^{q} (\mathbf{x}_{best}^i \in K_{a^*})) = 1 - (1 - (p_\epsilon)^\lambda)^q \tag{3.20}$$

where λ denotes the number of offspring per generation. Hence,

$$\lim_{q \to \infty} \Pr(\bigvee_{i=1}^{q} (\mathbf{x}_{bext}^i \in \mathbf{X}_\epsilon^*)) = 1 \tag{3.21}$$

With expression 3.15 and expression 3.17 we get an contradiction to our assumption that $\Delta_\infty > 0$. In other words, any vector with a distance $\Delta_t > 0$ will be improved as $t \to \infty$ with probability one.

3.4 Summary

Targeting at solving challenging mixed-integer parameter optimization problems in the real world, we proposed a promising algorithm - the so-called Mixed-Integer Evolution Strategies (MIES) - in this chapter. MIES are derived from the canonical Evolution Strategies (ES), which are often applied to optimization problems in continuous search space. MIES, by contrast, use specific variation operators to deal with different parameter types (continuous, integer and discrete) of decision variables. In particular, MIES are capable of tackling *black-box* mixed-integer optimization problems in practice.

Inspired by the previous works [7, 38] on mixed-integer parameter optimization and their applications to some representative real-world applications, we explained the design philosophy of the framework of MIES explicitly. Furthermore, in this chapter we made some theoretical studies on MIES regarding, for instance, the global convergency property and self-adaptation of stepsize. In the rest of this thesis, we will do more experimental studies on MIES to learn more about such an algorithm. For instance, MIES will be applied to feature detection in medical images, and several advanced techniques will be studied for further improving the algorithm performance.

Chapter 4

Synthetic Mixed-Integer Landscapes

In the previous chapter 3, we introduced Mixed-Integer Evolution Strategies (MIES) and related theoretical study results. In this chapter we will present some artificial test problems (fitness landscapes), which are specially designed for mixed-integer parameters search spaces. Through these proposed test problems, we can gain deep insights about MIES algorithm. Some selected empirical results will be presented which demonstrate the algorithm performance, such as its convergence behavior. These synthetic mixed-integer landscapes also provide readers with the opportunity to compare results of this kind of evolutionary algorithm with that of other optimization algorithms, for instance with traditional Evolution Strategies (ES).

The whole chapter is organized as follows: First, in section 4.1, fitness landscapes, which have been proved to be one of the most important concepts in evolutionary theory, will be reviewed briefly especially in the computer science research domain. In section 4.2, we introduce the Barrier function and show some experimental results about it. Next, Mixed-Integer NK Landscapes (MINKL) are explained in detail in section 4.3, as well as some important theorems on the existence and position of local/global optima and some implementation details of the model.

4.1 Fitness Landscapes

Fitness landscapes are very often encountered in the community of people, who are working on evolutionary computation. It is a powerful tool that researchers can use to develop comprehensive insights about the working mechanism of a complex searching process, for instance, a searching process when evolution strategies are applied to some real-world application. Because of their importance, we would

like to give a brief review of fitness landscapes and several important definitions in this part.

4.1.1 Motivation

Fitness landscapes were originally introduced by Sewall Wright in his 1932 paper [124], in which fitness landscapes were used as a way to visualize sophisticated dynamics of population genetics. According to Wright's description, each individual gene combination corresponded to a point on a fitness landscape and there was one axis which represented every possible gene combination. Under certain mathematical conditions, a *potential function* $\mathcal{F}$ can be employed to describe the deterministic dynamics of such kind of evolutionary process. The corresponding definition of a *potential function* $\mathcal{F}$ is defined as follows:

$$\mathcal{F} : \mathcal{S} \rightarrow \mathbb{R}, s \mapsto \mathcal{F}(s) \tag{4.1}$$

where $\mathcal{F}$ is a *potential function* from the state space $\mathcal{S}$ with its neighbourhood structure into the real numbers $\mathbb{R}$ [115]. Each possible state $s \in \mathcal{S}$ can be associated to one number, such that the value of this number reflects the degree to which a certain state is preferable to another state.

Since Wright introduced fitness landscape in his work, this metaphor has been widely adopted by scientists from different research areas, such as biology, chemistry, physics and computer science. The interpretation of fitness can be different when referring to different application areas. In biology, increasing fitness means that a population moves uphill on a fitness landscape. On the contrary, lower points represent low energy states and thus are more desirable in physics. In computer science, as Jones has clearly stated in his PhD thesis [57], a fitness landscape is an artifact of the neighbourhood structure, which is induced by the operators (e.g. mutation operators in evolutionary algorithms) the algorithm employs. In practice, the difference between *maximization* and *minimization* is trivial and they are equivalent apart from an inversion of sign of $\mathcal{F}$.

4.1.2 Local Optima

Given a fitness landscape and its *potential function* $\mathcal{F}$, in the case of minimization, local optima are defined as follows:

Definition 1
A point s in the state space $\mathcal{S}$ is a local optima of the $\mathcal{F}$ if there exists a neighbourhood $\mathcal{N}$ of s such that $\forall s' \in \mathcal{N}, \mathcal{F}(s) - \mathcal{F}(s') \leq 0$.

The number of local optima is one important characteristic of a fitness landscape, and it gives an impression of how rugged a landscape is. In general, a landscape with fewer local optima result in a larger correlation and thus is easier to be tackled by optimization algorithms. By contrast, more local optima means that the corresponding landscape is more rugged and therefore more challenging for

algorithms to deal with. Figure 4.1 shows an example fitness landscape in 2D. According to the definition of local optima, the fitness values of points A, B, C and D are better (smaller) than all their neighbours and are local optima in the case of minimization.

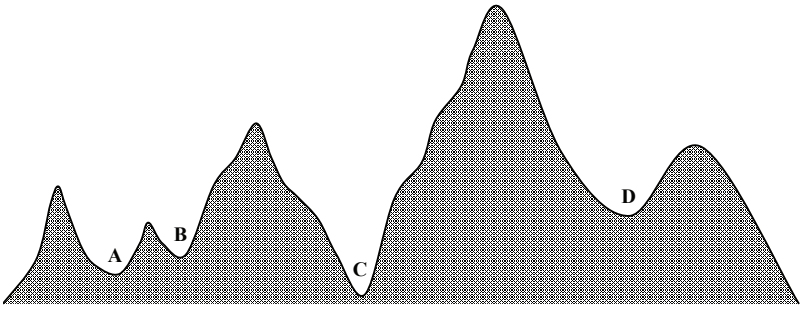


Figure 4.1: Illustration of local optima in a 2D fitness landscape.

4.1.3 Unimodality vs. Multimodality

A landscape is said to be unimodal if it only has one global optimum, that is, has one peak (maximum) or valley (minimum) in a given interval. Otherwise, it is called multimodal landscape if it has several local optima, such as the landscape in Figure 4.1. From mathematical perspective, unimodal functions can be defined as follows [93]:

Definition 2
A function $\mathcal{F}$ is unimodal if (1) $x_1 < x_2 < x^*$ implies that $\mathcal{F}(x_1) < \mathcal{F}(x_2)$, and (2) $x_2 > x_1 > x^*$ implies that $\mathcal{F}(x_2) > \mathcal{F}(x_1)$, where x^* is the minimum point.

Generally speaking, a multimodal landscape is more difficult compared with a unimodal landscape. However, in some extreme cases, a unimodal landscape can also present difficulties for searching algorithms.

4.2 Barrier Function

Barrier function[1] is a multi-modal problem generator that produces integer optimization problems with a scalable degree of ruggedness (determined by parameter C) by generating an integer array `A` using Algorithm 8.

For $C = 0$ the ordering of the variable $y \in [0, 19]$ values corresponds to the ordering of values of $A(y)$. If the value of C is slightly increased, still part of the

[1] In Dutch, it is called "Drempels" function

Algorithm 8 Barrier Function.

1: $\mathtt{A}[i] = i, i = 0, \ldots, 19$
2: **for** $k \in \{1, \ldots, C\}$ **do**
3: $j \leftarrow$ uniform random number out of $\{0, \ldots, 18\}$
4: swap values of $\mathtt{A}[j]$ and $\mathtt{A}[j+1]$
5: **end for**

order will be preserved under the mapping A, and thus similarity information can be exploited. Then a barrier function is computed:

$$f_{barrier}(\mathbf{r}, \mathbf{z}, \mathbf{d}) = \sum_{i=1}^{n_r} \mathtt{A}[\lfloor r_i \rfloor]^2 + \sum_{i=1}^{n_z} \mathtt{A}[z_i]^2 + \sum_{i=1}^{n_d} \mathtt{B_i}[d_i]^2 \rightarrow \min$$

$$n_r = n_z = n_d = 5, \mathbf{r} \in [0, 19]^{n_r} \subset \mathbb{R}^{n_r},$$
$$\mathbf{z} \in [0, 19]^{n_z} \subset \mathbb{Z}^{n_z}, \mathbf{d} \in \{0, \ldots, 19\}^{n_d} \subset \mathbb{D}^{n_d}.$$

Here, $B_i (i = 1, \ldots, n_d)$ denotes a set of i permutations of the sequence $0, \ldots, 19$, each of which is a random permutation fixed before the run. This construction prevents that the value of the nominal value d_i is quantitatively (anti-)correlated with the value of the objective function f. Such a correlation would contradict with the assumption that d_i are nominal values. Whenever a correlation between neighboring values can be assumed it is wiser to assign them to the ordinal type and treat them accordingly.

The parameter C controls the ruggedness of the resulting function with regard to the integer space. Higher values of C result in more rugged landscapes with many barriers. To get an intuition about the influence of C on the geometry of the function we included plots for a two-variable instantiation of the barrier function in Figure 4.2 for $C = 20$, 100, 500, and 1000. Intuitively, barrier functions with a higher control parameter C may have many local optima and a search procedure can easily get trapped by them. As we can see from these plots, when the control parameter C increases the landscape becomes more rugged. For instance, the landscape of $C = 1000$ shows much more barriers compared to $C = 20$. We also noticed that above certain C value (threshold), the landscape difficulty will not change too much as C increases.

4.2.1 Experimental Results

Suggested by our former studies [38, 75, 77], the following MIES and ES settings are chosen for the experiments on the barrier problems: $(\mu = 4, \lambda = 28)$ for the population and offspring sizes, and $(n_\sigma = n_\varsigma = n_p = 1)$ for the step-size mode[2]. Since Evolution Strategies are stochastic algorithms, in the empirical experiments

[2] In contrast, $(n_\sigma = n_r, n_\varsigma = n_z, n_p = n_d)$ represents n step-size mode.

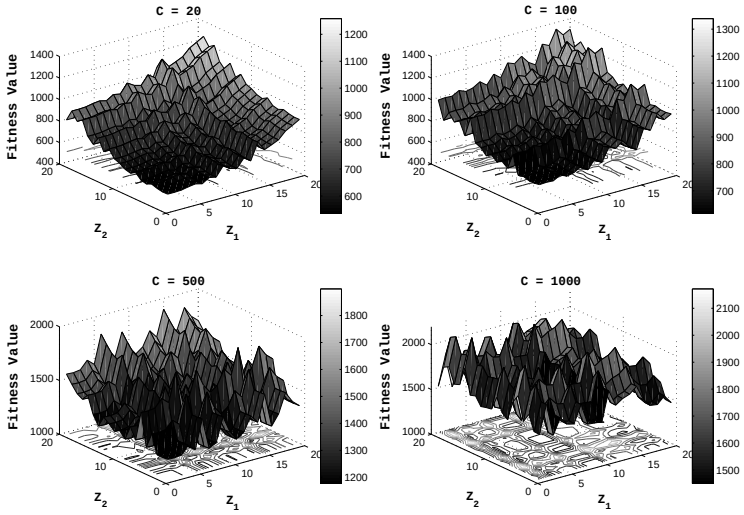


Figure 4.2: Surface plots of the barrier test functions for two integer variables Z_1 and Z_2, the control parameter $C = 20$, 100, 500 and 1000. All other variables were kept constant at a value of zero, Z_1 and Z_2 values were varied in the range from 0 to 19.

we create 10 instantiations[3] for each control parameter C, and for each of them we let the algorithm perform 20 repeated runs (there are in total $20 \times 10 = 200$ runs for each value of C).

Figure 4.3 shows average best fitness values found by one step-size $(4, 28)$ MIES and one step-size $(4, 28)$ ES[4] on barrier functions with different control parameters C. As we can see that it is more difficult for both MIES and ES to find the global optimum on barrier functions with a higher C value. This observation supports our finding from Figure 4.2: the landscape with a larger C value is more rugged and it is more challenging to tackle.

Based on our algorithm design in chapter 3, MIES is supposed to be more efficient for exploring the mixed integer landscapes compared to a standard ES. To check this assumption, we plot average best fitness values found by both MIES and ES with $C = 20, 100, 300, 500,$ and 1000 in Figure 4.4. The corresponding box plot for best fitness values found by both MIES and ES in the last *generation*$(= 100)$ is shown in Figure 4.5. When $C = 20$ or 100 ES performs a little bit better than MIES. This can be explained that constructed landscapes with $C = 20$ or 100 are still simple, and this gives the chance to standard ES algorithms to fully explore the searching space. However in the case of higher C values, MIES show

[3] By using different random seeds.

[4] Here, all parameters are evolved as continuous variables.

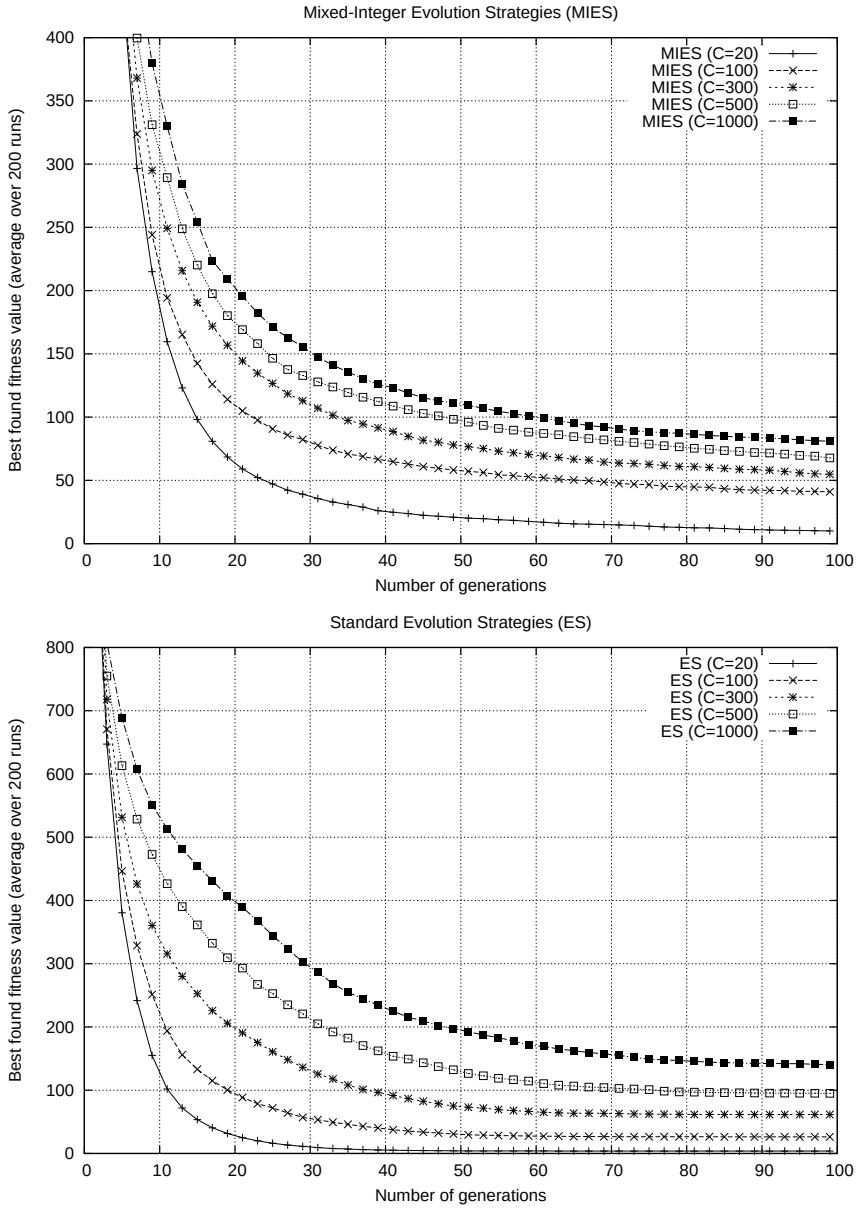


Figure 4.3: The average best results for $(4, 28)$ Mixed-Integer Evolution Strategies (Top) and standard Evolution Strategies (Bottom) using a single step-size on barrier functions with control parameter $C = 20$, 100, 300, 500, and 1000.

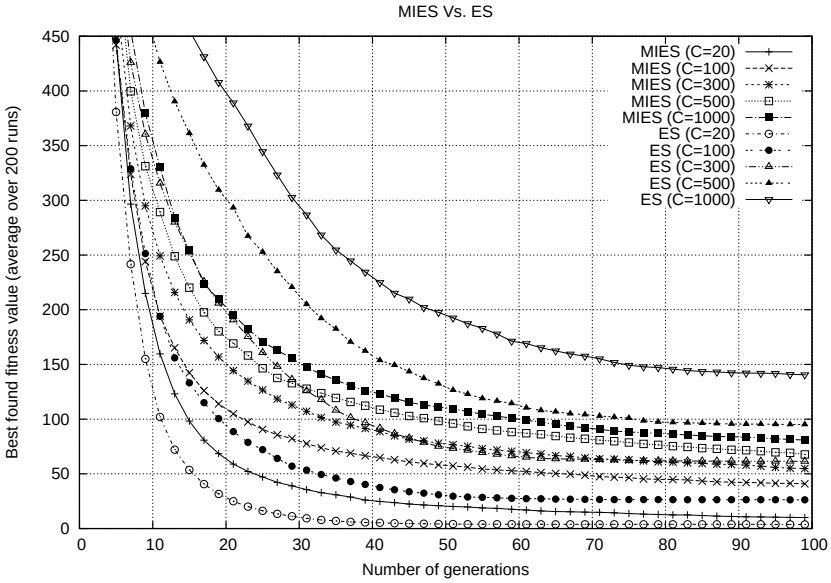


Figure 4.4: Comparison between average best fitness values found by $(4, 28)$ MIES and $(4, 28)$ ES for $C = 20, 100, 300, 500$, and 1000.

the advantage over standard ES. For $C = 300$ or 500 the overall performance MIES are already competitive to standard ES. For $C = 1000$ average best fitness values obtained by MIES are much lower than standard ES.

4.3 Mixed-Integer NK Landscapes

NK landscapes (NKL, also referred to as NK *fitness* landscapes), introduced by Stuart Kauffman [61], were devised to explore the way that epistasis controls the "*ruggedness*" of an adaptive landscape.

Frequently, NKL are used as test problem generators for Genetic Algorithms. NKL have two advantages. First, the ruggedness and the degree of interaction between variables of NKL can be easily controlled by two tunable parameters: the number of genes N and the number of epistatic links of each gene to other genes K. Second, for given values of N and K, a large number of NK landscapes can be created at random. A disadvantage is that the optimum of a NKL instance can generally not be computed, except through complete enumeration.

As NKL have not yet been generalized for continuous, nominal discrete, and mixed-integer decision spaces, they cannot be employed as test functions for a large number of practically important problem domains.

To overcome this shortcoming, we introduce an extension of the NKL model,

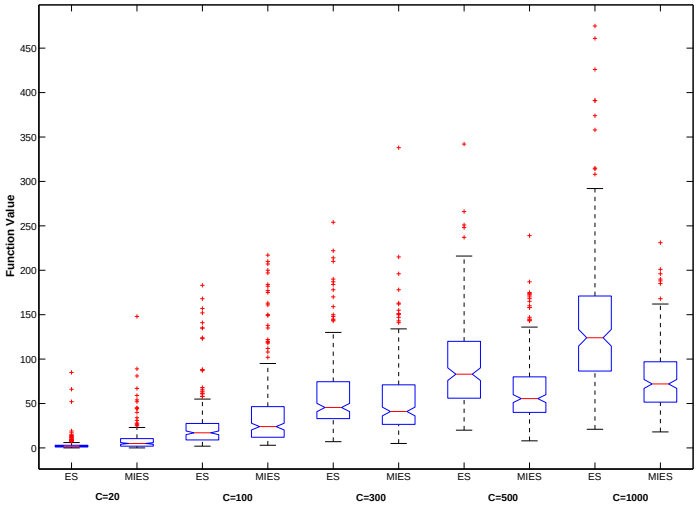


Figure 4.5: Notched box plot for fitness value (*generation* = 100) of barrier functions with different control parameter $C = 20$, 100, 300, 500, and 1000 by using standard ES and MIES.

mixed-integer NKL (MINKL), that capture these problem domains. They extend traditional NKL from the binary case to a more general situation, by taking different parameter types (continuous, integer, and nominal discrete) and interactions between them into account (cf. Figure 4.6).

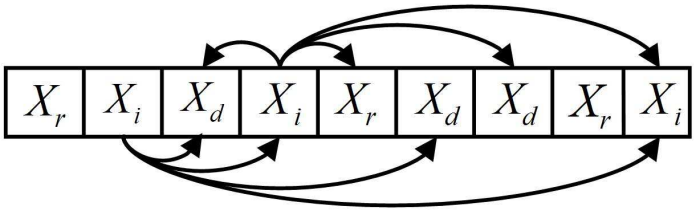


Figure 4.6: Example Genes and their interactions

4.3.1 NK Landscapes

Kauffman's NK Landscapes model defines a family of pseudo-boolean fitness functions $F : \{0,1\}^N \to \mathbb{R}^+$ that are generated by a stochastic algorithm.

It has two basic components: A structure for gene interaction (using an *epis-*

tasis matrix E), and a way this structure is used to generate a fitness function for all the possible genotypes [1].

The gene interaction structure is created as follows: The genotype's fitness is the average of N fitness components F_i, $i = 1, \ldots, N$. Each gene's fitness component F_i is determined by its own allele x_i, and also by K alleles at K ($0 \leq K \leq N-1$) epistatic genes distinct from i. The fitness function reads:

$$F(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^{N} F_i(x_i; x_{i_1}, \ldots, x_{i_k}), \; x \in \{0,1\}^N \tag{4.2}$$

where $\{i_1, \ldots, i_k\} \subset \{1, \ldots, N\} - \{i\}$. There are two ways for choosing K other genes: '*adjacent neighborhoods*', where the K genes nearest to position i on the vector are chosen; and '*random neighborhoods*', where these positions are chosen randomly on the vector. In this paper we focus on the latter case, '*random neighborhoods*'. However, a translation to the first case is straightforward.

The computation of $F_i : \{0,1\}^K \rightarrow [0,1)$, $i = 1, \ldots, N$ is based on a *fitness matrix* F. For any i and for each of the 2^{K+1} bit combinations a random number is drawn independently from a uniform distribution over $[0,1)$. Accordingly, for the generation of one (binary) NK landscape the setup algorithm has to generate $2^{K+1}N$ independent random numbers. The setup algorithm also creates an epistasis matrix E which for each gene i contains references to its K epistatic genes. Table 4.1 illustrates the *fitness matrix* and *epistasis matrix* of a NKL. A more detailed description of its implementation can be found in [34].

$E_1[1]$	$E_1[2]$	$\cdots$	$\cdots$	$\cdots$	$E_1[K]$
$E_2[1]$	$E_2[2]$	$\cdots$	$\cdots$	$\cdots$	$E_1[K]$
$\cdots$	$\cdots$	$\cdots$	$E_i[j]$	$\cdots$	$\cdots$
$E_N[1]$	$E_N[2]$	$\cdots$	$\cdots$	$\cdots$	$E_N[K]$

$F_1[0]$	$F_1[1]$	$\cdots$	$\cdots$	$\cdots$	$F_1[2^{K+1}-1]$
$F_2[0]$	$F_2[1]$	$\cdots$	$\cdots$	$\cdots$	$F_2[2^{K+1}-1]$
$\cdots$	$\cdots$	$\cdots$	$F_i[j]$	$\cdots$	$\cdots$
$F_N[0]$	$F_N[1]$	$\cdots$	$\cdots$	$\cdots$	$F_N[2^{K+1}-1]$

Table 4.1: Epistasis matrix E (left) and fitness matrix F (right)

After having generated the epistasis and fitness matrices, for any input vector $\mathbf{x} \in \{0,1\}^N$ we can compute the fitness in $\mathcal{O}(KN)$ computational complexity via:

$$F(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^{N} F_i[2^0 x_i + 2^1 x_{E_i[1]} + \cdots + 2^K x_{E_i[K]}] \tag{4.3}$$

Note, that the generation of F has an exponential computational complexity and space complexity in K, while being linear in N. The computational complexity for computing function values is linear in K and N for this implementation.

Properties of NK Landscapes

Kauffman's model makes two principal assumptions: first, that the fitness of a genotype is the sum of the contributions from each gene, and second, that the

effects of polygeny and pleiotropy make these interactions effectively random. Besides Kaufmann, some other researchers, e. g. Weinberger et al. [121, 113], did an extensive study on NKL. Some well-known properties are:

1. $K = 0$ (no epistasis): The problem is separable and there exists a unique global optimum. Assuming a Hamming neighborhood-structure, the problem gets unimodal.

2. $1 \leq K < N - 1$: For $K = 1$, a global optimum can still be found in polynomial time [121]. For $K \geq 2$, global optimization is NP-complete for the random assignment of neighbors and constant K. However, the problem can always be solved in a computational complexity of 2^N function evaluations and hence can practically be solved for problems of moderate dimension (N around 30). For adjacent neighbors, the problem can be solved in time $O(2^K N)$ (cf. Weinberger [121]).

3. $K = N - 1$: This corresponds to the maximum number of interactions between genes. Practically speaking, to each bitstring of $F : \{0,1\}^N \rightarrow [0,1)$ we assign a sum of N values, each of which is drawn independently from a uniform distribution in $[0,1)$. If we choose the Hamming neighborhood on $\{0,1\}^N$ the following results apply:

   - The probability that a random bit string is a local optimum is $\frac{1}{N+1}$
   - The expected number of local optima is $\frac{2^N}{N+1}$

4.3.2 Generalized NK Landscapes

As mentioned in the previous section, Kauffman's NKL model is a stochastic method for generating fitness functions on binary strings. In order to use it as a test model for mixed-integer evolution strategies, we extend it to a more general case such that the fitness value can be computed for different parameter types. Here we consider continuous variables in $\mathbb{R}$, integer variables in $[z_{min}, z_{max}] \subset \mathbb{Z}$, and nominal discrete values from a finite set of L values. In contrast to the ordinal domain (continuous and integer variables), for the nominal domain no natural order is given. Mixed-integer optimization problems arise frequently in practice, e.g. when optimizing optical filter designs [7] and the parameters of algorithms [75].

The idea about how to extend NKL to the mixed-integer situation will be described in three steps. First we propose a model for continuous variables, then for those with integer variables and nominal discrete variables. Finally, we will discuss the case of NKL that consists of all these different variable types at the same time and allow for interaction among variables of different types. This defines the full mixed-integer NKL model.

Continuous NK Landscapes

In order to define continuous landscapes, we choose an extension of binary NKL to an N-dimensional hypercube $[0,1]^N$. Therefore, all continuous variables are normalized between $[0,1]$. In the following we describe the construction of the objective function $F : [0,1]^N \rightarrow [0,1)$:

Whenever the continuous variable takes values at the corners of the hypercube, the value of the corresponding binary NKL is returned. For values located in the interior of the hypercube or its delimiting hyperplanes, we employ a *multi-linear interpolation technique* that achieves a continuous interpolation between the function values at the corner. Note that a higher order approach is also possible but we chose a multi-linear approach for simplicity and ease of programming. Moreover, the theory of multi-linear models as used in the design and analysis of experiments, introduces intuitive notions for the effect of single variables and interaction between multiple variables of potentially different types [19]. For each of the N fitness components $F_i : [0,1]^{K+1} \rightarrow [0,1)$, we create a multi-linear function

$$F_i(\mathbf{x}) = \sum_{j=0}^{2^{K+1}-1} a_j^i x_i^{[1 \texttt{ AND } j]} \prod_{k=1}^{K} x_{i_k}^{[2^k \texttt{ AND } j]/2^k}, \tag{4.4}$$

where AND is the bitwise *a*nd operator and x_{i_k} is the k-th epistatic gene of x_i.

For instance, in the case $K = 2$ the formula for $F_i(\mathbf{x})$ becomes[5]:
$a_{000}^i + a_{001}^i x_i + a_{010}^i x_{i_1} + a_{100}^i x_{i_2} + a_{011}^i x_i x_{i_1} + a_{101}^i x_i x_{i_2} + a_{110}^i x_{i_1} x_{i_2} + a_{111}^i x_i x_{i_1} x_{i_2}$.

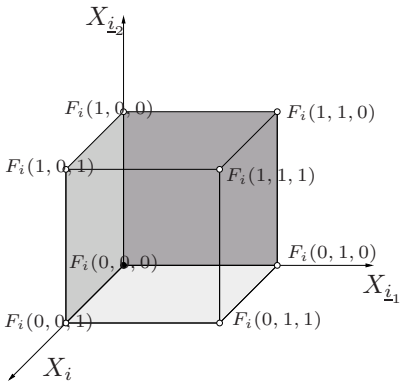
Once uniformly distributed random values have been attached to the corners of the K-dimensional hypercube (cf. Figure 4.7), we can identify the coefficients $a_0^i, \ldots, a_{2^{K+1}-1}^i$ by solving a linear equation system (LES). However, even for moderate K the computational complexity for applying general LES solvers would be prohibitively high. An advantage of the multi-linear form (as compared to other interpolation schemes like radial basis functions or splines) is that it allows for an efficient computation of the coefficients by exploiting the diagonal structure of the equation system. Accordingly, a_j^i can be obtained by means of the following formula:

$$a_0^i = F_i[0], a_j^i = F_i[j] - \sum_{\ell=0}^{j-1} \left[a_\ell^i \mathrm{I}(\ell = (\ell \texttt{ AND } j)) \right], j = 1, \ldots, 2^{K+1} - 1 \tag{4.5}$$

In order to compute the values, we have to start with $j = 0$ and increase the value of j. Hence, the number of additions we need for computing all coefficients is proportional to $(2^{K+1} - 1)(2^{K+1})/2 = 2^{2(K+1)-1} - 2^K$.

Once we have the a_j^i values, we can use Equation 4.2 to compute the model. Of course the domain of the $\mathbf{x}$ values has to be replaced by $[0,1]^N$ in that equation. For the computation of the global optimal value of the continuous NK landscapes the following lemma is useful:

[5] Note that we use binary instead of decimal numbers for the index to make the construction more clear.



$$
\begin{aligned}
a^i_{000} &= F_i(0,0,0)\\
a^i_{001} &= F_i(0,0,1) - a^i_{000}\\
a^i_{010} &= F_i(0,1,0) - a^i_{000}\\
a^i_{011} &= F_i(0,1,1) - a^i_{000} - a^i_{001} - a^i_{010}\\
a^i_{100} &= F_i(1,0,0) - a^i_{000}\\
&\vdots\\
a^i_{111} &= F_i(1,1,1) - a^i_{000} - a^i_{001} - a^i_{010}\\
&\quad -a^i_{011} - a^i_{100} - a^i_{101} - a^i_{110}
\end{aligned}
$$

Figure 4.7: Example HyperCube with $K = 2$ and the computation of a^i_j

Lemma 3
At least one global optimum of the function F will always be located in one of the corners of the N dimensional hypercube, such that the computation of the optimal function value upper bounds the computational complexity for the binary model.

Proof
The idea of the proof is that there is an algorithm that for any given input $\mathbf{x}^* \in [0,1]^N$ determines a corner of the hypercube, the function value of which is not higher than the function value at F, given that F has a multilinear form. Basically, the proposed algorithm can be described as a path oriented algorithm that searches parallel to the coordinate axis: First we fix all variables except one, say x_1, in F. It is now crucial to see that the remaining form $F(x_1, x^*_2, \ldots, x^*_N)$ is a linear function of x_1. Now, because the form is linear, it is obvious to see that either $(1, x^*_2, \ldots, x^*_N)^T$ or $(0, x^*_2, \ldots x^*_N)^T$ has a function value that is better or equal than the function value at $(x^*_1, \ldots, x^*_N)^T$. We fix x_1 to a value for which this is the case, i. e. we move either to $(1, x^*_2, \ldots, x^*_N)^T$ or to $(0, x^*_2, \ldots x^*_N)^T$ without increasing the function value. For the new position $\mathbf{x}^{1*}$ we again fix all variables except one. This time x_2 is the free variable. Again we can move the value of x_2 either to zero or to one, such that the function value does not increase. Now, the new vector $\mathbf{x}^{12*}$ will either be $(x^{1*}_1, 0, x^*_3, \ldots, x^*_N)^T$ or $(x^{1*}_1, 1, x^*_3, \ldots, x^*_N)^T$. After continuing this process for all remaining variables x_3 to x_N we finally obtain a vector $\mathbf{x}^{12\cdots N*}$, all values of which are either zero or one, and the function value is not worse than that of $\mathbf{x}^*$.

Theorem 4
The problem of finding the global optimal value for a continuous NKL is NP-complete for $K \geq 2$.

Proof
Finding the optimum in the corner is equivalent to the NP-complete binary case. By applying Lemma 1, we can reduce the continuous case to the binary case. On the other hand, whenever we find the global optimal solution for the continuous case, in polynomial time we can construct a good solution that is just as good where all optima are located at the corners in linear time. Thus, there exists a polynomial reduction of the binary case to the continuous case.

Integer NK Landscapes

Based on our design, NKL on integer variables can be considered to be a special case of continuous NKL. The integer variables can be normalized as follows: Let $z_{min} \in \mathbb{Z}$ denote the lower bound for an integer variable, and $z_{max} \in \mathbb{Z}$ denote its upper bound. Then, for any $z \in [z_{min}, z_{max}] \subset \mathbb{Z}$ we can compute the value of $x = (z - z_{min})/(z_{max} - z_{min})$ in order to get the corresponding continuous parameter in $[0, 1]$, which can then be used in the continuous version of F to compute the NKL. Note that the properties discussed in 4.3.2 and 4.3.2 also hold for integer NKL.

Nominal NK Landscapes

To introduce nominal discrete variables in an appropriate manner a more radical change to the NKL model is needed. In this case it is not feasible to use interpolation, as this would imply some inherent neighborhood defined on a single variable's domain $x_i \in \{d_1^i, \ldots, d_L^i\}$, $i = 1, \ldots, N$, which, by definition, is not given for the nominal discrete case. We will now propose an extension of NKL that takes into account the special characteristics of nominal discrete variables.

Let the domain of each nominal discrete variable x_i, $i = 1, \ldots, N$ be defined as a finite set of maximal size $L \geq 2$. Then for the definition of a function on a tuple of $K + 1$ such values we would need a table with L^{K+1} entries. Again, we can assign all fitness values randomly by independently drawing values from a uniform distribution. The size of the sample is upper-bounded by L^{K+1}. For $L = 2$ this corresponds to the binary case. After defining N fitness components F_i, we can then sum up the values of these components for the NKL model (Eq. 4.2). The optimum can be found by enumerating all input values, the computational complexity of which is now L^N. The implementation of the function table and the evaluation procedures are similar to that of the binary case. Note that for a constant value of L and K the space needed for storing the function values is given by NL^{K+1}, so is the computational complexity for generating the matrix. The time for the function evaluations is proportional to $N(K + 1)$.

Equipping the discrete search space with a Hamming neighborhood, in case $K = 0$ the problem remains unimodal. For $K > 0$, we remark that for the general problem with $L > 2$, the detection of the optimum is more difficult than in the binary case. Hence, the binary case can be reduced to the case $L > 2$, but not vice versa. For the case of full interaction (K=N-1) we show:

Lemma 5
For the nominal discrete NKL with $K = N - 1$, $L \equiv$ *constant*, and Hamming neighborhood defined on the discrete search space, the probability that an arbitrary solution $\mathbf{x}$ gets a local optimum is $\frac{1}{N(L-1)+1}$. Moreover the expected number of local optima is $\frac{L^N}{N(L-1)+1}$.

Proof
Given the preliminaries, $N(L-1)$ is the number of Hamming neighbors for any solution $\mathbf{x} \in \{1, \ldots, L\}^N$. Since we assign a different fitness value from the interval $[0, 1)$ independently to each neighbor, the probability that the central solution, i.e. $\mathbf{x}$ itself becomes the best solution, is $1/(N(L-1)+1)$. Since L^N is the number of search points in $\{1, \ldots, L\}^N$ we can compute the expected number of local optima as $\frac{L^N}{N(L-1)+1}$.

Mixed Integer NK Landscapes

It is straightforward to combine these three types of variables into a single NKL with epistatic links between variables of different types (cf. Figure 4.6). For mixed variables of the integer and continuous types there is no problem, since integers, after normalization, are treated like continuous variables in the formula of F. If there are D nominal discrete variables that interact with a continuous variable, then the values of these discrete variables determine the values at the edges of the $K - D$ dimensional hypercube that is used for the interpolation according to the remaining continuous and integer variables. Note that for different nominal discrete values the values at the corners of the $K - D$ dimensional hypercube will change in almost every case.

Instead of describing the mixed variable case in a formal manner we give an illustrating example (cf. figure 4.8). This example shows one individual with three parameters (one continuous, one integer and one discrete), and each gene interacts with both other genes. For each gene, a hypercube is created. We assume there are three levels for the discrete gene X_d ($L = 3$), so the hypercube is reduced to three parallel planes, and the value of the discrete gene decides which plane is chosen. More concretely, assuming the individual has the following values: $X_d = 0, X_i = 0.4, X_r = 0.8$, the value of the discrete parameter X_d determines which square is chosen ($X_d = 0$). The value for each corner is based on the fitness matrix in Table 4.2 (bold displayed). As mentioned in the previous chapter, we calculate the fitness value of this individual as follows:

$$
\begin{gathered}
F_r(\mathbf{a}, \mathbf{x}) = a_0 + a_1 X_r + a_2 X_i + a_3 X_i X_r \\
a_0 = F_r(0,0) = 0.8, \quad a_1 = F_r(0,1) - a_0 = -0.1 \\
a_2 = F_r(1,0) - a_0 = -0.1, \quad a_3 = F_r(1,1) - a_0 - a_1 - a_2 = -0.1 \\
F_r(0.4, 0.8) = 0.648
\end{gathered}
$$

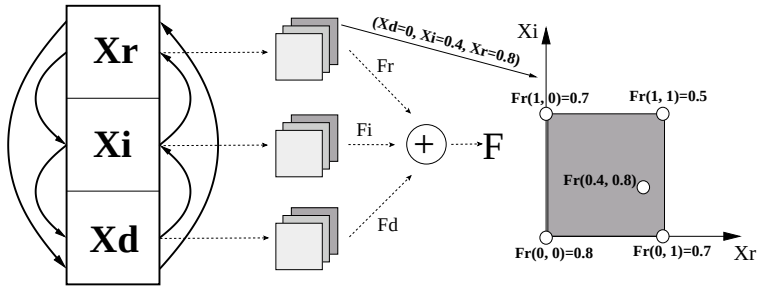


Figure 4.8: Example for the computation of a MI-NK landscape.

$E_r[1] = X_i$	$E_r[2] = X_d$		***0.8***	***0.7***	***0.7***	***0.5***		0.3	0.7	0.2	0.9		0.5	0.6	0.3	0.5
$E_i[1] = X_r$	$E_i[2] = X_d$	F_r	0.5	0.8	0.4	0.7	F_i	0.2	0.3	0.7	0.9	F_d	0.9	0.8	0.2	0.7
$E_d[1] = X_r$	$E_d[2] = X_i$		0.2	0.1	0.8	0.4		0.2	0.5	0.4	0.6		0.8	0.7	0.3	0.3

Table 4.2: Example epistasis matrix (left)and fitness matrix (right).

4.3.3 Experimental Results

In order to test our mixed-integer NKL problem generator, we choose a population size μ of 4, offspring size λ of 28 and comma strategy for our experiments. The maximum number of generations is set to 100. Similar to experiments for barrier functions, to evaluate the algorithm performance, we generated 10 problem instantiations for each $K \in \{1, 3, 5, 10, 14\}$ so that it is still feasible to find the global optimum by evaluating all bit strings of length $N = 15$. Each generated problem consists of 5 continuous ($N_r = 5$), 5 integer ($N_z = 5$) and 5 nominal discrete ($N_d = 5$) variables. The continuous variables are in the range [-10,10], the integer-valued variables are also in the range [-10,10] and we used $\{0, 1\}$ for the nominal discrete variables (Booleans). We ran both MIES and ES 20 times on each problem instance. To compare the results of the different experiments we define the following error-measure:

$$error = best\ found\ fitness - best\ possible\ fitness$$

The results are displayed in Figure 4.9. The x-axis shows the number of generations while the y-axis shows the average *error* (over all experiments). As can be seen, for both algorithms an increase in K results in an increase in *error* which indicates the problem difficulty increases with K.

Like we did in experiments on barrier functions, we plot average errors of different K for both MIES and ES in Figure 4.10. In addition, we create a box plot for the last generation's errors of both MIES and ES in Figure 4.11.

There we can compare overall performance between standard ES and MIES algorithms. As can be seen, in the case of $K = 1, 3, 5, 10$ the MIES show better results than the standard ES. For $K = 14$ the average errors for MIES and

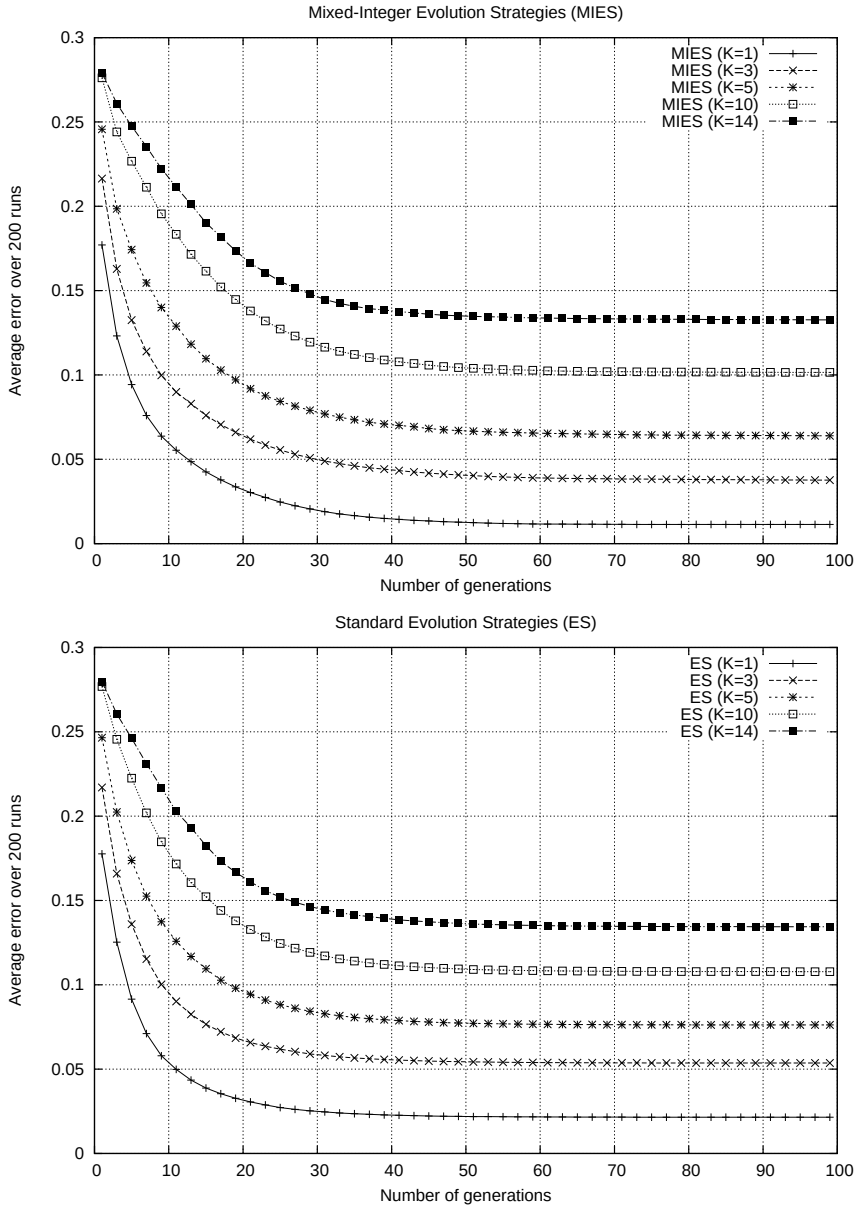


Figure 4.9: The error averaged over 10 mixed-integer NK landscape problems with $N = 15$ by using MIES (Top) and ES (Bottom). Each problem contained 5 continuous, 5 integer-valued, and 5 Boolean-valued variables.

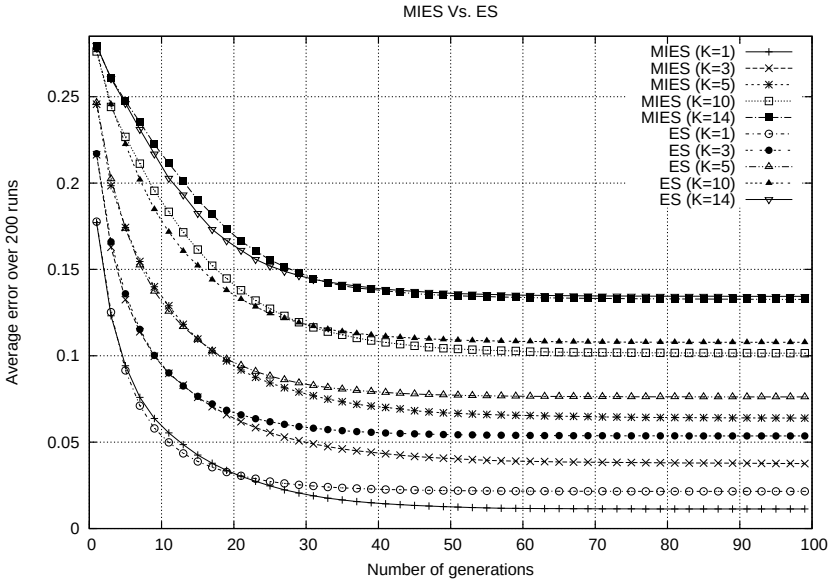


Figure 4.10: The error averaged over 10 mixed-integer NK landscape problems with $N = 15$ by using standard ES and MIES. Each problem contained 5 continuous, 5 integer-valued and 5 Boolean-valued variables.

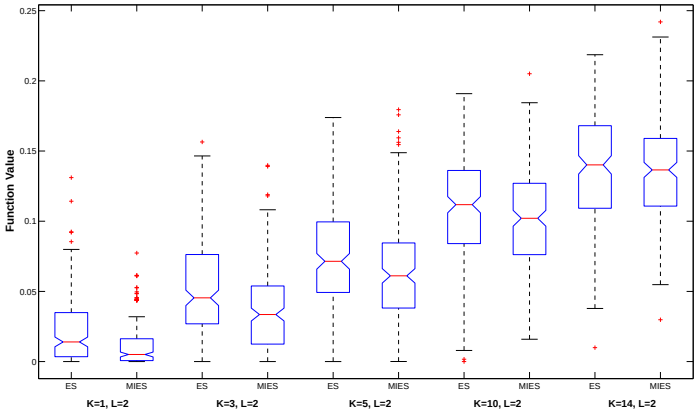


Figure 4.11: Notched box plot for fitness value (*generation* = 100) of mixed-integer NK landscape problems with $N = 15$ by using standard ES and MIES. Each problem contained 5 continuous, 5 integer-valued and 5 Boolean-valued variables.

ES are similar. As $K = 14$ means that each variable is connected to all other variables, the search space becomes extremely complex and it becomes too hard for both algorithms. Under such circumstances, MIES and ES show almost the same performance.

4.4 Summary

We presented two artificial landscapes in this chapter: Barrier function and Mixed-Integer NK Landscapes (MINKL) and they are intensively used as test cases in this work. By design, these artificial landscapes are very good for helping us to learn more about performance behavior of MIES algorithm, such as convergence property. In return, we can further improve the MIES and apply it to more complex real-world applications. Moreover, we compared MIES to the standard (continuous) ES using simple truncation of continuous variables. It turns out that the MIES approach has a much higher convergence reliability.

About MINKL, we make some remarks here: MINKL extends NK landscape model from discrete problem (binary case in general) domain to the mixed-integer problem domain. It turns out that a multi-linear interpolation approach for the continuous and integer variables provides a straightforward generalization of this model and can be easily implemented. Using Equation 4.4, function values can be computed in linear time. However, the detection of the global optimum turns out to be a NP-complete problem for $K > 2$ and can be reduced to the problem of detecting the global optimum for the binary case. However, an alleged drawback of the interpolation approach is that its optima are always located in the corners of the search space. There are possible some ways to address this problem. One way would be to transform the input variables by means of a periodic function and then map them back to $[0, 1]$, e.g. by substituting x_i by $s(x_i) = \frac{1}{2} + \frac{1}{2}\cos(\pi x_i + \pi)$ and restricting x_i to the interval $[-0.5, 1.5]$ for $i = 1, \ldots, N$. It is easy to show that the optima for this transformed function are in the same position as for the original model. For the nominal discrete variables the binary NK landscape was extended to a L-ary representation. In this case, the amount of random numbers increases exponentially with L. Also, for $N = K - 1$ it has been shown that the number of local optima increases exponentially with L.

Part II

Application to Medical Image Analysis

Do the difficult things while they are easy and do the great things while they are small. The journey of a thousand miles begins beneath one's feet.

Lao-tzu

Chapter 5

Parameter Optimization for Medical Image Analysis

In the previous chapters, we introduced mixed-integer parameter optimization with two representative real-world applications in industry - optimization of multilayer optical coatings and optimization of chemical engineering plants. We also presented some theoretical and experimental studies on our proposed Mixed-Integer Evolution Strategies (MIES), which show that MIES is a promising method to tackle mixed-integer parameter optimization especially in *black-box* scenarios. In this chapter, we will show another challenging optimization task which comes from the medical field and explain why and how MIES can be applied to the optimization of control parameters of a semi-automatic image analysis system for Intravascular Ultrasound (IVUS) images.

IVUS is a technique used to get real-time high resolution tomographic images from the inside of coronary vessels and other arteries. The IVUS image feature detectors used in the analysis system are expert-designed and the default parameters are calibrated manually so-far. The new approach, based on MIES, can automatically find good parameterizations for sets of images which achieve better result than with manually tuned parameters. From the algorithmic point of view the difficulty is to design a *black-box* optimization strategy that can deal with nonlinear functions and different types of parameters, including integer, nominal discrete and continuous variables. Compared with canonical Evolution Strategies (ES), which are often applied to optimization problems in continuous search spaces, the MIES turns out to be well suited for this task. The results presented in this chapter will summarize and extend recent studies on benchmark functions and on the IVUS image analysis optimization problem.

5.1 Introduction

Feature detection in medical images is a key task in the medical field. Often complex and variable structures, such as calcified plaque in arteries, are to be detected and modelled in images or sequences of images. The development of feature detection systems has received much attention in medical and computer science research. However, the performance of most systems depend on a large number of control parameters, and the setting of these control parameters is usually done by means of an educated guess or manual tuning using trial and error.

In this work we argue that manual tuning is often not sufficient to exploit the full potential of image detection systems, i.e. it leads to suboptimal parameter settings. We propose a versatile and robust procedure for automated parameter tuning based on evolutionary algorithms (EAs) such as MIES. Compared to the manual trial and error approach, with MIES the systems developer can search for optimized parameter settings automatically and will likely obtain parameter settings that lead to significantly higher accuracy of the feature detectors.

Among other image acquisition techniques, IVUS received major attention for analyzing the structure of coronary blood vessels. Due to noise, pullback movements of the catheter, and the variability of structures even for human experts it can be difficult to interpret IVUS image sequences. Therefore, the development of tailored computer assisted image analysis has received major attention in recent years [16, 89, 103].

However, today's methods, directed at the automated recognition of certain structures in images, are applicable only over a limited range of standard situations. To overcome this problem an image interpretation system, based on the paradigm of multi-agents [16, 17], using the cognitive architecture Soar [87], was successfully developed over the past years. Agents in this system dynamically adapt their segmentation algorithms. This adaptation is based on knowledge about global constraints, contextual knowledge, local image information and personal beliefs like confidence in their own image processing results.

Although in practice the multi-agent system has been shown to offer lumen and vessel detection with precision comparable to human experts [16], it is designed for symbolic reasoning, not numerical optimization. Besides, it is almost impossible for a human expert to completely specify how an agent should adjust its feature detection parameters in each and every possible interpretation context. As a result an agent has only control knowledge for a limited number of contexts and a limited set of feature detector parameters. This knowledge has to be updated whenever something changes in the image acquisition pipeline. Therefore, it would be much better if such knowledge could be acquired by learning the optimal parameters for different interpretation contexts automatically.

This chapter addresses the problem of learning these optimal parameter settings from a set of example segmentations. It is an optimization problem that is difficult to solve in practice with standard numerical methods (like gradient-based

strategies), as it incorporates different types of parameters, and confronts the algorithms with a complex geometry (rugged surfaces, discontinuities). Moreover, the high dimensionality of this problem makes it almost impossible to find optimal settings through manual experimentation.

Encouraged by previous work [8, 38, 24] on optimization of image segmentation algorithms in the medical domain and other application fields we consider MIES as a solution method. Unlike these previous approaches, MIES are more suitable for dealing with continuous parameters and can handle difficult mixed-integer parameter optimization problems as encountered in the image processing domain.

5.2 Intravascular Ultrasound Image Analysis

Cardiovascular disease is the leading cause of death in the USA and coronary artery disease has the highest percentage (53%) of death among the heart diseases according to the American Heart Association Heart Disease and Stroke Statistics [98]. Atherosclerosis is a disease characterized by a deposit of plaque in an arterial wall over time. The disruption of an atherosclerotic plaque is considered to be the most frequent cause of heart attack and sudden cardiac death. Studying vulnerable plaques constitutes a major research area in the field of clinical and medical imaging.

IVUS is a technique used to get real-time high resolution tomographic images from the inside of the coronary vessels wall and other arteries. It is able to show the presence or absence of compensatory artery enlargement. IVUS allows precise tomographic measurement of the lumen area and plaque size, distribution and, to some extent, composition of the plaque. An example of an IVUS image is shown in Figure 5.1.

To obtain insight into the status of an arterial segment, a so-called catheter pullback is carried out: an ultrasound probe (Figure 5.2) is positioned distally (downstream) of the segment of interest and then mechanically pulled back (today typically at a speed of 0.5mm/s) during continuous image acquisition to the proximal (upstream) part of the segment of interest. Experienced users may then conceptualize the complex 3D structure of the morphology and pathology of the arterial segment from this stack of images by reviewing such a sequence repeatedly. Typically, one such pullback sequence consists of 500-1000 images, which represents about 50 mm of vessel length.

As we can see from Figure 5.1, IVUS images contain image artifacts, drop-out regions and different kinds of tissue. Furthermore, manual segmentation of IVUS images is very time consuming and highly sensitive to intra- and inter-observer variability [16], while the data sets are very large. This makes IVUS image analysis a non-trivial medical application domain where a sophisticated image interpretation approach is warranted.

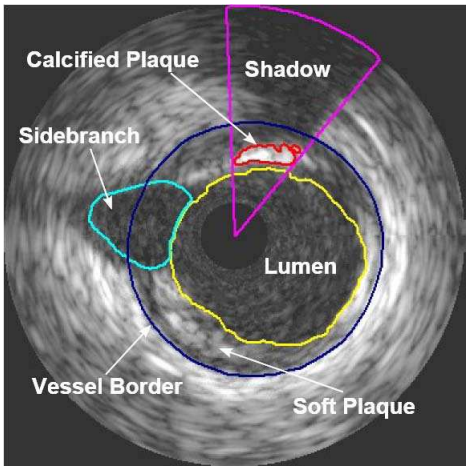


Figure 5.1: An Intravascular Ultrasound (IVUS) image with detected features. The black circle in the middle is where the ultrasound imaging device (catheter) was located. The dark area surrounding the catheter is called the *lumen*, which is the part of the artery where the blood flows. Above the catheter a *calcified plaque* is detected which blocks the ultrasound signal causing a dark *shadow*. Between the inside border of the vessel and the lumen there are some soft plaques, which do not block the ultrasound signal. The dark area left of the catheter is a *sidebranch* (another vessel).

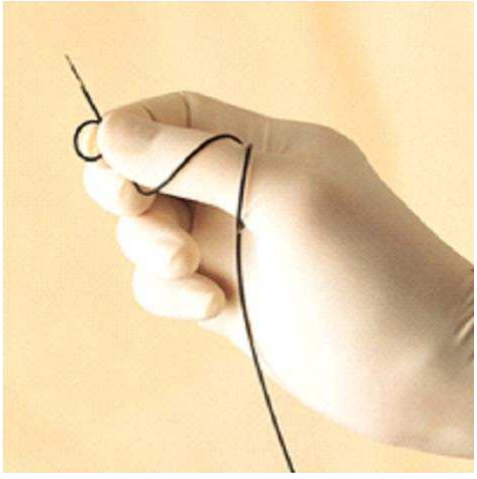


Figure 5.2: A catheter ($\varnothing \pm 1mm$) with a miniaturized ultrasound transducer at the tip.

5.2.1 Multi-Agent Segmentation of IVUS Images

In [16, 17] a state-of-the-art multi-agent system is used to detect *lumen, vessel, shadows, sidebranches* and *calcified plaques*. The system, as shown in Figure 5.3, is based on the cognitive architecture Soar (States, operators and results). Soar is an architecture for constructing general intelligent systems which has been tested successfully on many standard AI problems over the past 20 years and has been used in many real-world applications [87, 99]. It is a very good architecture for an image interpretation system as it satisfies the following robustness system requirements [16]:

- Its *subgoaling* architectural mechanism allows it to take appropriate actions in unknown situations,
- Its non-monotonic reasoning allows it to recover from faulty knowledge,
- An agent always takes into account all available knowledge.

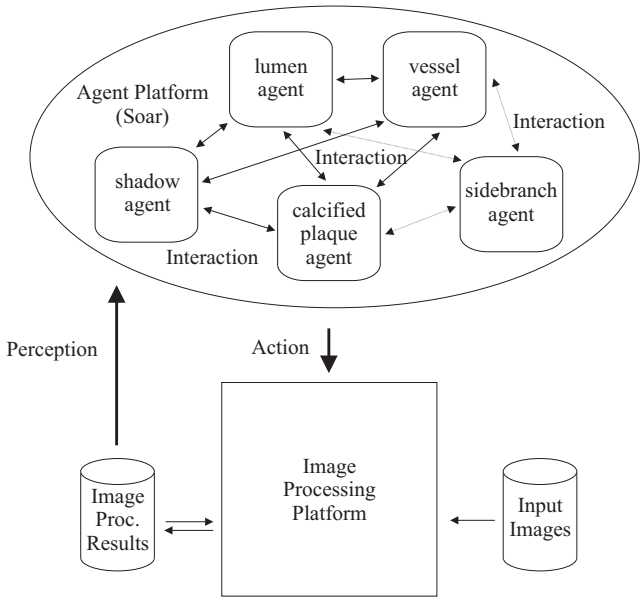


Figure 5.3: Global view of the multi-agent system architecture as applied to Intravascular Ultrasound (IVUS).

With regard to IVUS features detection, image processing agents in Figure 5.3 interact with other agents through communication, act on the world by controlling and adapting image processing operations and perceive that same world by accessing image processing results. Agents thereby dynamically adapt the parameters of low-level image segmentation algorithms based on knowledge of global

constraints, contextual knowledge, local image information and personal beliefs. The lumen-agent, for example, encodes and controls an image processing pipeline which includes binary morphological operations, an ellipse-fitter and a dynamic programming module, and it determines all relevant parameters. Generally, agent control allows the underlying segmentation algorithms to be simpler and to be applied to a wider range of problems with higher reliability.

5.3 Application to IVUS Lumen Detection

After testing different strategies of MIES on several artificial problems in chapter 4 which are more or less equivalent to the present problem, but can be evaluated much faster than image processing pipeline, we used MIES to find optimal parameter settings for the segmentation of the lumen in IVUS images instead of manual trial and error. Figure 5.4 shows how the MIES optimizer is integrated into the lumen detection system. The complete image processing pipeline is shown in Figure 5.5. We focused on the lumen detector, because it can produce good results in isolation about additional information about *sidebranches*, *shadows*, *plaques* and *vessels*.

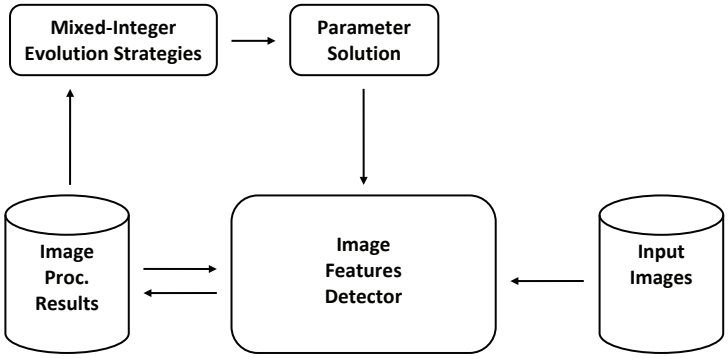


Figure 5.4: Optimizing parameter settings for lumen feature detector with MIES optimizer.

5.3.1 Fitness Functions

The fitness evaluation determines which offspring will serve as new parents in the next generation step. So the definition of the fitness function is crucially important for a successful application of MIES and should represent the success of the image segmentation very well. We first experimented with a similarity measure $S(c_1, c_2)$ between the contour c_2 found by the lumen feature detector and the desired lumen contour c_1 drawn by a human expert. The similarity measure is defined

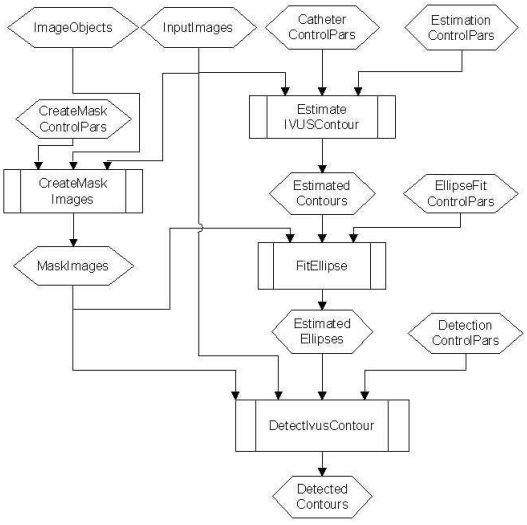


Figure 5.5: Simplified Intravascular Ultrasound (IVUS) lumen detection represented as a cascade of basic image segmentation algorithms linked together by the lumen agent.

as percentage of points of contour c_1 that are less than a τ distance away from contour c_2:

$$S(c_1, c_2) = \frac{\sum_{i=1}^{\text{nrofpoints}} \theta(i)}{\text{nrofpoints}}, \text{with } \theta(i) = \begin{cases} 1 & \textit{iff } d(c_1(i), c_2) < \tau \\ 0, & \text{otherwise} \end{cases} \tag{5.1}$$

where $d(c_1(i), c_2)$ is the Euclidian distance between a point i on contour c_1 and contour c_2, *nrofpoints* is the number of points on contour c_1, and τ is a preset threshold value. This threshold τ determines that two contours are to be considered similar when the distance between all points on contour c_1 are within a distance τ of c_2. The reason to allow for a small difference between the two contours is that even an expert will not draw the exact same contour twice in a single IVUS image. The fitness function itself is the calculated average *similarity* over all images in a training set.

Although this measure seemed to give good results while looking at the fitness values, visual inspection showed unexpected behavior as shown in Figure 5.6. The reason for this behavior is that there was no penalty on the *amount* of distance of contour points from the target contour. As a result contours with relatively few of these error points could still have a high similarity $S(c_1, c_2)$ value although visual inspection showed otherwise. To take such effects into account we changed from using a *similarity* measure $S(c_1, c_2)$ to a *dissimilarity* measure $D(c_1, c_2)$ which is

defined as follows[1]:

$$D(c_1, c_2) = \sum_{i=1}^{\text{nrofpoints}} \theta(i), \text{ with } \theta(i) = \begin{cases} d(c_1(i), c_2) & \textit{iff } d(c_1(i), c_2) > \tau \\ 0 & \text{, otherwise} \end{cases} \tag{5.2}$$

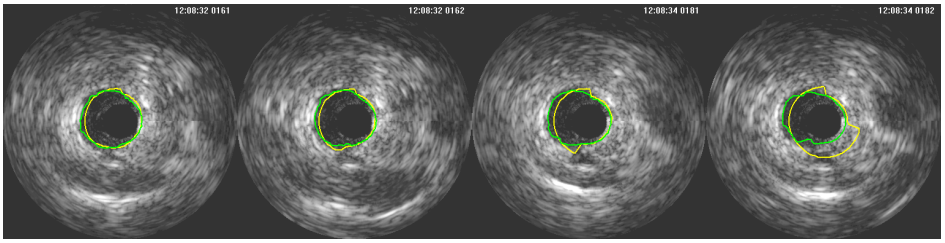


Figure 5.6: Expert-drawn lumen contours (green) compared with a MIES parameter solution (yellow) using the similarity measure $S(c_1, c_2)$ (Eq.5.1). The images show that large errors may occur even though the fitness of the solution is very good.

This measure penalizes each contour point which is more than a distance τ away from c_2 and is proportional to the distance. It leads to much better results, which stresses the importance of choosing an appropriate fitness function in this problem domain.

5.3.2 Optimizer Set-up

The settings used for the MIES algorithm were $(\mu = 4, \lambda = 28)$ with comma strategy. The evaluation of a fitness function is a very time-consuming task. To give an example: the evaluation of one setting of the MIES algorithm on 40 IVUS images for 100 iterations with 4 parents and 28 offspring took about 16 hours on a Pentium 4 (3.4GHz) computer. Evaluating these same settings on a fast to evaluate artificial problem with the same number of evaluations took 1 hour. Table 5.1 contains the parameters for the IVUS lumen image processing pipeline (cf. Figure 5.5) together with their type, range, dependencies and the default settings determined by an expert. As can be seen the parameters are a mix of continuous, ordinal discrete (integer) and nominal discrete(including boolean) variables.

5.3.3 Results

For the experiments we used five disjoint sets of 40 images. The images were acquired with a 20 Mhz Endosonics Five64 catheter using motorized pullback (1 mm/s). Image size is 384×384 pixels (8 bit grayscale) with a pixel-size of

[1]For MIES, the *similarity* measure means maximization while the *dissimilarity* measure means minimization.

name	**type**	**range**	**dependencies**	**default**
maxgray	integer	[2, 150]	> mingray	35
mingray	integer	[1, 149]	< maxgray	1
connectivity	nominal	{4,6,8}		6
relativeopenings	boolean	{false,true}		true
nrofcloses	integer	[0, 100]	used if not relativeopenings	5
nrofopenings	integer	[0, 100]	used if not relativeopenings	45
scanlinedir	nominal	{0,1,2}		1
scanindexleft	integer	[-100, 100]	< scanindexright	-55
scanindexright	integer	[-100, 100]	> scanindexleft	7
centermethod	nominal	{0,1}		1
fitmodel	nominal	{ellipse, circel}		ellipse
sigma	continuous	[0.5 10.0]		0.8
scantype	nominal	{0,1,2}		0
sidestep	integer	[0, 20]		3
sidecost	continuous	[0.0, 100]		5
nroflines	integer	[32, 256]		128

Table 5.1: Parameters for the IVUS lumen image processing pipeline.

0.02602 mm^2. For the fitness function we took the average dissimilarity over all 40 images with τ set to 2.24 pixels and *nrofpoints* = 128 (see Eq. 5.2). For each of the 5 datasets we used the $(4, 28)$ MIES algorithm and limited the number of iterations to 25 which resulted in 704 fitness evaluations for each dataset. The training results are displayed in Table 5.2, where MIES solution 1 was trained on dataset 1 by the MIES algorithm, MIES solution 2 was trained on dataset 2, etc ....

Table 5.2 shows that for most cases the MIES-generated parameter solutions result in lower average contour differences when applied to both test- and training data than the default parameters. Only parameter solution 3 applied to dataset 5 has a higher average contour difference (444.2 vs 446.4). To determine if the best results obtained by the MIES algorithm are also significantly better than the default parameter results, a paired two-tailed t-test was performed on the (40) difference measurements for each image dataset and each solution using a 95% confidence interval (p = 0.05). The t-test shows that all differences are significant except for the difference between MIES solution 3 applied to dataset 5 and the default parameters and the difference between MIES solution 5 applied to dataset 3 and the default parameters. Therefore we conclude that the MIES solutions are significantly better than the default parameter solution in 92% of the cases (23 out of 25) and equal in the other two cases.

When we look at the results of the ES parameter solutions compared to the default parameter solution we see that all the differences are statistically significant meaning that the ES solutions are significantly better than the default parameter solution in 23 out of 25 cases but worse in the other 2 cases (ES solutions 3 and 4 applied to dataset 5).

If we look at the performance of the MIES and ES algorithms when trained on a dataset we see that on Dataset 1 the ES solution is a little better, but the difference is not statistically significant. On all other datasets the MIES solution trained on that dataset is significantly better than the ES solution trained on the same dataset. On Dataset 5 MIES solution 4 has a slightly lower fitness than MIES solution 5 that was trained on the dataset but the difference is not statistically significant. On Dataset 3, ES solution 4 has a lower fitness than ES solution 3 but again the difference is not significant.

In order to learn from the results about advantageous parameter settings, we compared the variable settings of optimized solutions to solutions with an average fitness value (obtained at the beginning of the evolution). The results are displayed in the parallel coordinates diagram (Figure 5.7). It is apparent that the setting of some parameters seems to be clearly advantageous in a certain small range while for others the setting is either indifferent or it may depend on other parameter settings. One can observe that a scantype of 2, a medium value for the maxgray parameter and sidecost (around 70) and a high sigma value (close to 10.0) seems to be beneficial. Of course these results hold only for one image set and future research needs to clarify whether generalizations are feasible. Visual inspection of the results of the application of MIES parameter solution 4 to all 200 images

	Dataset 1		Dataset 2		Dataset 3		Dataset 4		Dataset 5	
	Fitness	**S.D.**	**Fitness**	**S.D.**	**Fitness**	**S.D.**	**Fitness**	**S.D.**	**Fitness**	**S.D.**
Default Parameters	395.2	86.2	400.2	109.2	344.8	66.4	483.1	110.6	444.2	90.6
MIES Solution 1	151.3	39.2	183.6	59.0	201.0	67.1	280.9	91.9	365.5	105.9
MIES Solution 2	160.3	45.9	181.4	58.7	206.7	70.3	273.6	74.5	372.5	99.2
MIES Solution 3	173.8	42.1	202.9	69.1	165.6	47.2	250.7	80.2	446.4	372.8
MIES Solution 4	154.0	51.7	243.7	67.7	198.8	80.1	186.4	59.0	171.3	57.8
MIES Solution 5	275.7	75.6	358.4	76.9	327.7	56.7	329.1	82.0	171.8	54.5
ES Solution 1	149.8	46.0	185.1	60.1	203.1	66.6	270.2	69.8	358.1	100.1
ES Solution 2	157.8	45.2	183.0	57.3	202.8	68.1	270.7	69.4	364.2	98.5
ES Solution 3	212.6	46.3	243.3	72.3	182.3	49.7	252.9	58.4	756.8	734.4
ES Solution 4	247.2	143.0	255.7	85.4	176.4	49.2	232.9	62.4	941.0	795.2
ES Solution 5	217.0	51.6	225.3	68.5	292.8	73.6	330.7	78.0	324.7	96.0

Table 5.2: Performance of the best found MIES and ES parameter solutions when trained on one of the five datasets ((MI)ES solution i was trained on dataset i). All parameter solutions and the (default) expert parameters are applied to all datasets. Average difference (fitness) and standard deviation w.r.t. expert drawn contours are given.

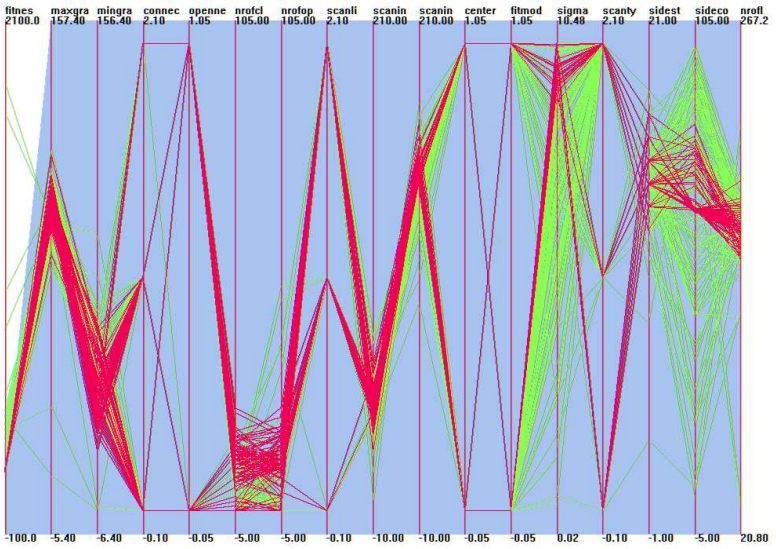


Figure 5.7: The parallel coordinate diagrams shows a comparison of optimized solutions (red polygons) to solutions with a relative bad fitness value (light green polygons). The first coordinate is the fitness value and the subsequent coordinates are the values of the variables (Lumen detector parameters) represented in the same order as in table 5.1.

shows that this solution is a good *approximator* of the lumen contours as can be seen in Figure 5.8 (bottom row). When we compare the contours generated with MIES solution 4 to the expert drawn contours we see that they are very similar and in some cases the MIES contours actually seem to follow the lumen boundary more precisely. Besides being closer to the expert drawn contours, another major difference between the MIES found contours and the contours detected with the default parameter settings is that the MIES solutions are smoother (see Figure 5.8, top and bottom row). Apart from looking at the average contour difference (or fitness) of the different parameter solutions we can also compare the performance between the MIES and ES algorithms by looking at their ability to "learn" the dependencies between the variables as displayed in Table 5.1. In Figure 5.9 the total number of illegal solutions evolved by both the MIES and ES algorithms are displayed. As can be seen the MIES algorithm manages to "learn" the dependencies much faster than the ES algorithm. In Figures 5.10 and 5.11 we have plotted the fitness and best fitness for both the MIES and ES algorithms on Dataset 2. Invalid solutions were given a very high fitness penalty and are omitted from the plots to improve readability. In the case of the MIES algorithm the spread of the entire population decreases as the population reaches the best solution, which indicates that the step-size adaptation works properly.

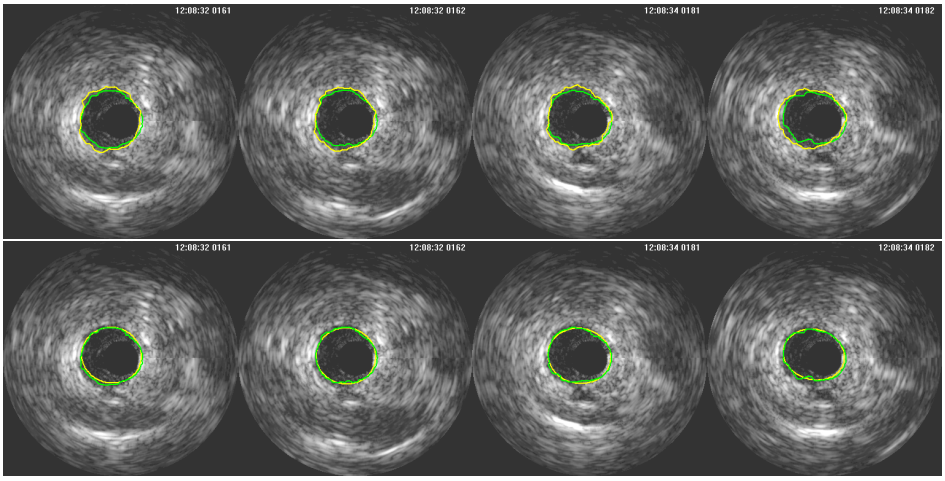


Figure 5.8: Expert-drawn lumen contours (green) compared with expert-set parameter solution (yellow, top row) and MIES parameter solution (bottom row, yellow).

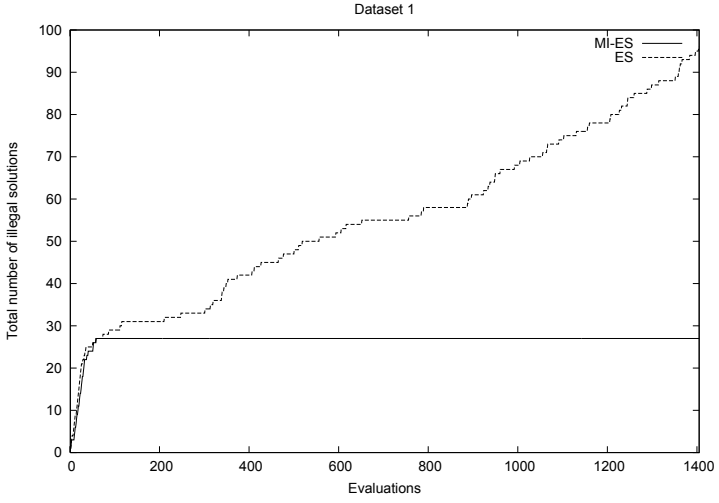


Figure 5.9: The accumulated number of illegal solutions on Dataset 2 evolved by both the MIES and ES (dotted line) algorithms. As can be seen the MIES algorithm manages to "learn" the dependencies quite fast while the ES algorithm keeps evolving invalid solutions even after 1400 fitness evaluations.

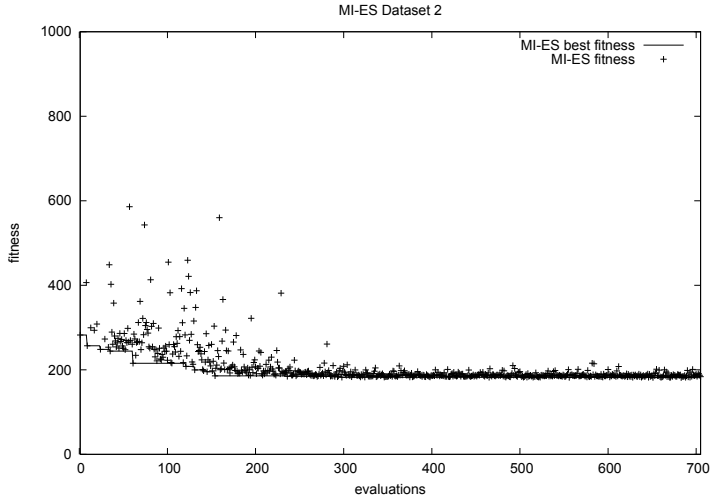


Figure 5.10: The fitness and best fitness values during the run of the MIES algorithm. As can be seen the entire population of the MIES algorithm quickly converges to the best found fitness.

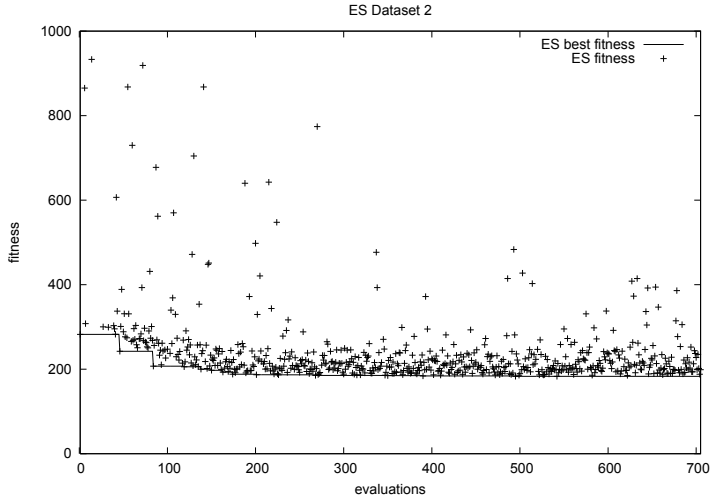


Figure 5.11: The fitness and best values fitness during the run of the ES algorithm. As can be seen the population does not converge to the best fitnes.

5.4 Summary

In this chapter we described how we applied MIES to a problem in medical image analysis, in particular the optimization of control parameters of a lumen detector in IVUS imaging. In particular, MIES use specific variation operators for different types (continuous, integer, and nominal discrete) of decision parameters which control the features detector of IVUS images. All three types of mutation operators support automatic adaptation of the mutation strength and avoid biased sampling. Besides this, they fulfill guidelines such as accessibility, uniformity, and maximal entropy, which makes them very amenable as search operators in settings with little or no a-priori knowledge about the search landscape.

Like the experimental results which we showed in chapter 4, a similar result is obtained for the medical image analysis. Here the MIES always produced better or equal results than the default parameter settings chosen by an expert. Moreover, on all five data sets the results of the MIES were significantly better (three times) or equal (one time) than those obtained with the standard ES, trained on the same data set.

In summary, the results show that the MIES is a valuable technique for improving the parameter settings of the lumen detector. The results encourage further studies on extended image sets and on other feature detectors. The results of this study suggest also its use in other problems where parameters of image-analysis modules need to be tuned based on training data, and more generally - in large-scale mixed-integer optimization problems that cannot be solved with standard mathematical programming techniques.

Chapter 6

Dynamic Fitness Based Partitioning

In the previous chapters 3, 4 and 5, Mixed-Integer Evolution Strategies (MIES) have been introduced and studied intensively from both theoretical and experimental viewpoints. Being a promising technique, MIES have been successfully used to tackle challenging parameter optimization of a multi-agent image interpretation system for Intravascular Ultrasound (IVUS) images lumen detection. However, with regard to the image analysis problem, because of the complexity of interpretation contexts, it is impossible to find one "*super optimal*" solution for each feature detector to work in all possible contexts and for all possible patients. Therefore, it would be wise to find specific optimal parameter settings for different groups of images instead of one global solution for all images, that is, let a set of MIES algorithms find a set of optimal solutions for sets of optimal images whereby the solutions and sets of images are evolved automatically.

In this chapter, we will investigate this issue and propose one technique, what we called *Fitness Based Partitioning*. By using *Fitness Based Partitioning*, we would like to find groups of images that require a similar parameter setting for the segmentation algorithm while, at the same time, evolving optimal parameter settings for these groups. More specifically, we will apply this methodology to both a challenging artificial test problem and feature detection of Computer Tomographic Angiography (CTA) images analysis. Experimental results not only demonstrate the feasibility of *Fitness Based Partitioning*, but also show that MIES can also be used for different types of medical images other than IVUS images, for instance, feature detection of Computer Tomographic Angiography (CTA) images analysis. It is rather trivial to see that the applicability of the optimization algorithm does not depend on the images but on the image analysis tools which are applied to the images - to the specific parameter encoding.

6.1 Introduction

Medical images often represent complex and variable structures that can not be easily modeled. Moreover, they can suffer from a range of imperfections due to the image acquisition modalities. Today's methods, directed at the automated recognition of certain structures in images, are applicable only over a limited range of standard situations and in some cases only reach suboptimal results. In chapter 5 we have compared Mixed-Integer Evolution Strategies (MIES) and standard Evolution Strategies (ES) for finding optimal parameter settings for the segmentation of Intravascular Ultrasound images. The results show that the parameter solutions evolved by the MIES and ES algorithms are better than the original parameter settings. However, the results also indicate that different sets of images require different parameter settings for an optimal image segmentation.

The ideal solution would be to cluster images according to their image segmentation context and optimize parameters for each individual context separately. Unfortunately the number of image segmentation contexts is not known a priori nor do their characteristics. There is usually also no natural distance measure [54] to cluster images into groups that need similar parameter settings for an optimal segmentation result. Only their degree of belonging to a group, characterized by a particular set of parameters, can be measured by means of a training error for that image, after the parameters have been optimized for that group.

A possible approach for this kind of multi-level optimization problem could be cooperative coevolution (e.g., see [120, 97]) in which one evolves both a set of parameter solutions and sets of images at the same time. However, this approach requires a large number of fitness evaluations which is very computationally (and thus time) intensive, since one has to do a lot of image processing, and therefore not attractive for our problem.

To solve the aforementioned problems we propose a multi-level optimization technique - the so-called *Fitness Based Partitioning*. Given a set of parameter solutions, we can partition the images according to which solution gives the best segmentation result. The fitness measure is then used as a "distance metric" to determine which partition (and corresponding MIES solution) is the best match for an IVUS image. By alternating partitioning and parameter optimization for each partition, images are dynamically repartitioned and parameter solutions are optimized.

This chapter is structured as follows: *Fitness based partitioning* will be introduced in Section 6.2. This approach will then be tested on an artificial test problem in Section 6.3 where the goal is to find multi-dimensional clusters by evolving combinations of uniform and normal distributions based on given data points in a multi-dimensional space. Next, in Section 6.4, Computed Tomographic Angiography (CTA) lumen detection will be introduced. *Fitness Based Partitioning* will then be applied to CTA lumen segmentation. The goal is to dynamically partition the CTA image training set during the MIES parameter optimization process into groups of images which require similar parameter settings for optimal

lumen segmentation. Each group of images would correspond to a similar image segmentation context (for the image segmentation algorithm) and have an optimal parameter solution. Some important experimental results will be presented as well in this section. The short conclusions and outlook for the future work will be given eventually.

6.2 Dynamic Fitness Based Partitioning

In general the multi-level optimization task is to find a proper fit of partitioning comprising N_P partitions; for each of the partitions P_k ($k \in [1, N_P]$) we search for parameter settings which will result in an optimal solution for all problem instances in P_k. More concretely, in the case of lumen segmentation, we try to partition all angiographic images, and for each image partition we look for a parameter solution which results in the best possible lumen segmentation for those images. In order to solve this multi-level optimization problem we designed a 2-level algorithm with an inner and an outer loop.

In the outer loop the goal is to redistribute problem instances in order to achieve an improved global quality and to balance the size of the partitions. Aiming for this, a deterministic approach will be employed to determine how problem instances should be (re-)partitioned and when to split or merge partitions.

In the inner loop the aim is to optimize parameter solutions for the problem instances in each of the N_P partitions. This task will be performed by evolutionary algorithms, in our case Mixed-Integer Evolution Strategies, since they can handle different parameter types simultaneously.

Let $\mathcal{I} = \{I_1, \ldots, I_N\}$ denote a set of images (or training instances), $\mathbf{a} \in A = \{1, \ldots, K\}^N$ an assignment of the images to one of K partitions, and $\mathbb{S}$ denote a set of control parameters for the segmentation algorithm. Then the optimization problem of finding an optimal partitioning is stated as follows:

$$\mathbf{a}^* = \arg\min_{\mathbf{a} \in A} \sum_{k=1}^{K} \mathrm{MME}_{\mathbf{a}}(k) \tag{6.1}$$

Here $\mathrm{MME}_{\mathbf{a}}(k)$ stands for 'minimized mean error' and denotes the average error on instances of a partition k over all training instances in that partition, provided the segmentation software uses an optimized set of control parameters for solutions on that partition, in symbols:

$$\mathrm{MME}_{\mathbf{a}}(k) = \min_{\mathbf{s} \in \mathbb{S}} \frac{1}{N} \sum_{j=1}^{N} \mathrm{Indicator}(a_j = k) \mathrm{error}_{\mathbf{s}}(I_j) \tag{6.2}$$

Here $\mathrm{Indicator} : \{true, false\} \rightarrow \{0, 1\}$ denotes the indicator function with $\mathrm{Indicator}(false) = 0$ and $\mathrm{Indicator}(true) = 1$. We are also interested in the

optimal parameter sets (or solution vectors) of the partitions $k = 1, \ldots, K$, i.e.

$$\mathbf{s}^*(\mathbf{a}, k) = \arg\min_{\mathbf{s}\in\mathbb{S}} \frac{1}{N} \sum_{j=1}^{N} \text{Indicator}(a_j = k)\text{error}_{\mathbf{s}}(I_j), \tag{6.3}$$

in particular in those for the optimized partitioning $\mathbf{a}^*$.

More concretely, in the case of lumen segmentation we want to automatically find groups of medical images while at the same time evolving a set of optimal parameters for detecting the lumen in the images in each of these groups.

In order to solve this multi-level optimization problem we use Fitness Based Partitioning. The top level goal is to optimize the (re-)assignment of problem instances, in our case medical images, to partitions so that the optimal solution for each partition is also the optimal solution for each particular problem instance in that partition.

The second level optimization task is to find an optimal solution for all problem instances within a partition. For this we use Mixed-Integer Evolution Strategies (MIES), introduced in [38]. Mixed-Integer Evolution Strategies are a special type of evolution strategy that can handle mixed-integer parameters (continuous, ordinal discrete, and nominal discrete) by combining mutation operators of Evolution Strategies in the continuous domain [107], for integer programming [101], and for binary search spaces [3].

6.2.1 Algorithm

The detailed procedure for this 2-level optimization method is described in Algorithm 9. During the initialization phase all the problem instances (e.g., images) are distributed over the K partitions. Next a MIES algorithm MIES_k is assigned to each partition P_k.

The main loop of the Fitness Based Partitioning algorithm consists of four steps. The first step is to run each MIES_k algorithm on the problem instances in its corresponding partition P_k for G iterations. This step performs the second level optimization task.

The second step is to select the best evolved parameter solution $\mathbf{s}_k$ evolved by each MIES_k algorithm and to test it on all problem instances.

Step 3 is then to reassign all problem instances so that each problem instance I is assigned to the partition whose corresponding MIES algorithm offers the best parameter solution. This step performs the top level optimization task.

After all the problem instances have been reassigned to their "new" partitions the fourth step is to check for "empty" partitions (partitions with no problem instances). Empty partitions are not useful, since their corresponding MIES algorithms cannot optimize anything. The solution we have chosen is to move half the problem instances of the largest partition to the empty partition. Additionally, we replace the population of the MIES algorithm associated with the empty partition with a copy of the population of the MIES algorithm associated with the largest

partition. This effectively removes a non-useful empty partition and splits a large partition into two. Another choice which might be more effective sometimes could be to split the partition in which results of MIES_s have the largest variance.

Algorithm 9 Fitness Based Partitioning

```
/* Initialization */
Divide the set of problem instances 𝓘 randomly over the partitions.
Initialize the populations of the K MIES algorithms.
for T main loop iterations do
   /* step 1 */
   for each partition P_k do
      run MIES_k on P_k for G iterations.
   end for
   /* step 2 */
   for each MIES_k do
      select best individual/solution s_k
      apply best individual/solution s_k to all problem instances in 𝓘
   end for
   /* step 3 */
   for each problem instance I ∈ 𝓘 do
      redistribute I to the partition P_k for which s_k offered the best solution.
   end for
   /* step 4 */
   while the smallest partition P_S is empty do
      copy the population of MIES_L of the largest partition P_L to MIES_S
      divide the problem instances of P_L over P_L and P_S.
   end while
end for
```

6.3 Artificial Test Problems and Results

In this section we test the feasibility of "*fitness based partitioning*" on artificial problems as a first step toward its application to the real CTA lumen feature detector system. This is done because testing out various algorithm settings and learning about their behavior using medical images is computationally too demanding to be practical. However, our test problems are designed in such a way that success may be expected on real problems, for instance, the data used in the test problems are representative for real cases.

The basic idea of our test setup, as visualized in Figure 6.1, is the task of finding a set of multidimensional distributions based on given data points. Two parts of the test problem need to be distinguished: (1) initialization/setup phase, (2) evaluation of a solution. Next, we give a brief description of both phases, followed by the detailed description of experiments and results.

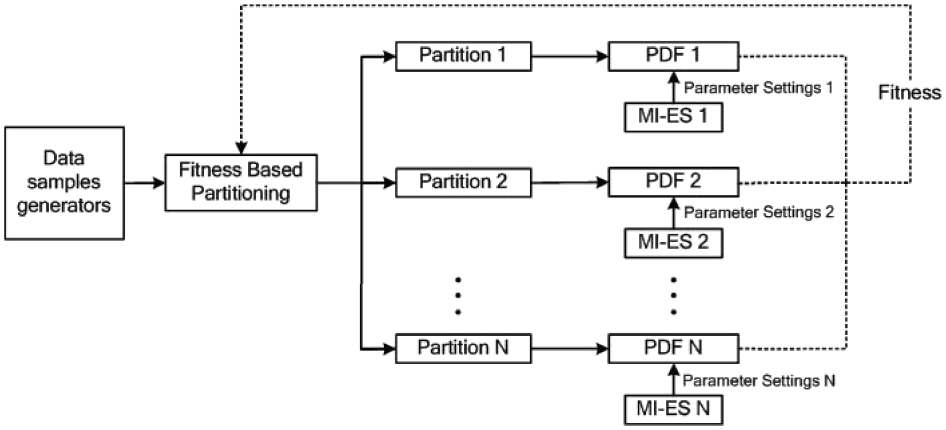


Figure 6.1: Fitness based partitioning for randomly generated data samples

6.3.1 Initialization/Setup

In the *initialization/setup phase*, the problem generator creates sample points in D-dimensional space using a random number generator which can generate values using either a uniform or normal distribution. Using the problem generator we created N_P "clusters" of sample points.

In more detail, the initialization procedure samples a set of N_I points $\mathcal{I} = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N_I)}\} \in (\mathbb{R}^D)^{N_I}$. The points are realizations of N_P different D - dimensional random variables $X_1, \ldots, X_{N_P}$. For each random variable N_I/N_P points are generated independently. For any $k \in [1, N_P]$, the distribution of the random variable X_k is determined by the parameters $\mu_d^{(k)}$, $\sigma_d^{(k)}$. The distribution of each random variable is an independent joint distribution composed of uniform and normal distributions. The values at the odd vector positions are sampled from 1-D normal distributions with mean value $\mu_d^{(k)}$ and standard deviation $\sigma_d^{(k)}$. The values at the even vector positions are sampled from 1-D uniform distributions with interval width $4\sigma_d^{(k)}$ and mean value $\mu_d^{(k)}$.

6.3.2 Evaluation

The test problem is to estimate the parameters and distribution types of the N_P multivariate distributions based on the initialized data points. We work with the following representation of solutions, encoded in the individuals of the EA. For each dimension $d \in [1, D]$ an individual has three parameters: an estimated mean value $\hat{\mu}_d \in \mathbb{R}$, an estimated standard deviation or, in case of uniform distribution, interval width $\hat{\sigma}_d \in \mathbb{R}$ and an estimated distribution type $\hat{\tau}_d$ (0: uniform, 1: normal). In case of a uniform distribution the minimum and maximum possible value are now defined as $\hat{\mu}_d^{(k)} - 2\hat{\sigma}_d^{(k)}$ and $\hat{\mu}_d^{(k)} + 2\hat{\sigma}_d^{(k)}$ respectively. Thus each individual

looks like: $(\hat{\mu}_1^{(k)}, \hat{\sigma}_1^{(k)}, \hat{\tau}_1^{(k)}, \ldots, \hat{\mu}_d^{(k)}, \hat{\sigma}_d^{(k)}, \hat{\tau}_d^{(k)}, \ldots, \hat{\mu}_D^{(k)}, \hat{\sigma}_D^{(k)}, \hat{\tau}_D^{(k)})$. where k denotes the partition the individual represents.

For the fitness function we use a maximum log-likelihood approach, whereby for each individual the fitness is calculated as:

$$\text{fitness} = \sum_{i=1}^{|P_k|} \sum_{d=1}^{D} \log[PDF_{\hat{\mu}_d^{(k)}, \hat{\sigma}_d^{(k)}, \hat{\tau}_d^{(k)}}(\mathbf{x}_d^{(\underline{k}_i)})], \tag{6.4}$$

where $\mathbf{x}_d^{(\underline{k}_i)}$ denotes the d-th dimensional value of the i-th sample point from partition P_k.

The probability density function $PDF_{\hat{\mu}_d^{(k)}, \hat{\sigma}_d^{(k)}, \hat{\tau}_d^{(k)}} : \mathbb{R} \to [0, 1]$ is described as:

$$PDF_{\hat{\tau}_d^{(k)}} = \begin{cases} \mathrm{I}(\mathbf{x}_d^{(\underline{k}_i)} \in [\hat{\mu}_d^{(k)} - 2\hat{\sigma}_d^{(k)}, \hat{\mu}_d^{(k)} + 2\hat{\sigma}_d^{(k)}]) \frac{\mathbf{x}_d^{(\underline{k}_i)} - \hat{\mu}_d^{(k)}}{4\hat{\sigma}_d^{(k)}} & \hat{\tau}_d^{(k)} = 0 (\text{uniform}) \\ \frac{1}{2\pi} \exp(-\frac{\mathbf{x}_d^{(\underline{k}_i)} - \hat{\mu}_d^{(k)^2}}{2(\hat{\sigma}_d^{(k)})^2}) & \hat{\tau}_d^{(k)} = 1 (\text{normal}) \end{cases}$$

with $d = 1, \ldots, D$, $i = 1, \ldots, |P_k|$, $k = 1, \ldots, N_P$, and $\mathrm{I} : \{true, false\} \to \{0, 1\}$ being the indicator function: $\mathrm{I}(true) = 1, \mathrm{I}(false) = 0$.

6.3.3 Experimental Results

In this experiment, the MIES algorithms were programmed using the *Evolving Objects* library (EOlib) [62]. EOlib is an Open Source C++ library for all forms of evolutionary computation and is available from `http://eodev.sourceforge.net`. The test-data generator was created using the random number generator from EOlib. For each combination of dimensionality and number of clusters we created 10 problem instantiations and on each problem instantiation we ran the fitness-based partitioning system 20 times using different random seeds for the MIES algorithms. Each generated cluster consists of 100 sample points.

For the MIES algorithms we used a plus-strategy with a population size of 40 and an offspring size of 280. After each redistribution cycle MIES algorithms were run for T iterations, with T dependent on the dimension D of the sample points. T was set to 50 for $D = 2$, to 75 for $D = 4$, and to 100 for $D = 6$.

The results in Table 6.1 show that, in most cases, the fitness-based partitioning system manages to evolve a combination of uniform and normal distributions to describe each cluster. However, for $D = 4$ and $N_P = 20$, the system fails in two cases. In the first case a partition of 101 and a partition of 99 sample points result (vs. 100 each). In the second case, one partition is split into 2 smaller partitions containing 50 sample points each, while 2 other partitions are merged into one larger partition with 200 sample points. For the 6 dimensional problem with 10 clusters the only failure was a single sample point that was mispartitioned as well.

Dimensions D	T	Partitions N_P	Succesful/Total runs	Iterations Outer Loop			
				Average	S.D.	Minimum	Maximum
2	50	3	200/200	7.375	3.57	1	18
4	75	10	200/200	16.07	3.44	9	31
4	100	20	198/200	23.21	3.44	16	36
6	100	10	199/200	12.35	4.45	5	43
6	100	20	197/200	14.87	3.66	9	34

Table 6.1: The results of the different experiments. Iterations outer loop (successful runs) means that all the N-dimensional data points that were originally created in a cluster end up in the same partition. Since the MIES algorithms have to find the optimal distribution parameters for each dimension, the number of variables to optimize is three times the dimension of the data points to be partitioned. For the successful runs we have measured the average, minimum and maximum number of iterations as well as standard deviation (S.D.), until a stable partitioning was reached.

6.4 Computed Tomographic Angiography and Experimental Results

Since the introduction and increasing propagation of modern multi-slice computed tomography scanners, computed tomographic angiography has become a popular diagnostic modality in the visualization and evaluation of arteries and the detection of narrowings (stenoses). Computed Tomography is an imaging technique which results in a 3D image of the internals of an object using a series of 2D X-ray images.

In Leiden University Medical Center (LUMC), a system has been developed for the quantitative analysis of coronary Computed Tomographic Angiography (CTA) [84] which consists of 5 steps. In the first step the vessels are segmented in the 3D image, followed in step 2 by the extraction of the vessel centerline. The third step is to construct a curved multiplanar reformatted (CMPR) image using the detected centerline (see Figure 6.2). The resulting 3D image stack contains 2D images perpendicular to the centerline, and allows for the visualization of the the entire length of the vessel in a single 3D image. The fourth step is the segmentation of the lumen boundary (the part of the vessel where the blood flows) using a combination of longitudinal and transversal contour detection. It is this step that we will optimize using Mixed-Integer Evolution Strategies (MIES) and Fitness Based Partitioning. The fifth step is the quantification of the vessel morphological parameters.

6.4.1 Experiments and Results

The Fitness Based Partitioning approach as described above is tested on 9 CMPR image stacks of coronary arteries. Each CTA image stack consists of 59 to 82 images and each image consists of 32×32 pixels (16 bit signed grayscale with a

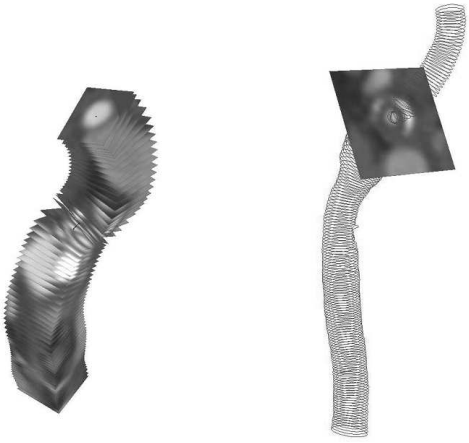


Figure 6.2: A stack of CMPR images on the left with the centerline going through the center of each image and the corresponding lumen contours with a single CMPR slice on the right.

spacing of 0.5mm).

To test the effect of the number of partitions, we experimented with up to 6 partitions. In case 1 partition is used the algorithm behaves like a normal single MIES algorithm since there is no need to redistribute the images to other partitions. For each data set and number of partitions we run the Fitness Based Partitioning algorithm 10 times using different random seeds to initialize the MIES algorithms. To initialize the K partitions with images we simply divided a data set sequentially into K (almost) equally sized parts. We also experimented with other initialization techniques (e.g., random), but they gave slightly worse results. This is probably caused by the fact that two consecutive images in a stack correspond to two consecutive pieces of artery and therefore, in general, require a similar parameter solution.

For the MIES algorithms in step 1 of Algorithm 9 we use a plus-strategy ($\mu+\lambda$) with $\mu = 4$ parents and $\lambda = 28$ offspring individuals. All variables have their own stepsize or mutation probability parameter which undergo self-adaptation as described in [38]. The parameters for the CTA lumen segmentation consists of 13 integer and 2 nominal discrete (Boolean) parameters.

6.4.2 Evaluation

In order to evaluate the fitness of a parameter solution evolved by a MIES algorithm, the lumen contour resulting from a particular parameter setting is compared to the expert contour drawn by a physician. The fitness function computes

the average error $F_k(I)$ for each image I in partition P_k as:

$$F_k(I) = \sum_{p=1}^{|\text{points}|} \frac{d(C_p, E_p)}{|points|}, \tag{6.5}$$

where $d(C_p, E_p)$ is the Euclidean distance between the p-th point of the "evolved" contour C and the expert drawn contour E. Note that F_k corresponds to the function error in the general problem definition given in Eq. 6.1 to 6.3. Both contours have the same number of points since we resample all contours from the center of the image every 2 degrees resulting in 180 points for each contour.

The fitness of an individual parameter solution is then computed as the average minimized error of all images I in partition P_k:

$$\text{fitness} = \sum_{I \in P_k} \frac{F_k(I)}{|P_k|} \tag{6.6}$$

To determine the overall fitness result of our Fitness Based Partitioning algorithm we compute the average fitness of all images $I \in \mathcal{I}$ as:

$$\text{overall fitness} = \sum_{k=1}^{K} \sum_{I \in P_k} \frac{F_k(I)}{|\mathcal{I}|} \tag{6.7}$$

6.4.3 Results

The results in Tables 6.2 and 6.3 show that generally more partitions results in better average fitness values and thus better contours. The only exception is data set 6 where the fitness results for 6 partitions are worse than for 5 partitions, but this difference is not statistically significant (using an independent samples t-test with a 95% confidence level(p=0.05)). If we look at the differences in fitness values between 1 and 2 partitions we see that only for data set 9 the difference in fitness values is not statistically significant. This indicates that for our problem we should use at least 2 partitions. For data set 2 all differences between consecutive number of partitions (1 and 2, 2 and 3, ...) are statistically significant. For data sets 4 and 5 the difference between 3 and 4 partitions is statistically significant which could mean that for these data sets we should use at least 4 partitions. We see the same for data sets 7 and 8 where the difference in average fitness value between partitions 4 and 5 is statistically significant.

When we look at the final image partitioning after the algorithm has ended we see that different random seeds (and thus MIES population initializations) do not always lead to exactly the same partitions. However, an analysis of the found partitions shows that we can clearly see groups of images which repeatedly end up in the same partitions. There are several reasons why we do not see all images end up in similar partitions every single run. The main reason seems to be that the partitioning process does not always stabilize for some random seeds.

Data Set	Number of Partitions											
	1				2				3			
	avg	s.d.	min	max	avg	s.d.	min	max	avg	s.d.	min	max
1	0.242	0.004	0.236	0.248	0.203	0.011	0.183	0.227	0.196	0.010	0.183	0.214
2	0.152	0.003	0.148	0.159	0.135	0.005	0.128	0.146	0.124	0.003	0.119	0.129
3	0.176	0.004	0.169	0.182	0.165	0.006	0.156	0.173	0.157	0.005	0.150	0.168
4	0.186	0.006	0.175	0.193	0.169	0.005	0.162	0.182	0.162	0.004	0.156	0.169
5	0.327	0.016	0.297	0.364	0.272	0.009	0.259	0.293	0.261	0.009	0.240	0.275
6	0.320	0.026	0.275	0.343	0.257	0.041	0.195	0.321	0.209	0.032	0.185	0.298
7	0.307	0.011	0.276	0.313	0.253	0.014	0.232	0.278	0.232	0.017	0.204	0.254
8	0.169	0.001	0.167	0.170	0.155	0.009	0.143	0.167	0.150	0.007	0.137	0.160
9	0.199	0.011	0.185	0.219	0.186	0.027	0.153	0.227	0.153	0.013	0.141	0.181

Table 6.2: The average (plus standard deviation), minimum and maximum overall fitness values for 1 to 3 partitions. Lower values correspond to better contours.

Data Set	Number of Partitions											
	4				5				6			
	avg	s.d.	min	max	avg	s.d.	min	max	avg	s.d.	min	max
1	0.184	0.007	0.177	0.203	0.175	0.006	0.166	0.184	0.171	0.005	0.159	0.178
2	0.118	0.003	0.113	0.124	0.113	0.002	0.110	0.116	0.108	0.003	0.104	0.113
3	0.151	0.004	0.144	0.160	0.149	0.005	0.141	0.156	0.144	0.004	0.140	0.152
4	0.153	0.004	0.149	0.163	0.152	0.005	0.144	0.161	0.151	0.005	0.144	0.159
5	0.233	0.009	0.215	0.249	0.229	0.012	0.213	0.253	0.223	0.012	0.209	0.249
6	0.207	0.025	0.185	0.276	0.188	0.008	0.180	0.199	0.192	0.011	0.178	0.213
7	0.228	0.010	0.215	0.250	0.210	0.010	0.194	0.234	0.197	0.010	0.184	0.218
8	0.143	0.005	0.137	0.156	0.136	0.003	0.132	0.142	0.135	0.003	0.129	0.138
9	0.146	0.010	0.134	0.170	0.140	0.006	0.135	0.156	0.135	0.006	0.130	0.150

Table 6.3: The average (plus standard deviation), minimum and maximum overall fitness values for 4 to 6 partitions. Lower values correspond to better contours.

Other possible reasons for finding different image partitionings are that there are more image segmentation contexts than partitions or maybe there are no real distinct groups of images with respect to the image segmentation parameters. Naturally, the number of image segmentation contexts also depends on our image segmentation algorithm and how robust or sensitive it is.

In Figures 6.3 and 6.4 results are shown for 2 images from data set 2 after fitness based partitioning using 2 partitions. The light gray contours in the left images are found using parameter settings evolved for a partition including the image in Figure 6.3. The light contours in the right images are found using parameter settings evolved for the other partition, which included the image in Figure 6.4. As can be see both parameter settings result in contours similar to the dark gray expert drawn contours for the image for which they were optimized but fail to find satisfactory contours in the other images.

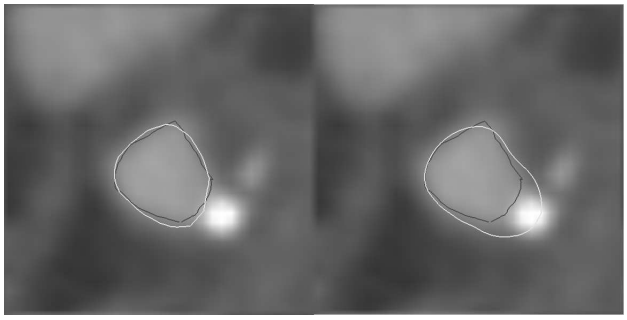


Figure 6.3: Found lumen contours segmented using two different parameters settings. The light gray contour in the left image was found using parameter settings evolved for the partition to which this image was assigned. The light gray contour on the right was found using the parameter settings evolved for the other partition. The dark contour in both images indicates the expert-drawn contour.

6.5 Summary

In this chapter we investigate the use of Fitness Based Partitioning in order to find sets of optimal parameters for the segmentation of the lumen in Computed Tomographic Angiography images. The purpose of Fitness Based Partitioning is to group images into partitions which require similar parameters settings while at the same time evolving optimal parameter settings for each group. Grouping images into different partitions is done, because one optimal parameter setting for each and every image is not to be expected.

The results in Tables 6.2 and 6.3 show that Fitness Based Partitioning does indeed produce sets of parameter settings which lead to better lumen segmentations when compared to one global optimal solution for all images.

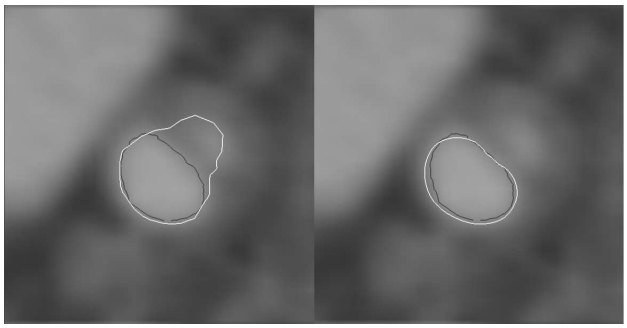


Figure 6.4: Found lumen contours segmented using two different parameters settings. The light gray contour in the right image was found using parameter settings evolved for the partition to which this image was assigned. The light gray contour on the left was found using the parameter settings evolved for the other partition. The dark contour in both images indicates the expert-drawn contour.

Analysis of the final image partitioning results, obtained by running the algorithm with different random seeds, shows that groups of images (but not all) usually end up on the same island. However, there remains some sensitivity to the random seed used.

In the future we want to reduce this sensitivity by using larger populations which cover the search space more completely. This does, however, have a negative impact on computation time. Another option is to make the image re-assignment method more flexible and less "greedy". We intend to extend the Fitness Based Partitioning algorithm with merge and split heuristics to automatically find an optimal number of partitions.

Once the partitions found by the Fitness Based Partition algorithm become more stable we are interested in extracting common features from these images that can act as a kind of image fingerprint, so we can automatically determine which parameter solution to use for a new image.

Part III

Advanced Topics

There is never an end to learning. The dye extracted from the indigo is bluer than the plant; so is the ice colder than the water.

Xunzi

Chapter 7

Metamodel Assisted Mixed Integer Evolution Strategies

So far we have introduced Mixed-Integer Evolution Strategies (MIES) and their applications to parameter optimization for feature detection of a multi-agent medical image analysis system. One of the big challenges is that the evaluation of the fitness function is computationally expensive. This chapter discusses MIES assisted by metamodels, which is based on radial basis function networks (RBFN). The goal is to make MIES more suitable for optimization with time consuming evaluation functions.

A RBFN is an artificial neural network that uses radial basis functions as activation functions. They are quite often used in function approximation, time series prediction, and control. A novelty of our presented research here is that RBFN are studied for metamodeling in heterogeneous (mixed-integer) parameter spaces. A heterogeneous metric (HEOM) is adopted that is in conformity with the design philosophy of the MIES. In addition, cross-validation based optimization techniques are suggested for adjusting hyper-parameters of the model and avoid singularities. Empirical studies on prediction of random sets indicate good prediction capabilities of the proposed RBFN for functional landscapes of moderate dimension/smoothness. The influence of the training set size as well as of the dimension on computational complexity and accuracy of the RBFN is investigated.

In the metamodel-assisted MIES, a RBFN metamodel is built and updated after each generation. The metamodel is used for selecting a small subset of offspring individuals from a bigger set of variations and thereby increase the number of promising solutions in the offspring population. The algorithm is designed in such a way that, in case of failure of the metamodel (e.g. “random" predictions), the metamodel-assisted MIES behaves like a standard MIES. Experimental results, both on artificial test problems and on a real world application, namely the optimization of feature detectors in ultrasound images, indicate that a clear acceleration can be achieved by using heterogeneous RBFN.

7.1 Introduction

As we learned already, MIES are a special instantiation of evolution strategies that can deal with different parameter types (continuous, integer and nominal discrete) simultaneously. In the previous chapters, we already demonstrated that being a promising method, MIES have been successfully applied in optical filter design, the optimization of control parameters of chemical engineering plants and the optimization of multi-agent image interpretation systems for medical image (e.g., Intravascular Ultrasound (IVUS) and Computer Tomographic Angiography (CTA) image) analysis.

However, as it is the case for other evolutionary algorithms, one main challenge in applying MIES to real-world applications is that it needs a large number of fitness evaluations before an acceptable result can be obtained. For instance, for IVUS image lumen detection, one candidate parameter solution must be tested by the hundreds of IVUS images. This is very time consuming and the computation time for one evaluation on a single-processor machine ranges from several minutes up to hours depending on the amount of training data used.

A promising approach for reducing computation time in such cases is to assist the evolutionary algorithms with fast-computable prediction models. Metamodels are data-driven function approximations that are learned from the set of evaluations of a deterministic objective function (or a subset of it). Metamodels are now widely used for function approximation in continuous search spaces [37, 35, 45, 44, 22, 56, 118]. However, their application in discrete search spaces remains sporadic [126], and to our knowledge there are not yet metamodel-assisted evolutionary algorithms for mixed-integer search spaces. This chapter proposes a promising algorithm for the latter problem domain.

In this work we focus on radial-basis function networks [21, 45] and, by using a heterogeneous distance measure, we use these techniques for prediction in mixed-integer search spaces. Radial basis function networks are distance-based predictors, i.e. they compute the prediction based on a weighted approach, where the influence of neighboring points is measured by means of a non-linear distance-based kernel (or activation function). The fact that RBFN are based on relative distances to neighbors rather than on absolute position in Euclidean space makes them suitable to application in metric spaces which are not vector spaces, such as mixed-integer search spaces. A crucial point, however, is still the choice of a metric. In this chapter we choose a heterogeneous metric that takes into account the inherent properties of the parameter types involved (continuous, integer, and discrete).

In literature many ways of how to integrate metamodels in an Evolutionary Algorithm (EA) have been proposed [55]. In this chapter we choose a straightforward approach using metamodels as a filter. The basic idea is to generate a large “pre-population" of offspring individuals and then - based on the predictions of the metamodel - select a small subset of them for precise evaluation. Only the subset of precisely evaluated individuals is considered for replacement. The choice

of population sizes is governed by the idea that, in case of random predictions, the behavior of the metamodel-assisted MIES resembles that of an canonical MIES. However, we assume that in most cases the predictions with the metamodel are better than pure random predictions and therefore the metamodel can help to improve the quality of the sample generated with the randomized search operators.

This chapter is organized as follows. In section 7.2 some classical functional approximation models, such as Polynomial and Kriging models, will be reviewed briefly. Particularly, radial basis function networks will be discussed thoroughly in section 7.3. Next, in section 7.4, the metamodel-assisted mixed-integer evolution strategies are described. The proposed metamodel assisted mixed integer evolution strategies are applied to evaluation on artificial test problems in section 7.5. Then they are applied to the parameter optimization of an IVUS feature detector in section 7.6. Finally, a short summary and future work are presented in the last section.

7.2 Functional Approximation Models

Traditionally, there are two basic approaches which can be applied to approximation in optimization: functional approximation and problem approximation. Here we will discuss functional approximation in detail. About other types of approximation methods, we recommend [56]. In functional approximation, an alternate and explicit expression is constructed for the fitness function. Taking the intravascular ultrasound image analysis as an example, instead of evaluating its performance using a multi-agent feature detection system, an explicit mathematical model can be constructed and used to predict outputs according to given inputs.

7.2.1 Polynomial Models

Polynomial approximation model is widely used and its form can be given as follows:

$$\hat{y} = \beta_0 + \sum_{1 \le i \le n} \beta_i x_i + \sum_{1 \le i \le j \le n} \beta_{n-1+i+j} x_i x_j \tag{7.1}$$

where β_0 and β_i are the coefficients to be estimated, and the number of terms in the quadratic model is $n_t = (n+1)(n+2)/2$ in total, where n is the number of input variables. Least square method (LSM) and gradient method can be used to estimate the unknown coefficients of the polynomial model.

7.2.2 Kriging Model

The Kriging model is another popular approximation model. Kriging was originated by the mining engineer Krige, who used this method to estimate ore concentrations in gold mines. In recent years it has been successfully used in metamodelling and optimization [35, 94]. It can be seen as a combination of a global

model plus a localized “deviation":

$$y(\mathbf{x}) = g(\mathbf{x}) + Z(\mathbf{x}) \tag{7.2}$$

where $g(\mathbf{x})$ is a known function of $\mathbf{x}$ as a global model of the original function, and $Z(\mathbf{x})$ is a Gaussian random function with zero mean and non-zero covariance that represents a localized deviation from the global model. Usually, $g(x)$ is a polynomial and is reduced to a constant β in many cases.

7.2.3 Neural Networks

An Artificial Neural Network (ANN) is defined as a data processing system consisting of a large number of simple, interconnected processing units. The architecture of ANN has been inspired by information processing structures found in the multilayered cerebral cortex of brains. Neural networks have also shown to be effective tools for function approximation. Feedforward multilayer perceptrons (MLP) [50] and radial basis function networks (RBFN) are two well studied models among others. In the next section, we will discuss RBFN in detail.

Feedforward multilayer perceptrons

A feedforward multilayer perceptrons with one input layer, two hidden layers and one output neuron can be described by the following equation:

$$y = \sum_{l=1}^{L} v_l f(\sum_{k=1}^{K} w_{kl}^{(2)} f(\sum_{i=1}^{n} w_{ik}^{(1)} x_i)) \tag{7.3}$$

where n is the input number, K and L are the number of hidden nodes, and $f(\cdot)$ is called activation function, which usually is the logistic function

$$f(z) = \frac{1}{1 + e^{-az}} \tag{7.4}$$

7.3 Radial Basis Function Networks

Radial basis function networks (RBFN) were proposed as artificial neural networks for function interpolation in [21]. They were proposed to assist evolutionary algorithms in [45], and were combined with evolution strategies in [44]. Formally, they are similar to Kriging interpolation techniques [36], though Kriging methods are motivated in a different way. As distance based interpolation function RBFN are suitable for functions, interpolation in metric spaces that not necessarily need to be continuous vector spaces. Also, differentiability of functions is not explicitly required. However, we do assume that the difference in function values is positively correlated with the distance to a given point, and choose the metric accordingly.

Radial basis function networks [21, 45] are three-layer fully connected feedforward networks (cf. Figure 7.1). They perform a nonlinear mapping ($\mathbb{R}^d \rightarrow \mathbb{R}^m$)

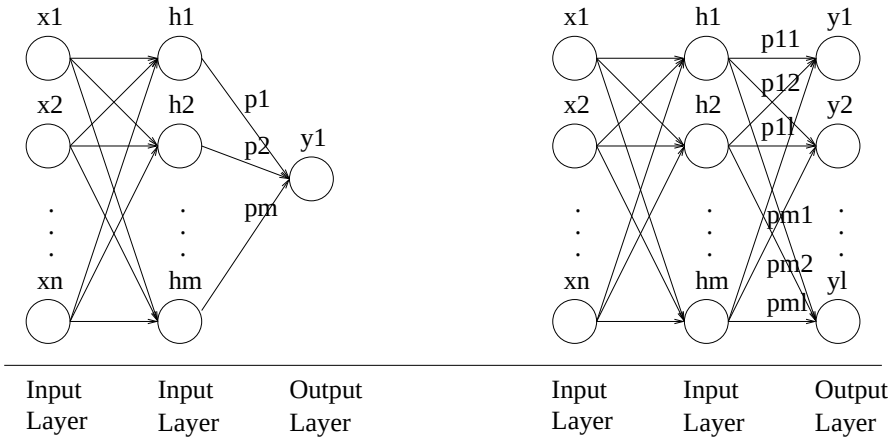


Figure 7.1: Possible structures of Radial Basis Function Network.

from the d inputs to the m hidden units followed by a linear mapping ($\mathbb{R}^m \to \mathbb{R}^l$) from the hidden units to the l outputs. In this chapter only the case of $l = 1$ will be first considered (Figure 7.1).

In our case we will deviate from the standard definition and define radial basis function networks for metric spaces. Let $\mathbb{M}$ denote a metric space with distance measure $\Delta : \mathbb{M} \times \mathbb{M} \to \mathbb{R}_0^+$, then $\mathbb{M} \to \mathbb{R}^m$ denotes a radial basis function network for a general metric space.

When applied for function approximation, the neural network is trained during a *training phase*, with data from known function evaluations. The weights of the linear function from the hidden layer to the output are adapted in a way that the deviations between the known output values to the predicted output values are minimized. Then, in the *prediction phase*, a point $\mathbf{x} \in \mathbb{M}$ is presented to the neural network and the neural network predicts the response.

Giannakoglou et al. [45] introduced a straightforward approach on how to employ RBF networks for function interpolation in the sense that results for points in the training set shall be reproduced exactly. Its architecture is described as follows: Let again $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}$ denote the evaluated points of the database, and $y^{(1)} = y(\mathbf{x}^{(1)}), \ldots, y^{(m)} = y(\mathbf{x}^{(m)})$. Then define for each evaluated point $\mathbf{x}^{(i)}$ a *RBF center*:

$$\mathbf{b}^{(i)} := \mathbf{x}^{(i)}, i = 1, \ldots, m \tag{7.5}$$

Let $r : \mathbb{R}_0^+ \to \mathbb{R}_0^+$ a positive definite function on $\mathbb{R}_0^+$, then we define the activation function of the hidden layer via:

$$h(\mathbf{x}, \mathbf{b}^{(i)}) = r(\Delta(\mathbf{x}, \mathbf{b}^{(i)})), i = 1, \ldots, m \tag{7.6}$$

The activation function based on r is called a *radial basis function* because its value depends on the distance of $\mathbf{x}$ to the RBF center. For $r : \mathbb{R}_0^+ \to \mathbb{R}_0^+$ Giannakoglou suggests the function

$$r_\theta(\Delta(\mathbf{x}, \mathbf{x}')) = \exp(-\theta\Delta(\mathbf{x} - \mathbf{x}')^q), \text{ with } q = 2 \tag{7.7}$$

with a value for θ that, as a default, was set to 1.

The prediction function $\hat{y}$ from the input values to the output value of the RBFN is defined as a linear function with a-priori unknown weights:

$$\hat{y}(h^{(1)}, \ldots, h^{(m)}) = \sum_{i=1}^{m} \psi^{(i)} h(\mathbf{x}, \mathbf{b}^{(i)}) \tag{7.8}$$

The values of $\psi^{(i)}$ need to be adapted in the training phase. The output values of the training points have to be reproduced by the neural network, whenever we demand for exact interpolation of the results. This is expressed by the system of equations:

$$\sum_{i=1}^{m} \psi^{(i)} h(\mathbf{x}^{(j)}, \mathbf{x}^{(i)}) \stackrel{!}{=} y^{(j)}, j = 1, \ldots, n \tag{7.9}$$

Rewritten in matrix form this reads:

$$\underbrace{\begin{bmatrix} h(\mathbf{x}^{(1)}, \mathbf{b}^{(1)}) & \cdots & h(\mathbf{x}^{(1)}, \mathbf{b}^{(m)}) \\ \vdots & \ddots & \vdots \\ h(\mathbf{x}^{(m)}, \mathbf{b}^{(1)}) & \cdots & h(\mathbf{x}^{(m)}, \mathbf{b}^{(m)}) \end{bmatrix}}_{\mathbf{H}} \underbrace{\begin{bmatrix} \psi^{(1)} \\ \vdots \\ \psi^{(n)} \end{bmatrix}}_{\psi} \stackrel{!}{=} \underbrace{\begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix}}_{\mathbf{y}} \tag{7.10}$$

Note that $\mathbf{H}$ is a symmetric $m \times m$ matrix. The symmetry of the matrix $\mathbf{H}$ follows immediately from the equivalence of the RBF centers $\mathbf{b}^{(i)}, i = 1, \ldots, m$ with the input patterns $\mathbf{x}^{(i)}, i = 1, \ldots, m$ and the symmetry of the distance measure.

Assuming that there are no equal points in the database and that the RBF is positive definite, the weights $\psi^{(i)}, i = 1, \ldots, m$ are given by the solution of this system, i.e.

$$\psi = \mathbf{H}^{-1}\mathbf{y} \tag{7.11}$$

After computing the values of the vector ψ (training phase) we can now use them for predicting output values for any $\mathbf{x} \in \mathbb{M}$ by using equation 7.9.

7.3.1 Heterogeneous Metric

A crucial step in adapting the RBFN method for mixed-integer spaces is the choice of an appropriate distance measure. For continuous spaces the Euclidean metric seems to be a straightforward choice, while for nominal discrete spaces an overlap metric seems suitable, as it does not assume any continuity of the objective function w.r.t. to a particular ordering of the domain. For two integer parameter vectors the distance can be measured by means of the Manhattan distance in a straightforward way. This is the accumulated distance when computing the difference of single parameter values of the variables. In combination with the MIES the choice of the Manhattan distance is also in conformity with the mutation operator, who generates samples with a ℓ_1 symmetric distribution. To combine different metrics we adopt the HEOM approach by Wilson and Martinez [122], which suggests to take the square root of the sum of distances of the partial parameter vectors. Let $\Delta_r(\mathbf{r},\mathbf{r}') = \sum_{i=1}^{n_r}(r_i - ri')^2$, $\Delta_z(\mathbf{z}) = \sum_{i=1}^{n_z}|z_i - z_i'|$, and $\Delta_d(\mathbf{d},\mathbf{d}') = \sum_{i=1}^{n_d} I(d_i \neq d_i')$ with $I(true) = 1, I(false) = 0$. Then the combined heterogeneous metric Δ_x for $\mathbf{x} = (\mathbf{r} \circ \mathbf{z} \circ \mathbf{d})$ reads:

$$\Delta_x(\mathbf{x},\mathbf{x}') = \sqrt{\Delta_r(\mathbf{r},\mathbf{r}') + \Delta_z(\mathbf{z},\mathbf{z}') + \Delta_d(\mathbf{d},\mathbf{d}')}. \tag{7.12}$$

In order to improve prediction accuracy, we adapted the parameter θ during the training phase. This was done by means of a global minimization of the cross-validated error using a grid sampling method. The effect of this procedure is significant as our preliminary experiments revealed. In order to adapt the parameter we take compute

$$\theta^* = \arg\min_{\theta \in \{10^{\Theta_{min}},\ldots,10^{\Theta_{max}}\}} \mathrm{LCVE}(\theta) \tag{7.13}$$

with LVCE being the quadratic error of leave one-out-cross-validation for an equidistant set of values $T = \{\Theta_{min}, ..., \Theta_{max}\}$, in the experiments we choose the set $T = \{-4, \ldots, 1\}$. The motivation of using a logarithmic grid based optimization of θ is that we need (1) a fast and (2) a reliable optimization routine for θ. It is due to our experience much more important to hit the right order of magnitude with θ than to fine-tune its value. For much too high values of θ the matrix $\mathbf{H}$ will be close to a unit matrix, and in case of too low θ values it will be a matrix filled with ones. In the latter case the matrix is almost singular (causing problems with matrix inversion). We omit a fine tuning of θ, as, due to the high cost for matrix inversion, this would be very time consuming and the added value is questionable.

The *computational complexity* of the training step is governed rather by the number of samples than by the dimension of the search space. The time complexity of computing the matrix $\mathbf{H}$ scales as $\mathcal{O}(dm^2)$, where d is the number of input variables, whereas the inversion of the matrix scales with $\mathcal{O}(n^3)$ if we use, for example, Gaussian elimination. There are more efficient inversion routines available, such as Strassen's algorithm, but to our knowledge the decreased complexity leads to an effective decrease in computation time only with very high problem

dimensions. Given this the overall complexity $\mathcal{T}_{training}(m, d, T)$ of the training phase reads:

$$\mathcal{T}_{training}(m, d, T) = |T|(m^2 d + m^3) \tag{7.14}$$

In contrast, the time for predicting the output value, given a set of weights ψ is only linear. More precisely it scales with $\mathcal{O}(dm)$

As a conclusion, these considerations show that we should consider the training phase as the main computational effort and the number of samples being the main determinant of the effort in that phase. This observation will govern the decision on how to build the RBFN-MIES in the following chapter.

7.4 Metamodel Assisted MIES

Although MIES were already successfully applied to some real-world application, they usually need a large number of fitness evaluations before an acceptable result can be obtained. To accelerate the MIES it could be interesting to estimate the fitness function by constructing an approximate model. Here we propose metamodel-assisted MIES which use radial basis function networks (RBFN) to predict fitness values. The main loop of the RBFN-MIES is displayed in algorithm 10. Some features distinguish RBFN-MIES from standard MIES:

1. All exactly evaluated individuals are recorded.
2. The metamodel is updated in each generation based on the K^+ latest records from database.
3. The $\lambda^+ (\gg \lambda)$ offspring are created in each generation and evaluated by the metamodel.
4. The best λ individuals are taken for the precise evaluation.

The proposed scheme widely corresponds to the metamodel-assisted evolution strategy (MAES) as proposed by Emmerich et al. [37]. However, there is an important difference: While in the MAES a metamodel is trained for each individual, the RBFN-MIES trains one single metamodel for a whole generation of individuals. This allows to use a larger size of the training population, which, due to our initial studies is of crucial importance to achieve a good prediction quality.

7.5 Study on Artificial Test Problem

7.5.1 Prediction Accuracy Study

We first study the prediction accuracy of the radial basis function networks on the mixed integer domain depending on the dimension of the search space and sample size. As we described in section 7.3, the heterogeneous metric is used to

Algorithm 10 Main loop of RBFN-Assisted MIES

1: $t \leftarrow 0$
2: **Initialize** population P_t of K^+, including μ, individuals randomly generated within the individual space $\mathbb{I}$
3: **Evaluate** the P_t and insert results to database D
4: **while** Termination criteria not fulfilled **do**
5: **Train** RBFN based on K^+ latest evaluations
6: **Generate** the λ^+ offspring
7: **Predict** fitness of λ^+ offspring
8: **Select** the best λ individuals out of λ^+ offspring
9: **Evaluate** λ selected individuals by using original fitness function, and insert results to database D
10: **Select** the μ best individuals for P_{t+1} from λ offspring
11: $t \leftarrow t + 1$
12: **end while**

compute distance for different parameter types. The mixed-integer sphere function (Eq. 7.15) will be used as our test problem.

$$f_{sphere}(\mathbf{r}, \mathbf{z}, \mathbf{d}) = \sum_{i=1}^{n_r} r_i^2 + \sum_{i=1}^{n_z} z_i^2 + \sum_{i=1}^{n_d} d_i^2 \rightarrow \min \tag{7.15}$$

Note that the discrete values are treated as nominal discrete values by the evolution strategy. Therefore it is not possible to exploit the ordering on the integers as it does for the integer variables.

The experiment was set up as follows: Firstly, we generate a certain number of training samples. Each sample consists of three parameter types. The boundary for each parameter type is defined as $r_i \in [0, 10], z_i \in [0, 10], d_i \in \{0, \ldots, 9\}$. These training samples, as well as their precise fitness values, will be used to train a RBFN. Secondly, we use this trained RBFN to make prediction on other randomly generated 1000 test samples. Differences between predicted and precise fitness value on test samples will indicate how good the approximation ability of RBFN is. Figure 7.2 displays results of the RBFN for a mixed-integer sphere problem in different dimensions and for different numbers of training samples. The results indicate that the number of training points is crucial for achieving a metamodel of good quality (from top to down). The dimension (from left to right) of the search space has slightly less impact. However, the results indicate that the accuracy of the prediction decreases clearly when increasing the dimension. With moderate dimensions, however, the approximations are much better than random guesses. These results prove that RBFN can be applicable not only in continuous but also in mixed-integer spaces, if the distance measure is chosen appropriately.

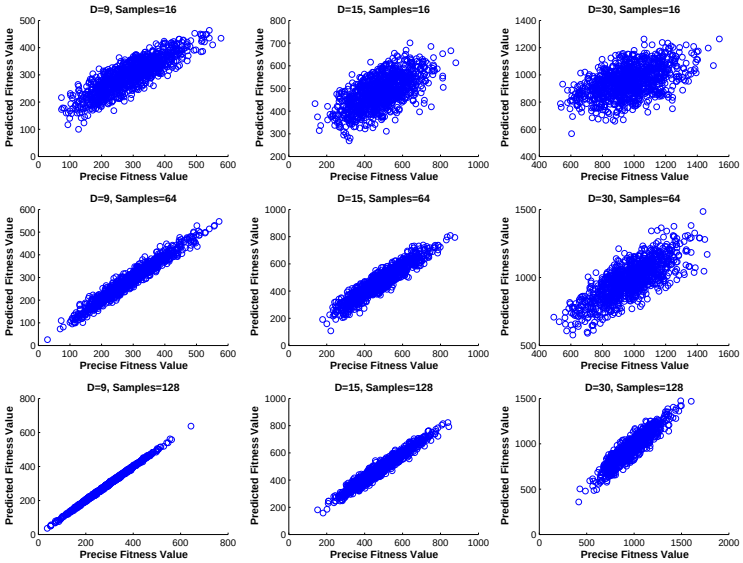


Figure 7.2: Scatter plots for exact fitness value (X-Coordinate) and predicted fitness value (Y-Coordinate) of 1000 test samples by using different training samples on generalized sphere function.

7.5.2 Applying RBFN-MIES to Test Problems

Before we apply RBFN-MIES to a medical image analysis problem, we study its behavior on two artificial test problems. For both cases we use $(\mu = 4, \lambda = 28)$ strategy. We first test RBFN-MIES on the generalized unimodal sphere function, with $n_r = n_z = n_d = 5$ and the same boundary condition as described above. The test result on the sphere function is shown in Figure 7.3. It shows that MIES assisted by RBFN can accelerate the convergence speed compared to the strandard MIES.

Another test problem is the more complex and multimodal barrier problem (cf. Chapter 4). Barrier functions produce mixed-integer optimization problems with a scalable degree of ruggedness (determined by control parameter C) by generating an integer array `A` using Algorithm 8 in chapter 4.

Fig. 7.4 shows experimental results on the 15-D drempels function with control parameter C=20. For the test in Fig. 7.4, we set K^+ of RBFN-MIES to 64. It turns out that, when optimization problems become more difficult, that is more rugged, the training number of the training samples must be increased to ensure a good quality of RBFN predictions. Moreover, on this rugged landscape the MIES without RBFN-assistance detects the global optimum more reliably in the long

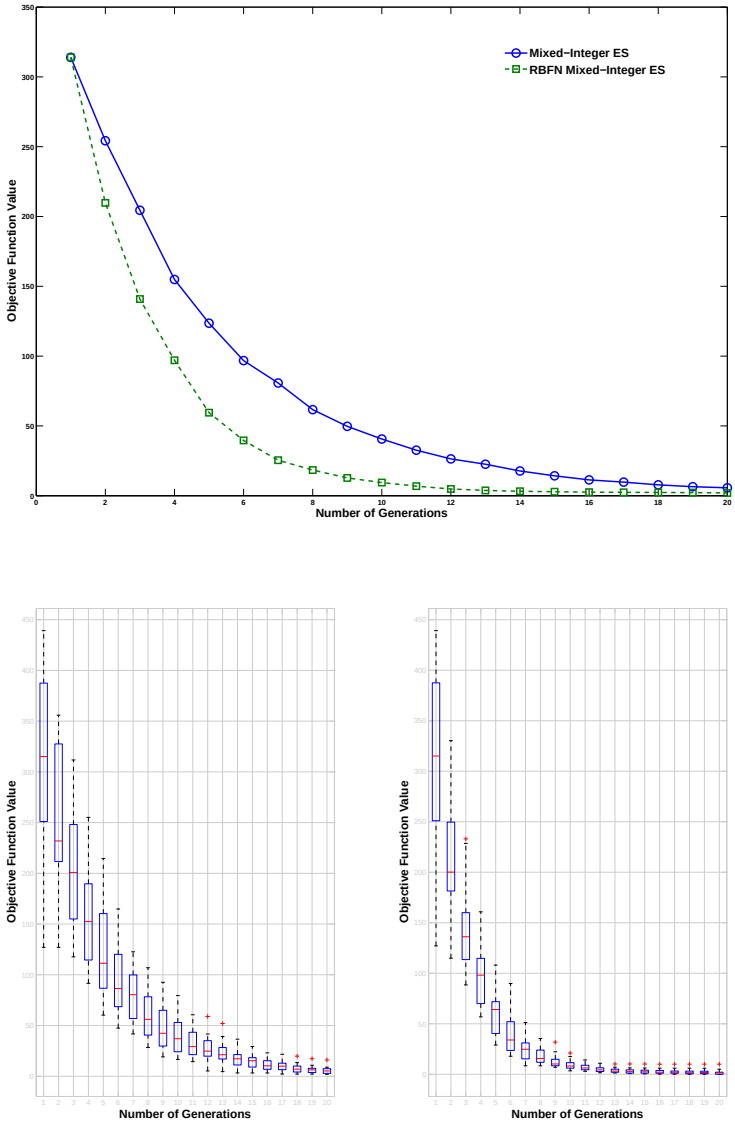


Figure 7.3: Average convergence histories of 20 runs of the 15-D sphere optimization problem with the MIES and RBFN-MIES ($K^+ = 64$). The upper figure shows the average results for both strategies. In the lower figure additional information on outliers and confidence margins are displayed using box error plots for the runs with the MIES (left) and RBFN-assisted MIES (right).

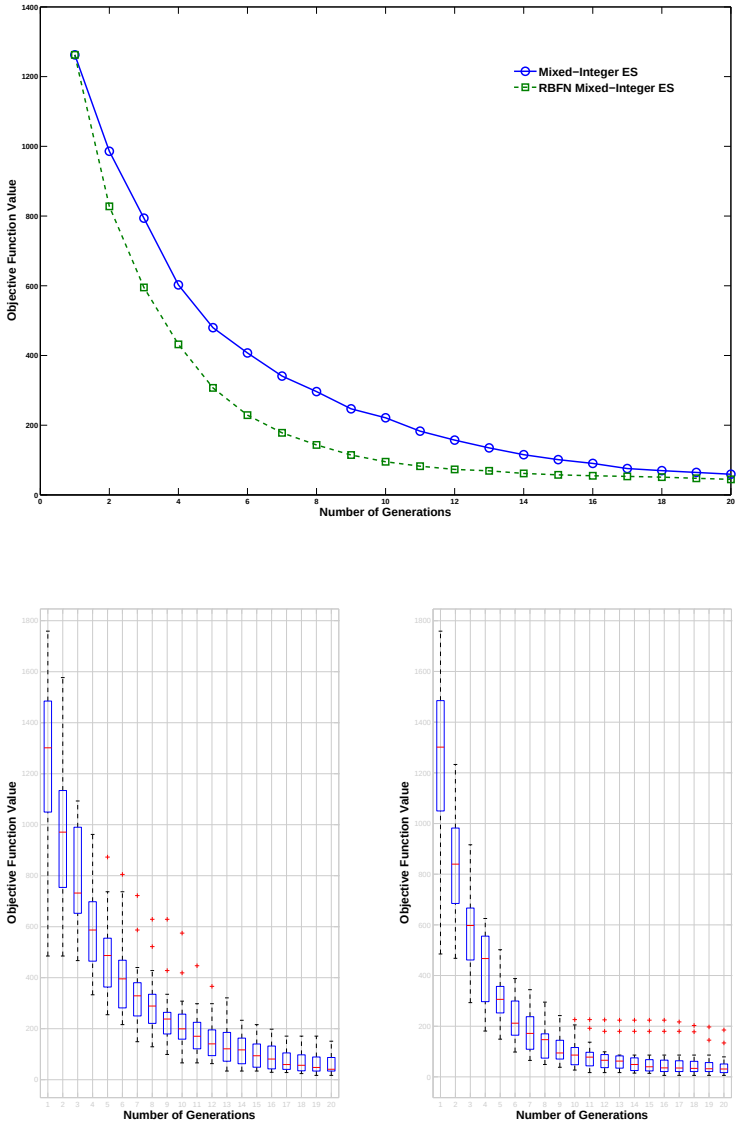


Figure 7.4: Average convergence histories of 20 runs of (RBFN-)MIES ($K^+ = 64$) on 15-D drempels problem with C=20. The upper figure shows the average results for both strategies. In the lower figure additional information on outliers and confidence margins are displayed using box error plots for the runs with the MIES (left) and RBFN-assisted MIES (right).

run. This indicates an increased tendency of convergence towards sub-optimal solutions in case of multimodal optimization. A possible explanation can be that using a prediction makes it more difficult to leave the attractor of a local optimum. Counteracting this problem will be a topic of future research. However, on average, the results obtained with the RBFN-assisted MIES are better and good results are obtained faster.

7.6 Apply RBFN-MIES to IVUS Image Lumen Detection

MIES were already used to find optimal parameter settings for the segmentation of the lumen in IVUS images. In this work, we optimize a new image processing pipeline with 23 optimization variables (as compared to 16 parameters in the previous chapter).

Evaluating an image processing pipeline on given parameter settings is very time consuming. The evaluations of one setting of the MIES algorithm on 100 IVUS images took about 1 minute, i.e. for 10 generations with 4 parents and 28 offspring took about 5 hours on a Pentium 4 (3.4GHz) computer. Therefore a metamodel-assisted approach seems promising as we can make use of existing evaluation results, which could help MIES to accelerate convergence speed and does not considerably increase the total computational time, in particular in cases when training data for the metamodel is already available.

Like we did for the artificial test problems, we use $(\mu = 4, \lambda = 28)$ MIES strategy. We set $K^+ = 32$, that is, the latest 32 precisely evaluated individuals are used to update the RBFN in each generation. We run both standard MIES and RBFN-MIES on 100 IVUS images. The first preliminary experimental result is shown in Fig. 7.5. As we can see from the result, RBFN-MIES slightly accelerate the convergence speed compared to the standard MIES without metamodel assistance. Of course, we can increase the K^+ to make RBFN prediction more precisely. However, as we mentioned during the discussion of the runs of test cases, by doing this we will also need some extra computation time to train RBFN in each generation.

7.7 Summary

In this chapter we propose radial basis function networks assisted mixed integer evolution strategies. To study the behavior of RBFN-MIES, we first applied it to different artificial test problems, and then to parameter optimization of the IVUS image lumen feature detector. The experimental results indicate that by constructing/updating such an approximate model in each generation, acceleration on convergence speed can be achieved, provided training data for the RBFN is available (e.g. from previous runs). Moreover, we showed by scatter plots that

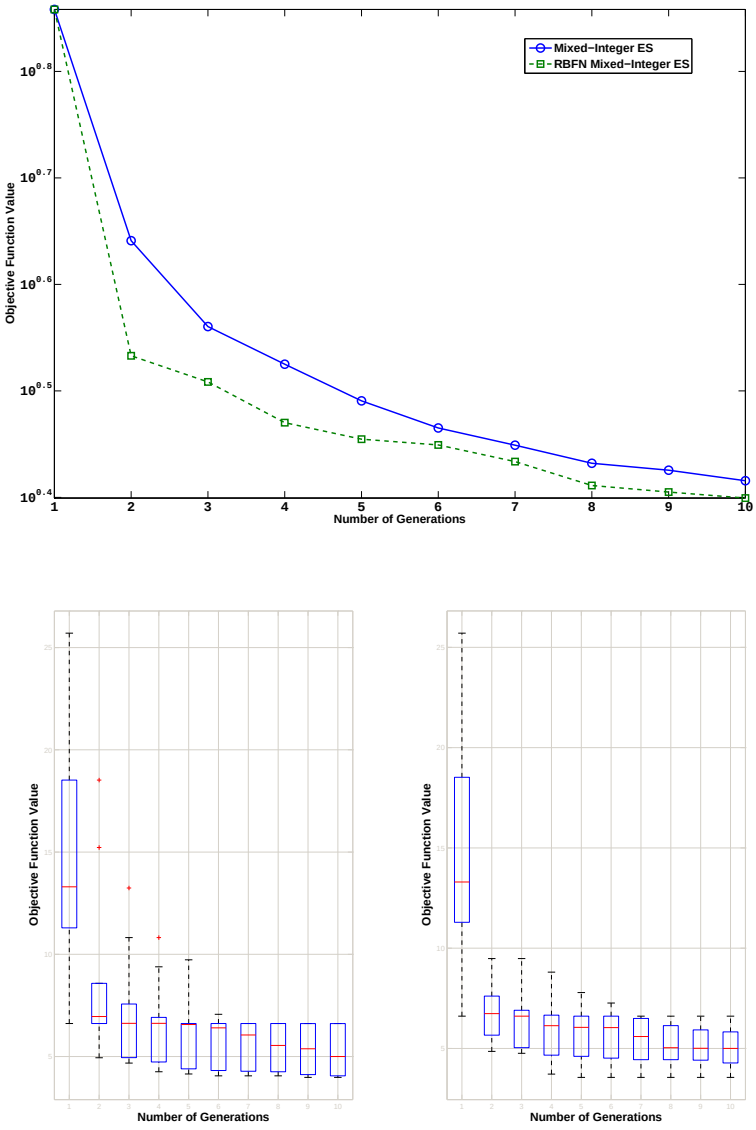


Figure 7.5: Average convergence histories of 10 runs of the IVUS lumen detection optimization problem with (RBFN-)MIES. The upper figure shows the average results for both strategies. In the lower figure additional information on outliers and confidence margins are displayed using box error plots for the runs with the MIES (left) and RBFN-assisted MIES (right).

the predictions of the RBFN on the mixed-integer data are highly correlated to the true function values. When the optimization task difficulty increases, it will be more difficult for the metamodel to make good predictions and the size of the training set needs to be increased. According to our design, in case of failure of the metamodel the RBFN-MIES regresses to a standard MIES.

As mentioned in the previous part, this is the first study on using RBFN-assisted MIES. The results are promising, but still there remain challenges. Firstly, in this chapter we need initial training data for the RBFN in order to achieve a significant acceleration. This data can be taken ("recycled") from previous runs or otherwise needs to be computed causing an increased computational effort. A possible way to generate training data efficiently would be to generate them "on the fly", that is to apply the RBFN in the first generations until enough training data are available. The performance of this 'cold start' strategy needs to be assessed in future work.

Another question that arises is how we can decide whether one precisely evaluated individual should be recorded or not. By discarding points that are very similar to existing points the diversity of the training samples can be increased, which can lead to enhanced quality and numerical stability. In this context, it is also interesting to study problems like overfitting and prevention of deceptive prediction. A promising approach to avoid such effects could be an online monitoring of the model quality. To counteract overfitting also regularization techniques can be considered in future work as well as unsupervised methods like self-organizing maps.

In addition to parameter studies the performance of the RBFN-MIES needs to be tested on a larger number of problems, including further, more challenging, problems from medical image analysis. From a theoretical point of view it will be interesting to access how continuity assumptions, such as Lipschitz continuity, can be generalized for mixed-integer domains and how the quality of the mixed-integer RBFN is related to such properties. A goal of such considerations would be a theory that allows to mark the boundary between functions that can be approximated and those where approximation fails.

Chapter 8

Mixed-Integer Evolution Strategies with Dynamic Niching

Mixed-Integer Evolution Strategies (MIES) are a natural extension of standard Evolution Strategies (ES) for addressing optimization of various types of variables – continuous, ordinal integer, and nominal discrete – at the same time. Like most Evolutionary Algorithms (EAs), they experience problems in obtaining the global optimum in highly multimodal search landscapes. Niching methods, the extension of EAs to multimodal domains, are designed to treat this issue. In this study we present a dynamic niching technique for Mixed-Integer Evolution Strategies, based upon an existing ES niching approach, which was developed recently and successfully applied to continuous landscapes. The new approach is based on the heterogeneous distance measure that addresses search space similarity in a way consistent with the mutation operators of the MIES. We apply the proposed Dynamic Niching MIES framework to a test-bed of artificial landscapes and show the improvement on the global convergence in comparison to the standard MIES algorithm.

8.1 Introduction

Evolutionary Algorithms (EAs) have the tendency to converge to a single solution [3, 83], even if the search landscape has multiple globally optimal solutions. This is due to effects such as genetic drift [104] , fast takeover [3], and disruptive recombination [92]. Population diversity loss in EAs does not only make it difficult to obtain multiple global optima, but may also prevent the algorithm from locating the global optimum.

Niching techniques have been proposed to counteract population diversity loss

in EAs. They support parallel convergence into multiple attraction basins in a multimodal landscape within a single run. Niching techniques have been mainly developed within the framework of Genetic Algorithms (GAs) in the past decades (see, e.g. [112] and [83]), and have recently also received increasing attention from the Evolution Strategies (ES) community [92, 111, 116, 117].

The application of niching in ES proved to be very successful in improving convergence reliability and solution diversity in multimodal continuous optimization. However, it remains an open question, whether niching can also be incorporated into mixed-integer search spaces, which are of great practical relevance [7]. In this chapter we investigate whether niching is also beneficial in this problem domain by combining the niching approach by Shir et al. [111] with the Mixed-Integer Evolution Strategy (MIES) [38, 77].

A crucial step will be the definition of an appropriate metric that is compatible with the neighborhood structures used by the search operators of the Mixed-Integer Evolution Strategies. Thereby we aim for a coherent algorithm design which will make a theoretical analysis of the algorithm more accessible. It is a known drawback that the MIES has difficulties to converge to global optima of highly multimodal landscapes [78]. Based on selected test problems, such as Mixed-Integer NK Landscapes [78] and Barrier Functions [77], we study whether the introduction of niching improves the MIES performance on such landscapes.

8.2 Niching with Evolution Strategies

Niching methods are techniques that originally promote the formation and maintenance of interim subsolutions in the genetic algorithms (GA) on the way to single, final solution [83]. Not only are they necessary if one is interested in finding multiple solutions to a problem of multimodal function optimization and multi-objective function optimization classification, but also they are useful for finding better single solutions to very hard problems. In this section, we will give a brief overview of ES niching techniques with respect to Mahfoud's niching methods.

8.2.1 Motivation

As we addressed in the former section, the canonical ES suffered from several effects - *select pressure*, *operator disruption* and *random genetic drift*, which interrupt the formation and maintenance of multiple solutions [110]. As a result of these effects, the evolution process are pushed towards a rapid convergence into a single solution, even when multiple solutions are required by a given problem.

Selective Pressure

Traditional deterministic selection strategies of the standard ES intuitively implies high selective pressure. A quantitative analysis of selective pressure was given by Goldberg and Deb by introducing the *takeover time* [46] concept, which is defined

as the minimal number of generations until repeated application of the selection operator yields a uniform population filled with copies of the best individual. Bäck analyzed the ES selection mechanisms and showed that both (μ, λ) and $(\mu + \lambda)$ selection strategies have very short takeover times (or high selective pressure).

Operator Disruption

In the standard ES, the mutation operator can be regarded as an operator with negligible disruption effect, while the recombination operator, by contrast, has a disruptive nature and modifies a coordinate of the decision parameters to be optimized.

Genetic Drift

Genetic drift is a stochastic process in which the diversity is lost in finite populations [64]. Due to the finite number of offspring, a distribution of genetic properties is transferred to the next generation in a very limited manner and consequently the distribution will approach an equilibrium distribution. Since small population sizes are used in the standard ES, the genetic drift occurs and causes the loss of diversity within the population. Especially in multimodal functions, such an effect causes a convergence to an equilibrium distribution around a single attractor [104].

8.2.2 Dynamic ES Niching

In the following part, we will describe the framework for applying niching techniques in the standard ES, which was originally proposed by Shir in [110, 111]. In particular, distance metric d_{x_i,x_j} and niche radius ρ will be discussed.

Distance Metric

Originally, the canonical ES was developed for tackling problems in real-valued searching space. The metric for measuring the distance between individuals is defined as follows: given two individuals in the search space with dimension n, $\vec{x_i} = [x_{1,i}, x_{2,i}, \ldots, x_{n,i}]$ and $\vec{x_j} = [x_{1,j}, x_{2,j}, \ldots, x_{n,j}]$, the distance d_{x_i,x_j} is calculated using a *euclidean distance* norm given in Equation 8.1 below.

$$d_{x_i,x_j} = \sqrt{\sum_{k=1}^{n}(x_{k,i} - x_{k,j})^2} \tag{8.1}$$

The Niche Radius

The original formula for calculating niche radius ρ in GA was derived by Deb and Goldberg in [29]. It is straightforward to adopt the formula but using the distance metric which was defined by equation 8.1.

Given the number of peaks q in the solution space, every niche is considered to be surrounded by a n-dimensional hypersphere with radius ρ which occupies $\frac{1}{q}$ of the entire volume V of the space. The volume V can be computed through formula below:

$$V = cr^n$$

where c is a constant and given explicitly by:

$$c = \frac{\pi^{\frac{n}{2}}}{\Gamma(\frac{n}{2}+1)}, \qquad \Gamma(n) = \int_0^\infty x^{n-1}\exp(-x)dx$$

Given the lower and upper boundary values $x_{k,min}$, $x_{k,max}$ in the decision parameter space, r is defined as follows:

$$r = \frac{1}{2}\sqrt{\sum_{k=1}^{n}(x_{k,max} - x_{k,min})^2}$$

If we divide the volume into q parts, we can get following formula:

$$c\rho^n = \frac{1}{q}cr^n$$

which yields

$$\rho = \frac{r}{\sqrt[n]{q}} \tag{8.2}$$

Dynamic Niching ES Algorithm

Next, we outline and discuss the Dynamic Niching ES Algorithm [109] in detail. The algorithm starts with the initialization of q niches with μ individuals and their evaluation. Then, the following loop is repeated until a termination criterion is met: Firstly, for each niche the algorithm generates λ offspring based on the μ parents. Depending on the instantiation of the algorithmic ES kernel, mutation and recombination operators are employed for this purpose.

By restricting recombination to the dynamically updated niches, the algorithm enforces a mating restriction scheme which allows competitive mating only within the niches. This is done to prevent disruptive effects of the recombination operator [92]. The concept of fixed mating resources is strictly enforced: For every niche the same number of offspring is generated, also referred to as the *niche hosting capacity*. This measure is taken in order to prevent genetic drift effects, as described e.g. in [104].

Upon fitness evaluation of the new individuals, offspring and parent individuals are merged into one population comprising now $q \times (\mu + \lambda)$ individuals. The algorithm then employs a sub-routine for dynamically identifying the various fitness-peaks of every generation (which uniquely define the niches) and then assigns each individual to a niche. The classification into niches is carried out in

a *greedy* manner, by means of the so-called Dynamic Peak Identification (DPI) algorithm [85]. The latter is outlined as Algorithm 11.

Besides the global selection phase taking place in the niche forming process, which will be described later, a local environmental selection takes place within each niche, that enables step-size adaptation to the local topography of the niches. If the number of individuals in a peak set is less than μ, the algorithm creates new samples in the search space and adds them to the niche until it contains μ individuals. A summary of the algorithm is given in Algorithm 12.

Algorithm 11 Dynamic Peak Identification (DPI)

in: Population Pop, # niches q, niche radius ρ
out: Peak sets DPS

1: Sort Pop in decreasing fitness order
2: $i := 1$
3: $NumPeaks := 0$
4: $DPS := \varnothing$ {Set of peak elements in population}
5: **while** $NumPeaks \neq q$ and $i \leq popSize$ **do**
6: **if** $Pop[i]$ is not within sphere of radius ρ around peak in DPS **then**
7: $DPS := DPS \cup \{Pop[i]\}$
8: $NumPeaks := NumPeaks + 1$
9: **end if**
10: $i := i + 1$
11: **end while**

The number of expected niches, q, is given as input to the algorithm. The distance calculation is implemented with the Euclidean metric (Equation 8.1) in the decision parameter space since all parameters are continuous. The niche radius ρ itself is approximated a-priori with Equation 8.2 and remains fixed during the run.

8.3 Dynamic Niching for Mixed-Integer ES

To incorporate MIES into the Dynamic Niching ES framework we must define a proper distance metric for the mixed-integer space. For continuous spaces the Euclidean metric seems to be a straightforward choice, while for nominal discrete spaces an overlap metric seems suitable, as it does not assume any continuity of the objective function w.r.t. a particular ordering of the domain. For two integer parameter vectors the distance can be measured by means of the Manhattan distance in a straightforward way. This is the accumulated distance when computing the difference of single parameter values of the variables. In combination with the MIES the choice of the Manhattan distance is also in conformity with the symmetry assumptions used in the design of the mutation operator, which generates samples from an ℓ_1 symmetric distribution. We combine the different metrics using

Algorithm 12 Niching-ES.

in: Number of niches q, Niche radius ρ
out: Optimized solution(s)

```
Initialize q equally-sized niches of size μ randomly
Evaluate all new individuals in all niches
while Termination criteria not full filled do
    for every niche i = 1...q do
        generate λ offspring from μ parents
        Evaluate fitness of λ offspring individuals
        Update best found solution(s)
    end for
    Combine all μ + λ individuals from niches into one population
    Compute the Dynamic Peak Set with DPI (Algo. 11)
    Select μ best individuals per niche
    for every niche i = 1...q do
        if μ_i = number of individuals in niche i < μ then
            Generate and Evaluate μ − μ_i new individuals
        end if
    end for
end while
```

the Heterogeneous Euclidean-Manhattan-Overlap Metric (HEMOM) approach by Wilson and Martinez [122]. According to the parameter type (cf. the search space definition in section 3.3.1), the specific distance metric can be used to compute distance. More specifically, we summed these different distance metrics up according to Equation 8.3 below:

$$\Delta(x_i, x_i') = \begin{cases} \Delta_r(x_i, x_i') = (x_i - x_i')^2 & \text{if } x_i, x_i' \in \mathbb{R}; \\ \Delta_z(x_i, x_i') = |x_i - x_i'| & \text{if } x_i, x_i' \in \mathbb{Z}; \\ \Delta_d(x_i, x_i') = \mathbf{I}(x_i, x_i') = \begin{cases} 1 & \text{if } x_i \neq x_i' \\ 0 & \text{if } x_i = x_i' \end{cases} & \text{if } x_i, x_i' \in \mathbb{D}. \end{cases} \tag{8.3}$$

Then the combined heterogeneous distance metric Δ_{mixed} for $\mathbf{h} = (\mathbf{r} \circ \mathbf{z} \circ \mathbf{d})$ reads:

$$\Delta_{\text{mixed}}(\mathbf{h}, \mathbf{h}') = \sqrt{\Delta_r(\mathbf{r}, \mathbf{r}') + \Delta_z(\mathbf{z}, \mathbf{z}') + \Delta_d(\mathbf{d}, \mathbf{d}')}. \tag{8.4}$$

By using the aforementioned heterogeneous distance metric, the niche radius ρ_{mixed} in the mixed-integer search space now can be approximated as follows:

$$\rho_{\text{mixed}} = \frac{r}{\sqrt[n]{q}} \quad \text{with} \quad r = \frac{1}{2}\sqrt{\sum_{i=1}^{n} (\max \Delta x_i)} \tag{8.5}$$

Here $\max \Delta x_i$ denotes the maximum distance value of parameter x_i within its possible boundary. For $x_i \in \mathbf{D}$, the maximum distance is always 1 according to the definition of overlap metric. In practice, one parameter can overpower the other parameter because of different range. To avoid this, distances are often normalized relative to their acceptable range values. For different normalization techniques, please refer to [122]. q denotes the number of peaks in the solution space. We assumed that every niche with radius ρ_{mixed} occupies $\frac{1}{q}$-th of the entire volume of the space.

8.4 Test Functions and Experimental Results

To investigate the behavior of our algorithm, we applied it to two carefully designed mixed-integer multimodal functions in various dimensions. Specifically, we are interested in the global convergence. Performance comparison between Dynamic Niching MIES with standard MIES is also presented.

8.4.1 Barrier Function

Barrier functions, introduced in chapter 4, create mixed-integer optimization problems with a scalable degree of ruggedness (determined by parameter C). To test the Dynamic Niching MIES and standard MIES algorithm we generated barrier functions for $C = 20$, $C = 200$, $C = 2000$ and $C = 5000$ and ran both the Dynamic Niching MIES and a standard MIES algorithms 20 times with different random seeds. For the Dynamic Niching MIES we used 5 niches with $\mu = 15$ and $\lambda = 75$ for each niche. For the MIES algorithm we used a $(75 + 500)$ strategy thereby making sure that the number of parents, offspring and fitness evaluations per generation is the same for both algorithms.

The results of the experiments are displayed in Figure 8.1. Although the Dynamic Niching MIES converges a little slower than the standard MIES algorithm it does reach the same performance in the end. In the case of C=2000 Dynamic Niching MIES performs slightly better than the standard MIES on average. The possible explanation is that the barrier function landscape with C=2000 is harder than others. The standard MIES converges faster but Dynamic Niching MIES has a better chance of getting rid of local traps at last.

8.4.2 Mixed-Integer NK Landscapes

NK landscapes (NKL, also referred to as NK fitness landscapes), introduced by Kauffman [61], were devised to explore the way that epistasis controls the 'ruggedness' of an adaptive landscape. They are particularly used as test problem generators for Genetic Algorithms (GAs) to understand the dynamics of evolutionary search. The ruggedness and the degree of interaction between variables of NKL can be easily controlled by two tunable parameters: the number of genes N and the number of epistatic links of each gene to other genes K. Moreover, for given

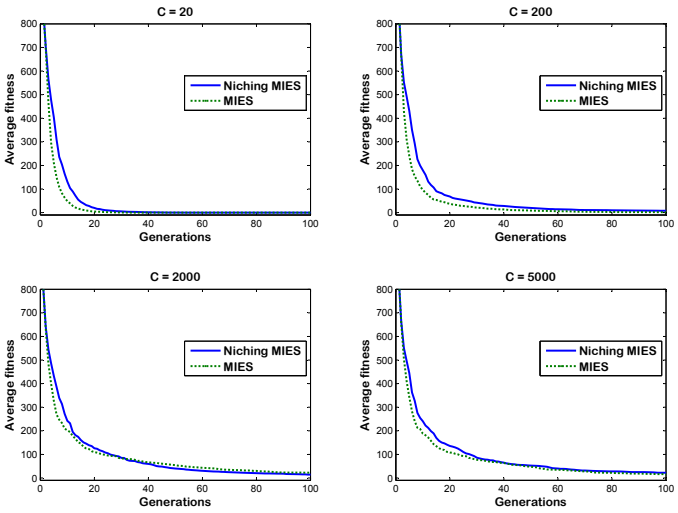


Figure 8.1: Average best fitness results over 20 experiments for barrier functions with $C = 20$, $C = 200$, $C = 2000$ and $C = 5000$ for both the Dynamic Niching MIES and standard MIES algorithms.

values of N and K, a large number of NK landscapes can be created at random. Mixed-Integer NK-Landscapes (MI-NKL) were introduced in chapter 4 and are an extension of NKL from the traditional binary case to a mixed variable case with continuous, nominal discrete, and integer variables. The resulting test function generator is a suitable test model for our dynamic niching Mixed-Integer Evolution Strategy.

In order to test our Dynamic Niching MIES algorithm we tested it on different Mixed-Integer NK landscapes with 15 variables (5 continuous (range $[-10, 10]$), 5 integer variables (also range $[-10, 10]$) and 5 nominal discrete variables (Boolean ($\{0, 1\}$)). We generated 10 random MI-NKL for different levels of K (2, 5, 10, and 14) to simulate different problem difficulties and both the Dynamic Niching MIES and standard MIES algorithms were run 20 times on each MI-NKL using different random seeds. We used a total population size of 75 for both the standard MIES and Dynamic Niching MIES algorithm (15 individuals per niche) and an offspring size of 500 (100 per niche). To compare (and average) the results of the different experiments we used the following error-measure:

$$\text{error} = \text{best found fitness - best possible fitness}$$

The results of the experiments are displayed in Figure 8.2. For $K = 2$ and $K = 5$ we see, similar to the results of the barrier functions, that the standard

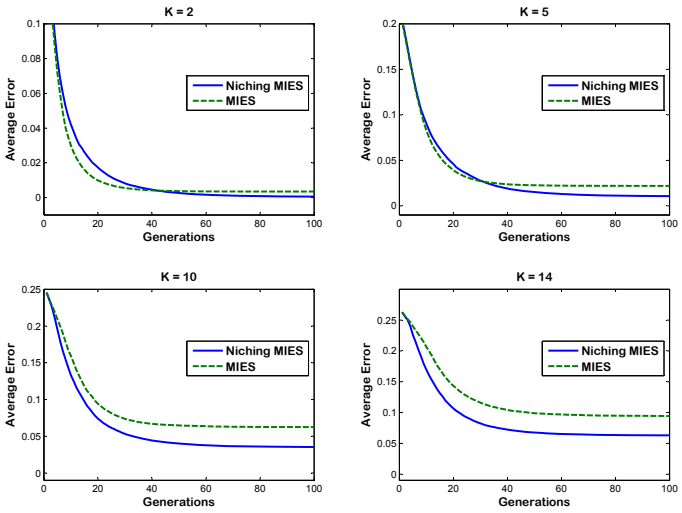


Figure 8.2: The error average of both Dynamic Niching MIES and standard MIES on different mixed-integer NK landscape problems with N = 15.

MIES algorithm converges faster. However, on the MI-NKL the Dynamic Niching MIES algorithm manages to achieve a better result on average. If we look at the results for more rugged (and harder) MI-NKL with $K = 10$ and $K = 14$ we see that the Dynamic Niching MIES outperforms the standard MIES algorithm both in convergence speed and final solution quality.

We also compared the number of experiments the Dynamic Niching MIES and MIES algorithms find the global optimum, and the results are presented in Table 8.1. For $K = 2$ the Dynamic Niching MIES algorithm finds the optimum 174 times out of 200 (10 different MI-NKL times 20 runs) while MIES finds it 143 times. As K increases both algorithms find the optimum less often, which is expected since the difficulty increases. For $K = 5$, 10 and 14 the Dynamic Niching MIES finds the optimum 92, 19 and 8 times respectively. MIES only manages to finds the optimum 67, 6 and 3 times for $K = 5$, 10 and 14. Thus, the Dynamic Niching MIES algorithm does not only result in a lower average error but also manages to find the global optimum more often.

8.5 Summary

Studies on artificial landscapes reveal that the proposed heterogeneous niching can be a useful ingredient in highly rugged landscapes. On MI-NK Landscapes it

K	Dynamic Niching MIES	MIES
2	174	143
5	92	67
10	19	6
14	8	3

Table 8.1: The number of times the Dynamic Niching MIES and MIES algorithms found the global optimum out of a total of 200 experiments (10 different MI-NKL times 20 runs).

clearly improves the chances to obtain the global optimum. In more simple landscapes it only slightly slows down the convergence speed compared with standard MIES. In conclusion, it can be said that in case of simple problems the usage of the new strategy will not be harmful and in the case of highly rugged problems it can lead to solutions of better quality than standard MIES.

In the future the Dynamic Niching MIES should be tested on additional problems, including real-world applications. Moreover, a deepened understanding of niche formation process in mixed-integer landscapes and the influence of strategy parameters may help to further improve its performance.

Chapter 9

Mixed-Integer Evolution Strategies with Bayesian Networks

As we learned from the previous sections of this thesis, mixed-integer optimization problems arise in various application fields, such as chemical engineering and the medical image processing. Stochastic optimization algorithms, such as evolution strategies and estimation of distribution algorithms, can be used as solution methods for solving these problems approximately. Especially for real-world problems they often prove to be powerful methods due to their flexibility and robustness.

However, a shortcoming of existing mixed-integer evolutionary algorithms, such as Mixed-Integer Evolution Strategies (MIES), is that their variation procedures mutate each decision variable *independently*. Therefore, dependencies between variables, even if they are known a-priori, cannot be taken into account. This chapter aims at designing and testing a mixed integer evolutionary algorithm that can utilize knowledge about such dependencies. The development of the new approach is motivated by problems in medical image analysis where the parameters of a medical image processing pipeline are to be optimized (cf. chapter 5). Though the optimization of these systems is essentially a black-box optimization problem, dependence information can be extracted heuristically from the known structure of the processing pipeline (Figure 5.5 in chapter 5).

Inspired by existing works, we propose a Mixed-Integer Bayesian Optimization Algorithm (MIBOA), that is a variant of Estimation of Distribution Algorithms (EDAs) and extends the Bayesian Optimization Algorithm (BOA[1]), from binary optimization problems to mixed-integer optimization problems using special types of Bayesian Networks dealing with random variables of mixed-type. EDAs do neither have a crossover nor a mutation operator. Instead, a new population is gen-

[1] With fixed network structure.

erated by sampling the probability distribution, which is estimated and updated based on the distribution of recently obtained "*successful*" individuals. Different instantiations of EDAs differ by the distribution types and update rules they use. For instance, the classical Population-Based Incremental Learning (PBIL) algorithm samples from an independent joint distribution of Bernoulli type [9], while the Univariate Marginal Distribution Algorithm (UMDA) [69, 108] features independent joint distributions of Gaussian type.

We show that a-priori knowledge on dependencies between decision variables can be exploited by this algorithm in order to improve convergence speed and reliability. In discussing the properties of heterogeneous Bayesian Networks, representing multivariate distributions of mixed-variable type, we point out which kind of dependence information can be utilized. Moreover, a special type of mixed-integer NK-landscape (cf. chapter 4) that is well suited for testing the new approach, the so-called Acyclic Directed Graphic Models (ADG) based NK-landscape, will be introduced.

The chapter is structured as follows: Section 9.1 introduces the basic knowledge of graph theory and Bayesian Networks. In Section 9.3 we discuss briefly estimation of distribution algorithms with independent sampling distributions in contrast to canonical evolution strategies (ES). Section 9.4 introduces Bayesian optimization and generalizes it to the mixed-integer case. After introducing test problems based on NK-landscapes in Section 9.5, we present results of mixed-integer BOA on these landscapes. Finally, the main results of the chapter are summarized and directions of future research are discussed.

9.1 Learning with Bayesian Networks

In this section, we will provide a short introduction to Bayesian Networks, especially parameter learning with Bayesian Networks.

9.1.1 Graphical Models

"*Graphical models are a marriage between probability theory and graph theory. They provide a natural tool for dealing with two problems that occur throughout applied mathematics and engineering – uncertainty and complexity – and in particular they are playing an increasingly important role in the design and analysis of machine learning algorithms*" [59].

In general, there are two main kinds of graphical models (see Figure 9.1 below): *Undirected* graphical models and *Directed* graphical models. In this work, we focus our attention on acyclic directed graphical models, which are very popular within the Artificial Intelligence (AI) and statistics communities .

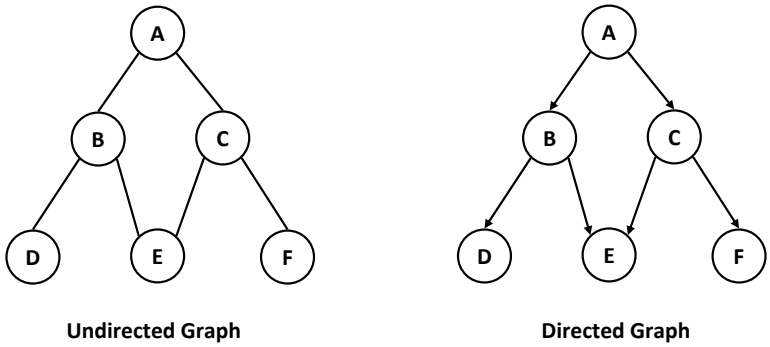


Figure 9.1: A sample undirected graphical model (left) and a sample directed graphical model (right).

9.1.2 Bayesian Networks

In a nutshell, a Bayesian Network is an acyclic directed graphical model that encodes probabilistic relationships among variables of interest, in which nodes represent random variables and the arcs denote conditional independency assumptions. That is, for a given ADG, an arc from node A to B can be interpreted as A "*causes*" B. Related to our case, nodes represent control parameters in the given image processing pipeline. Combined with statistical techniques, Bayesian Networks (BNs) have several advantages for data analysis in contrast to other data representations, such as decision trees and rule bases [51]:

(1) BNs can handle incomplete data sets, because the model encodes dependencies among all variables.

(2) BNs allow us to learn about *causal* relationships, and hence is very useful for us to gain understanding about a problem domain.

(3) In conjunction with Bayesian statistical techniques, BNs facilitate the combination of domain knowledge and data.

(4) BNs are a promising approach to avoid the overfitting of data.

Because of these aforementioned advantages, Bayesian Networks are applied to many real-world applications from different research domains. For instance, bioinformatics, document classification and decision support systems. In this work, Bayesian Networks will be used to model knowledge in medical domain. Specifically, we will use Bayesian Networks to encode probabilistic relationships among decision variables of a medical image feature detector (cf. "image processing pipeline for lumen feature detection" in chapter 5). For more detailed information about graph theory and Bayesian Networks, readers are suggested to check literature such as [51] and [59].

9.1.3 Bayesian Parameter Learning

Through BNs, we can learn the structure (topology) of the model, or the parameters, or learn both of them. The table 9.1 gives a clear view about how to classify learning methods in different situations. Referred to the optimization of the image

	Observability	
Structure	Full	Partial
Known	Closed form	Expectation Maximization (EM)
Unknown	Local search	Structure expectation maximization

Table 9.1: Contigency table for classifying learning methods

processing pipeline (Figure 5.5 in chapter 5), it is clear that the structure of the pipeline is known to us in advance and all nodes are observable. The learning in this case is to find the maximum likelihood estimates (MLEs) of the parameters of each conditional probability distribution (CPD), which contains M cases and are assumed to be independent. The normalized log-likelihood of the training set $D = \{D_1, \ldots, D_M\}$ is a sum of terms, one for each node:

$$L = \frac{1}{M} log \prod_{m=1}^{M} Pr(D_m|G) = \frac{1}{M} \sum_{i=1}^{n} \sum_{m=1}^{M} logP(X_i|P_a(X_i), D_m)$$

where $P_a(X_i)$ are the parents of X_i. The log-likelihood scoring function *decomposes* according to the structure of the ADG; hence we can maximize the contribution to the log-likelihood of each node independently.

9.2 Problem Definition of Mixed-Integer Optimization

Now, let's review the definition of mixed-integer optimization. In this contribution we define the mixed-integer optimization as follows:

$$\text{minimize } f(\mathbf{r}, \mathbf{z}, \mathbf{d}), \mathbf{r} \in \mathbb{R}^l, \mathbf{z} \in \mathbb{Z}^m, \mathbf{d} \in D_1 \times \ldots D_n \tag{9.1}$$

Here, $\mathbf{r}$ denotes a vector of real numbers, $\mathbf{z}$ is defined from a finite set of integer values (or ordinal discrete values), whereas $\mathbf{d}$ defines a n-tuple of nominal discrete variables with finite domains $D_i, i = 1, \ldots, n$. The function f is considered to be a black-box function, or, more precisely, a function the mathematical structure of which is mainly unknown to the user. The only a-priori knowledge that we can exploit about f are assumptions about parameter dependencies (interaction of variables). A common feature of functions in which interactions occur is that they cannot be decomposed into a sum of functions depending only on single variables

(separable function). For example, if r_1 interacts with z_1 and all other parameters are independent from each other, we can write the function as:

$$\begin{aligned} f(\mathbf{r},\mathbf{z},\mathbf{d}) \quad \equiv \quad & f_{1,l+1}(r_1,z_1) + f_2(r_2) + \cdots + f_l(r_l) + f_{l+2}(z_2) + \ldots \\ & + f_{l+m}(z_m) + f_{l+m+1}(d_1) + \cdots + f_{l+m+n}(d_n) \end{aligned}$$

where $f_{1,l+1}(r_1,z_1)$ cannot be written as a sum of functions of r_1 and z_1. Non-separability makes it potentially difficult to optimize these functions by optimization routines that exploit such an assumption, such as coordinate search but also evolutionary algorithms that mutate variables independently from each other. In Section 9.5, with the ADG-based NK-landscapes, an example for a function class in which various variable interactions can be introduced will be discussed.

9.3 Algorithms with independent sampling distributions

Next, let us introduce the *evolution strategy* (ES) and the *estimation of distribution algorithm* (EDA) as two basic evolutionary algorithms for parameter optimization[2]: The *canonical* $(\mu + \lambda)$ *evolution strategy* has the following iteration scheme:

Step 1 : Create initial population $P \leftarrow \{(a_1, \varsigma_1), \ldots, (a_\mu.\varsigma_\mu)\}$, where ς_i denotes a vector of dispersion parameters of the mutation distribution, e.g. standard deviations or mutation probabilities.

Step 2 : Create offspring population Q of size λ by choosing randomly elements from P and mutating first the distribution parameters ς_i to ς_i' and then the object variables a_i using distribution parameters ς_i'.

Step 3 : Set P to the μ best points (with respect to f) coupled with their mutated distribution parameters ς' out of $P \cup Q$.

Step 4 : If termination criterion is reached, return best found solution, otherwise go to Step 2.

In contrast to this, *estimation of distribution algorithms* apply the following main loop:

Step 1 : Initialize distribution parameters of distribution $\mathcal{D}_\theta$.

Step 2 : Create offspring population Q of size λ by sampling from the distribution $\mathcal{D}_\theta$.

Step 3 : Set P to the μ best points in Q with respect to f.

[2]The ES is introduced, as it is a state-of-the-art technique in mixed integer optimization we will compare to later.

Step 4 : Update parameters θ of the distribution $\mathcal{D}_\theta$ as a weighted average of the estimation of θ based on P and the current parameter set θ.

Step 5 : If termination criterion is reached, then return best found solution, otherwise go to Step 2.

While in ES the basic variation operator is *mutation*, the variation operator in EDA is *sampling* from a multivariate distribution the parameters of which are dynamically updated based on positive examples.

Next, let us describe the mutation and sampling procedure for the mixed-integer case (without parameter dependencies).

The mutation of mixed-integer evolution strategies can be described as a procedure:

Continuous mutation: Set $r_i = r_i + \texttt{Normal}(0, s_r)$, $i = 1, \ldots, l$.

Integer mutation: Set $z_i = z_i + \texttt{Geometric}(0, s_z) - \texttt{Geometric}(0, s_z), i = 1, \ldots, l$.

Nominal discrete mutation: If $\texttt{Uniform}(0, 1) < p_d$ set d_i to a random value from $D_i - \{d_i\}$.

Here $\texttt{Normal}(0, s_r)$ computes a normally distributed random number with standard deviation parameter s_r, $\texttt{Geometric}(0, s_z)$ generates geometrically distributed random variables with mean s_z [77], while $\texttt{Uniform}(0, 1)$ generates a uniformly distributed random number between 0 and 1. Before the mutation of the distribution parameter s_r we employ the log-normal distribution as proposed by Schwefel [107] et al. $s_r \leftarrow s_r \exp(\tau_r \texttt{Normal}(0, 1))$ with $\tau_r = 1/\sqrt{l}$ being the learning rate. Accordingly, $s_z \leftarrow s_z \exp(\tau_z \texttt{Normal}(0, 1))$, with $\tau_z = 1/\sqrt{m}$ is used to adapt the step-size for integer mutations. The probability parameter p_d is mutated based on a logistic mutation (see e.g., [105] et al.) that ensures that the value of p_d stays in $]0, 1[$. All three mutations of strategy parameters have the property that increments of the value are as likely as decrements. The ES discussed here is termed mixed-integer evolution strategy and was discussed in several publications [38, 77].

For the *sampling* in the mixed-integer estimation of distribution algorithm similar distribution types are used. We employ the joint distribution $\mathcal{D}_\theta$ composed of

- a vector of l independent multivariate normal distributions, with mean values $\mu_1, \ldots, \mu_l$ and standard deviations $\sigma_1, \ldots, \sigma_l$.
- a vector of m random variables of type $\xi_i + Z_1(s_z) - Z_2(s_z)$, whereas $Z_1(s_z)$ and $Z_2(s_z)$ denote indentically independent geometrically distributed random variables with mean value s_z.
- a vector of n Bernoulli distributed binary random variables with probability parameters $p_1, \ldots, p_n$.

The described estimation of distribution algorithm is new for the mixed-integer search space. However, for binary nominal discrete parameters the algorithm is the classical population based incremental learning (PBIL) algorithm [9] and, reduced to its continuous part, it equals the so-called Univariate Marginal Distribution Algorithm (UMDA) [108, 69]. In the sequel, we will refer to the EDA algorithm for mixed-integer search space as MIPBIL.

The aforementioned two algorithms are used as reference algorithms to find out whether the introduction of dependency information improves the algorithms behavior or not. Next, we will look at an extension of MIPBIL that allows to integrate dependency information.

9.4 Mixed-Integer Bayesian Optimization Algorithm

In order to design a new mixed-integer estimation of distribution algorithm that can take into account dependencies between variables of the objective functions we will replace the independent joint distribution $\mathcal{D}_\theta$ used in the MIPBIL approach by an heterogeneous Bayesian Network with fixed structure. This approach is also used in the Bayesian optimization algorithm (BOA) by Pelikan et al. [91]. Their BOA method is applied for binary search spaces and also learns the structure of the network, while our approach is defined for mixed-integer search spaces and requires a-priori knowlege on the dependency structure of variables in the objective function. To emphasize the similarity to the BOA algorithm, we will term the new approach Mixed-Integer BOA (MIBOA).

Bayesian Networks yield very powerful probabilistic graphical representations. The key to their popularity is their ease of representation of independency relations, and their support for reasoning with uncertainty.

A Bayesian Network is a graphical representation of a probabilistic problem, formally defined as a pair $\mathcal{B} = (G, P)$, where P is the joint probability distribution on the set of random variables and G is an ADG representing the dependency and independency relations among this set of random variables, where each graphically represented marginal and conditional independency also has to be valid in the joint probability distribution [90]. Clearly, the definition of Bayesian Networks implies as well that a dependence in the graph does not have to define a dependence in the joint probability distribution P.

Let $\{X_1, \ldots, X_d\}$ be a set of random variables. Then, based on the independency relations in the graph G, the joint probability distribution P can be factorised as follows:

$$P(X_1, \ldots, X_d) = \prod_{v=1}^{d} P(X_v \mid \pi(X_v)), \tag{9.2}$$

where $\pi(X_v)$ denotes the graphically represented set of parents of random variable X_v. This implies that a joint probability distribution can be defined in terms of local distributions resulting in significant computational savings.

For reasoning in Bayesian Networks there are several exact methods proposed that make use of local computations [26]. Here, local computations are based on the construction of join trees.

Hybrid Bayesian Networks consist of both discrete and continuous random variables [25]. In these networks, local computations are possible, however, the correctness of the inference method depends on whether parents of a variable are discrete, continuous, a mixture of discrete and continuous, and on the choice of the local probability distribution.

The first method, introduced by Lauritzen [70] using *exact* inference, is based on conditional Gaussian distributions. The restriction of this inference is that discrete random variables are not allowed to have continuous parents when hybrid Bayesian Networks are concerned. To overcome this problem, Koller proposed a method which defines the distribution of these discrete nodes by a mixture of exponentials. However, for the inference, Monte Carlo methods are used [65]. As another solution to this problem, we may discretise continuous variables, but discretisation introduces errors because we use approximation methods. However, in the experiment performed in this contribution we did not yet study the case of discrete nodes having continuous parents. For the Bayesian Networks related experiments the BNT tool developed by Murphy was used [86]. The same basic algorithm as for PBIL was used, except that the distribution type and the update procedure was changed. A detailed description of the update algorithm would exceed the scope of this work, and we refer to [86].

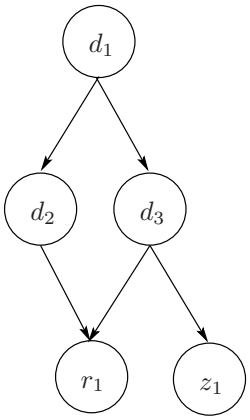
9.5 ADG-based NK-landscapes

ADG-based NK-landscapes (ADG-NKL), that we will introduce next, are attractive as models for optimization as their interaction structure corresponds to the dependence structure of Bayesian Networks. Let $x_1, \ldots, x_d$ denote a set of decision variables (the type of which can be continuous or discrete) and assume the interaction structure of the function is described by some ADGs. Each ADGs is basically defined by a function $\pi(\cdot)$ that assigns the set of parent nodes to each node, where the nodes represent parameters to be optimized. Then the ADG-based NK-landscape can be written as a function of component functions f_i:

$$f(x_1, \ldots, x_d) = \sum_{i=1}^{d} f_i(x_i, \pi(x_i)) \tag{9.3}$$

Note that this expression has the same structure as the expression $\log P(X_1, \ldots, X_d)$ (see Equation (9.2)). Note also that the $x_1, \ldots, x_d$ denote variables of the objective function in contrast to $X_1, \ldots, X_d$ which denote random variables.

The construction of the ADG-based NK-landscapes corresponds to that of classical mixed-integer NK-landscapes [78] with one exception. As for classical NK-landscapes for each decision variable (or *gene*) x_i we choose K epistatic genes



d1	f1
0	0.12
1	0.45

d3	d1	f3
0	0	0.42
0	1	0.56
1	0	0.67
1	1	0.92

r1	d2	d3	f4
0	0	0	0.34
0	0	1	0.26
0	1	0	0.45
0	1	1	0.21
1	0	0	0.81
1	0	1	0.46
1	1	0	0.51
1	1	1	0.48

d2	d1	f2
0	0	0.32
0	1	0.43
1	0	0.73
1	1	0.13

z1	d3	f5
0	0	0.98
0	1	0.34
1	0	0.84
1	1	0.09

$$f(x) = f_1(d_1) + f_2(d_2, d_1) + f_3(d_3, d_1) + f_4(r_1, d_1, d_3) + f_5(z_1, d_3)$$

Figure 9.2: Example for an ADG-based NK-landscape. The function values at the edge of the search space $[0, 1]^d$ are set randomly between 0 and 1. Values inbetween are interpolated [78].

that interact with x_i, in ADG-based NK-landscapes we chose exactly the parent nodes as epistatic genes. Note that the number of them can vary with the index of the decision variable in question. That is why the K in the expression 'NK-landscape' is not referring to the number of epistatic genes anymore - we kept it, however in the term, as it makes it easier to match the corresponding well known NK-landscapes with the ADG-based NK-landscapes. As with classical NK-landscapes, the definition of the component functions in ADG-based NK-landscapes is based on randomly generated function tables [78], as visualized in Figure 9.2. In the mixed-integer case multilinear functions are used to interpolate between the randomly chosen function values at the edges of a hypercube as described in [78].

9.6 Experimental Results

In order to check whether a-priori knowledge on the interaction structure integrated in the structure of the Bayesian Network helps to speed up search we have conducted experiments on various ADG types that are visualized in Figure 9.3. These ADGs were used to construct NK-landscapes that indicate that the represented independency and dependency relations respectively in an ADG are also included in the NK-landscape constructed from this ADG. The same ADG is used as a structure for the Bayesian Network as a-priori knowledge. For the probability tables, however, no a-priori knowledge is used. They are initialized based on the first population of selected individuals.

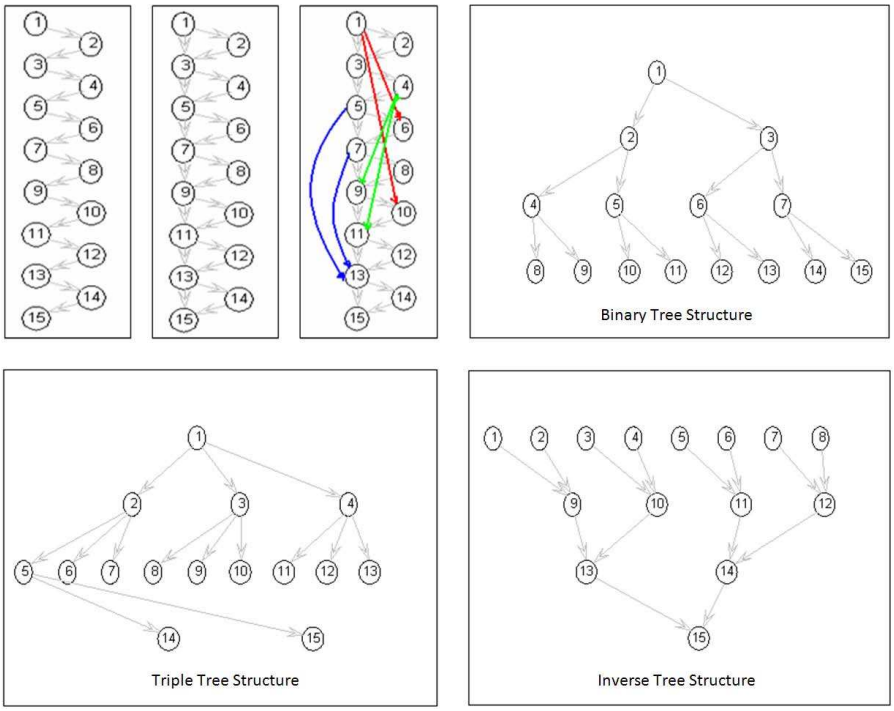


Figure 9.3: Various types of ADGs used to define ADG-based NK-landscapes and corresponding Bayesian Networks. From left to right, ADGs are termed 'chain', 'struct2', 'struct3', 'bitree', 'tritree', and 'invtree'. Node types are defined as follows: discrete nodes(1-5), continuous nodes(6-10), integer nodes(11-15).

We applied three types of algorithms on ADG-based NK-landscapes. 15 variables are considered, 5 for each type (l=m=n=5). As the population size turned out to be a crucial parameter, two different population sizes, 28 and 100, are tried. A number of 20 runs were statistically evaluated for each strategy.

Figures 9.4 to 9.6 show convergence dynamics for different sample landscapes defined by their ADG, each of which has a different structure. Averaged objective function values (difference to the global optimum) and standard deviations are plotted versus the number of evaluations performed.

On the landscape 'chain' (Figure 9.4), the MIBOA performs best, when the population size is set to 100. For a population size of 28 the MIBOA performs almost equally to the MIES. In both cases the MIPBIL algorithm was clearly outperformed.

On the landscape 'bitree' (Figure 9.5), a binary tree, the MIBOA performs best, when the population size is set to 100. For a population size of 28 the MIBOA is faster but in the long run MIPBIL results in (almost) the same good value. MIES seems to have a problem with this landscape, which may be due to

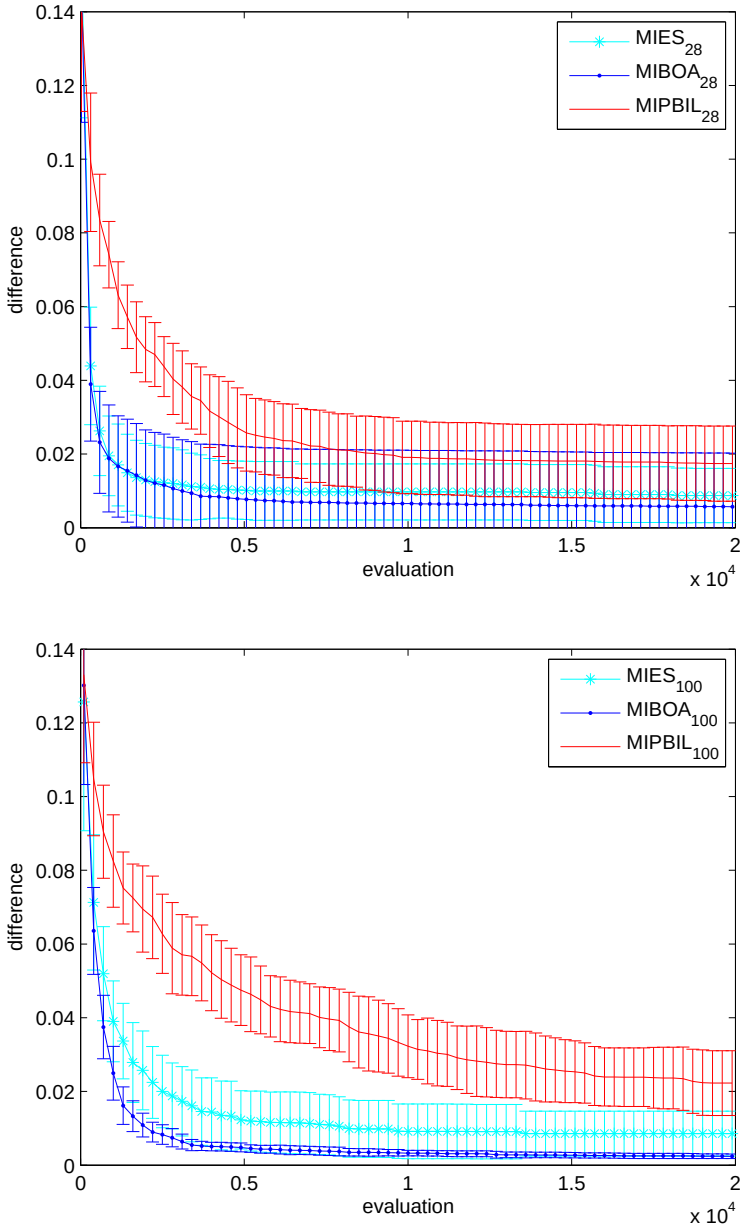


Figure 9.4: Convergence dynamics of MIES, MIPBIL, and MIBOA on a 'chain'-type ADG-NKL.

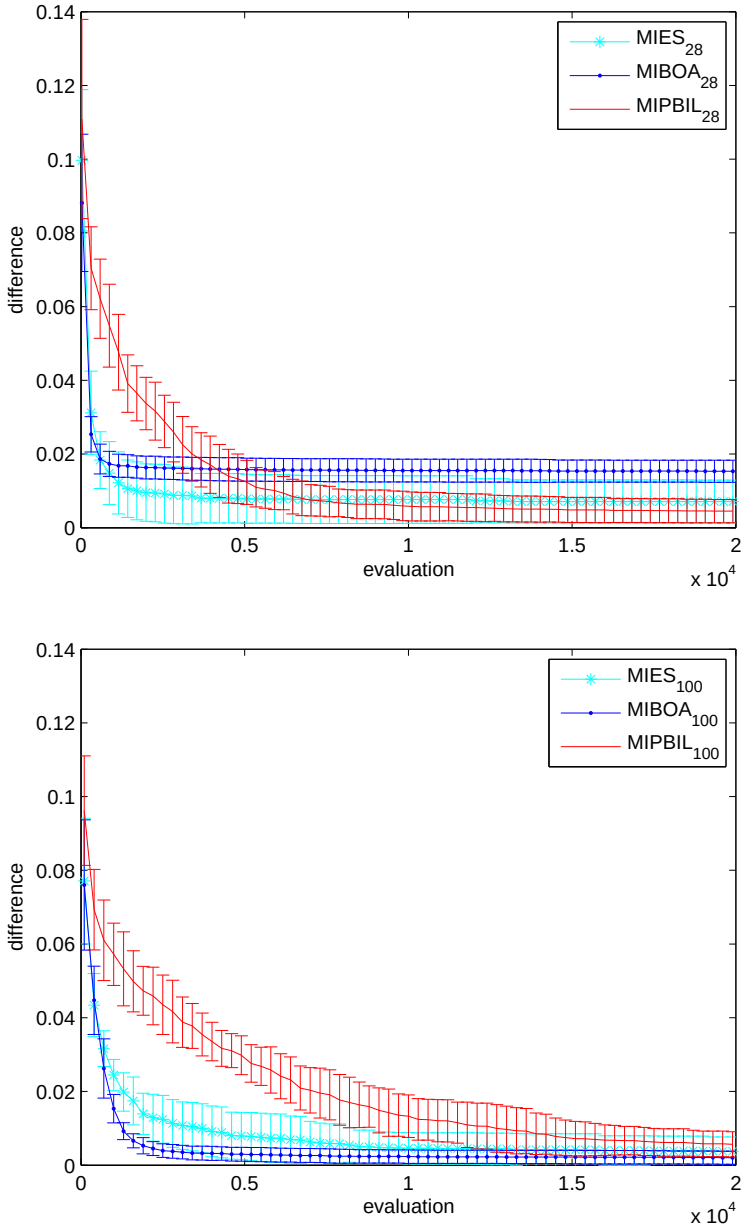


Figure 9.5: Convergence dynamics of MIES, MIPBIL, and MIBOA on a 'bitree'-type ADG-NKL.

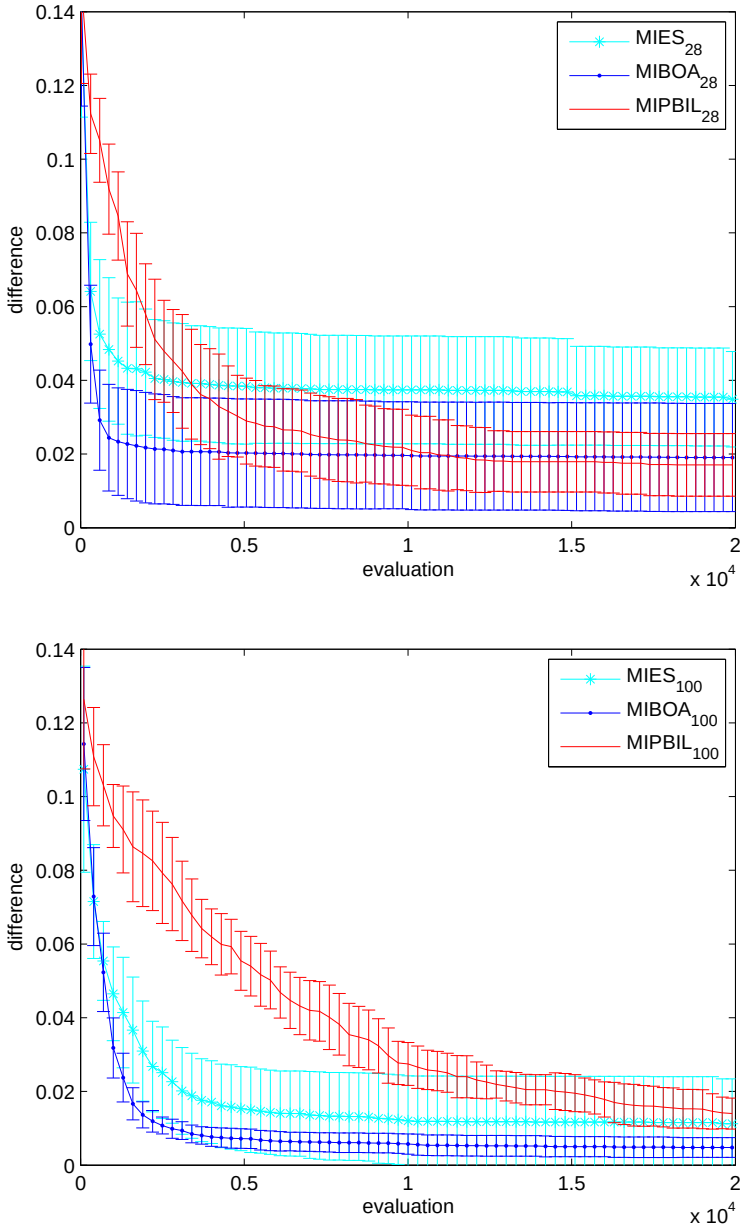


Figure 9.6: Convergence dynamics of MIES, MIPBIL, and MIBOA on the 'invtree'-type ADG-NKL.

step-size reduction which can be harmful in multimodal landscapes. The large standard deviation supports this conjecture.

On the landscape 'invtree' (Figure 9.6), again the MIBOA has a big advantage in the beginning. Here this acceleration is more visible than for the previous landscape types. Again the MIES algorithm seems to have problems to converge to the global optimum, while the MIPBIL is more reliable, but suffers from a low convergence speed.

Comparing a population size of 100 with a population size of 28, it was observed that the MIBOA algorithm performs better with the larger population size. The standard deviation of results in that case is remarkably lower, indicating a good reliability of the good results. In Table 9.2 we summarize more results, including the ADG types 'tritree', 'struct2', and 'struct3'. The ranking after 2000, 5000, 10000, and 20000 iterations is reported. This table provides further evidence for the hypothesis that the introduction of the dependence information in the MIBOA is beneficial. In addition, it can be observed that a small population size helps to speed up convergence of the algorithm in the short term, while a large population size improves its long term behaviour. For further details and results of this study we refer to [125].

9.7 Summary

In this chapter we studied how knowledge on acyclic dependency structures can be integrated into stochastic optimization for mixed-variable search spaces. The Mixed-integer Bayesian Optimizaton Algorithm (MIBOA), an estimation of distribution algorithm working with heterogeneous Bayesian Networks with a-priori set structure, was designed and studied. As a test environment mixed-integer NK-landscapes have been modified to ADG-based mixed-integer NK-landscapes. The dependence structure of their variables is defined as an ADG and, as a proof of concept, it had to be studied whether the MIBOA can exploit a-priori knowledge on this dependency structure or not. The test shows that the MIBOA algorithm can indeed take advantage of this a-priori information on dependencies. In all cases of ADGs discussed ('chain', 'struct2', 'struct3', 'bitree', 'tritree', and 'invtree') we observed a performance gain as compared to mixed-integer evolution strategies and estimation of distribution algorithms, both working with an independent joint distribution, namely MIES and MIPBIL. The population size of MIBOA turned out to be an important parameter to control the trade-off between fast convergence speed in the beginning and reliable convergence to the global optimum towards the end of the search. Future work will have to focus on studies on further synthetic and real-world problems, including cases where discrete parameters depend on continuous parameters, which turned out to be difficult to handle. In particular we are interested in applying the new algorithm in the context of optimization of image processing pipelines, the acyclic structure of which makes the MIBOA a particularly promising technique.

Algorithm	1000 Eval.		2000 Eval.		5000 Eval.		10000 Eval.		20000 Eval.	
	Ranks	$\sum$	Ranks	$\sum$	Ranks	$\sum$	Ranks	$\sum$	Ranks	$\sum$
$MIPBIL_{28}$	555565	31	555555	30	555555	30	353545	25	354655	28
$MIPBIL_{100}$	666656	35	666666	36	666666	36	666666	36	666546	33
$MIES_{28}$	222222	12	313333	16	433334	20	434334	21	223323	15
$MIES_{100}$	444444	24	444444	24	344441	20	545452	25	545462	26
$MIBOA_{28}$	111111	6	231111	9	221223	12	222223	13	432234	18
$MIBOA_{100}$	333333	18	122222	11	112112	8	111111	6	111111	6

Table 9.2: Ranking position of average objective function values for $MIES_{\mu}$, $MIPBIL_{\mu}$, and $MIBOA_{\mu}$ with population size 28 and 100 on ADG-based NK-landscapes after different numbers of evaluations.

Conclusion

Targeting specifically at challenging mixed-integer black-box optimization problems, in this dissertation, we proposed so-called Mixed-Integer Evolution Strategies (MIES) and did a thoroughly research on it from both theoretical and practical point of view. As a special variant of canonical Evolution Strategies (ESs), not only do MIES share some common characteristics with ESs – they both belong to the class of randomized search heuristics and use principles of organic evolution, such as selection, recombination, and mutation – MIES also expand ESs from traditional continuous optimization domain to more complicated mixed-integer parameter optimization field, in which simultaneous optimization of continuous, integer, and nominal discrete parameters is often required. In addition to systematic experiments on our carefully designed synthetic functions (e.g., barrier functions and mixed-integer NK-landscapes), MIES have been successfully applied to various real world problems, for instance, optimization of control parameters of a semi-automatic image analysis system for medical images.

In this thesis, our presented work is divided into three parts: (1) Mixed-Integer evolution strategies; (2) Application to medical image analysis; (3) Advanced topics. In the rest of this chapter, we summarize our conclusions chapter by chapter and furthermore discuss some issues for future work.

Part I: Mixed-Integer Evolution Strategies

Chapter 2

We presented different types of mixed-integer nonlinear programming problems and talked briefly about some classical methods which come from traditional mathematical programming research field. In comparison with these what we called "*white-box*" optimization problems, "*black-box*" optimization problems normally with unclearly objective function structure and high dimensionality are more difficult to deal with. Some heuristic methods, such as Genetic Algorithms (GAs) and Simulated annealing (SA), come into play under such circumstances.

Chapter 3

In this chapter, the design philosophy of the MIES, which are derived from standard ESs, were explained explicitly. Furthermore, we made some theoretical studies on MIES, such as self-adaptation of stepsize and the global convergency property.

Chapter 4

Two artificial landscapes – Barrier functions and Mixed-Integer NK Landscapes (MINLP) – were introduced in this chapter. Experimental results showed that these functions can be used as ideal test cases for helping us to learn more about MIES. Besides, they give readers a good chance to make comparison between MIES and standard ESs.

Future Work for Part I

In part I, mixed-integer optimization, especially black-box mixed-integer parameter optimization, was discussed at first. Next, mixed-integer evolution strategies were introduced and studied through several carefully designed artificial test functions. By analyzing (e.g., statistical study) some important experimental results, we gained deep insights about MIES algorithm, such as convergence behavior. As we always emphasized, by design, MIES are capable to tackle the "*black-box*" mixed-integer parameter optimization problems. However, as an alternative, MIES can also be used to solve some classical mixed integer nonlinear programming problems. In the future, we would like to do some further investigation on how to apply MIES to these classical optimization problems from mathematical programming field, for instance, study on how to construct proper penalty functions based on complex constraints.

Part II: Application to Medical Image Analysis

Chapter 5

In this chapter, we presented the complete framework of how to apply MIES to an optimization problem in medical image analysis. The experimental results showed that the MIES always produced better or equal results than the default parameter settings chosen by an expert. This observation underpinned our claim that MIES is a valuable technique for improve the parameter settings of the lumen detector.

Chapter 6

We investigated the use of fitness based partitioning in order to find sets of optimal parameters for the segmentation of the lumen in Computer Tomographic Angiography (CTA) images. The results showed that fitness based partitioning

does indeed produce sets of parameter settings which lead to better lumen segmentations when compared to one "super" solution for all images.

Future Work for Part II

In this part, our proposed MIES were applied to a specific application which comes from medical research field: the optimization of control parameters of a semi-automatic image analysis system for medical images, such as Intravascular Ultrasound (IVUS) and Computer Tomographic Angiography (CTA) images. Specifically, dynamic fitness based partitioning was proposed to help system to find specific optimal parameter settings for different groups of images instead of optimal solution for all images.

For the future work, it would be worth trying MIES on larger image sets as well as on other feature detectors except for lumen, such as calcified plaque, vessel border, shadow and sidebranch. About dynamic fitness based partitioning, we intend to extend this algorithm with merge and split heuristics to automatically find an optimal number of partitions.

Part III: Advanced Topics

Chapter 7

This chapter talked about the metamodel-assisted MIES, which is based on radial basis function networks (RBFN). The reason for this is that the evaluation of one parameter settings for feature detection of a multi-agent medical image analysis system is computationally expensive. By introducing a metamodel, such as RBFN, acceleration on convergence speed of MIES can be achieved.

Chapter 8

In this chapter, we presented a dynamic niching technique for MIES. In comparison with an existing ES niching approach, our approach is based on the heterogeneous distance measure that addresses search space similarity in a way consistent with the design philosophy of the MIES. The experimental results showed that MIES with dynamic niching perform well in obtaining the global optimum in highly multimodal search landscapes, such as mixed-integer NK landscapes.

Chapter 9

We proposed a Mixed-Integer Bayesian Optimization Algorithm (MIBOA) in this chapter to overcome a known shortcoming of existing mixed-integer evolutionary algorithms – their variation procedures mutate each decision variable independently, and as a result of it, a-priori dependencies knowledge between variables

cannot be taken into account. The test results showed that the MIBOA algorithm can indeed take advantage of such kind of a-priori information on dependencies.

Future Work for Part III

In the final part, we studied several advanced techniques – radial basis function networks (RBFN), dynamic niching and Bayesian networks – which can be used together with MIES to further improve the performance of algorithm. In the future, we would like to continue with our studies on these topics from the following perspectives: (1) RBFN-MIES needs to be tested on more challenging problems, such as problems from medical image analysis. A theoretical study on how continuity assumptions can be generalized for mixed-integer domains would also be an interesting topic for our future work; (2) MIES with dynamic niching should be tested on real-world applications, and we can gain a deepened understanding of niche information process; (3) For MIBOA, the future work could be focused on more difficult cases where discrete parameters depend on continuous parameters.

Appendix A

Selected Synthetic Functions

Besides the two artificial test problems Barrier function (section 4.2) and MINKL (section 4.3), we will now present four other mixed-integer test problems: Generalized sphere function, weighted sphere function, modified step function, and general quadratic function.

A.1 Generalized Sphere Function

The generalized sphere model (Function f_1) is an extension of a standard problem [38], This problem is relatively simple, as it is decomposable and unimodal. We can use it to gain some insights of how an algorithm behaves on rather simple problems and thus to estimate the best case behavior of the algorithm.

$$f_1(\mathbf{r}, \mathbf{z}, \mathbf{d}) = \sum_{i=1}^{n_r} r_i^2 + \sum_{i=1}^{n_z} z_i^2 + \sum_{i=1}^{n_d} d_i^2 \tag{A.1}$$

A.2 Weighted Sphere Function

The weighted sphere model (Function f_2) represents a function with an elliptical geometry. Experiments on this function can detect if a speed up can be achieved by the learning of individual strategy parameters for each parameter. Furthermore it is an example for a function with a simple quadratic and convex geometry.

$$f_2(\mathbf{r}, \mathbf{z}, \mathbf{d}) = \sum_{i=1}^{n_r} i r_i^2 + \sum_{i=1}^{n_z} i z_i^2 + \sum_{i=1}^{n_d} i d_i^2 \tag{A.2}$$

A.3 Modified Step Function

The step function (Function f_3) has been chosen to show that MIES is capable to tackle large plateaus in the fitness landscape. The plateau is used for linked areas of neighboured solutions in the search space, that lead to the same fitness value. Such plateaus happen in practical applications for example when searching for feasible points, using penalty functions that are proportional to the number of violated constraints or simulation errors.

$$f_3(\mathbf{r}, \mathbf{z}, \mathbf{d}) = \sum_{i=1}^{n_r} \lfloor r_i \rfloor^2 + \sum_{i=1}^{n_z} (z_i \text{ div } 10)^2 + \sum_{i=1}^{n_d} (d_i \bmod 2)^2 \tag{A.3}$$

A.4 General Quadratic Function

The general quadratic function (Function f_4) represents a strong interaction between all parameters. The contour lines of this function have approximately the shape of ellipsoids.

$$f_4(\mathbf{r}, \mathbf{z}, \mathbf{d}) = \sum_{i=1}^{n} (\sum_{j=1}^{i} r_j + z_j + d_j)^2 \tag{A.4}$$

Bibliography

[1] L. Altenberg. Nk fitness landscapes. In T. Bäck, D.B. Fogel, and Z. Michalewica, editors, *Handbook of Evolutionary Computation*, volume 2. Oxford University Press, 1997.

[2] T. Bäck. Evolution strategies: An alternative evolutionary algorithm. In et al. J-M. Alliot, editor, *Artificial Evolution, European Conference, AE 95, Brest, France, September 4-6, 1995, Selected Papers*, volume 1063 of *Lecture Notes in Computer Science*, pages 3–20. Springer, 1995.

[3] T. Bäck. *Evolutionary Algorithms in Theory and Practice.* Oxford University Press, New York, NY, USA, 1996.

[4] T. Bäck. Evolutionary computation: A guided tour. *Bulletin of the EATCS*, 77:132–166, 2002.

[5] T. Bäck. Evolutionary computation: A guided tour. In G. Paun et al., editor, *Current Trends in Theoretical Computer Science*, volume 1, pages 569–612. World Scientific, 2004.

[6] T. Bäck, U. Hammel, and H.-P. Schwefel. Evolutionary computation: Comments on the history and current state. *IEEE Trans. Evolutionary Computation*, 1(1):3–17, 1997.

[7] T. Bäck and M. Schütz. Evolution strategies for mixed-integer optimization of optical multilayer systems. In *Evolutionay Programming*, pages 33–51, 1995.

[8] L. Ballerini and L. Franzén. Genetic optimization of morphologicial filters with applications in breast cancer detection. In *Applications of Evolutionary Computing*, LNCS 3005, pages 250–259, Coimbra, Portugal, 2004. Springer.

[9] S. Baluja and R. Caruana. Removing the genetics from the standard genetic algorithm. In A. Prieditis and S. Russel, editors, *The Int. Conf. on Machine Learning 1995*, pages 38–46, San Mateo, CA, 1995. Morgan Kaufmann Publishers.

[10] R. Bellman. *Dynamic Programming.* Dover Publications Inc., 1957.

[11] H.-G. Beyer. *The Theory of Evolution Strategies.* Springer-Verlag New York, Inc., New York, NY, USA, 2001.

[12] H.-G. Beyer and H.-P. Schwefel. Evolution strategies: A comprehensive introduction. *Natural Computing*, 1(1):3–52, 2002.

[13] H.-G. Beyer, H.-P. Schwefel, and I. Wegener. How to analyse evolutionary algorithms. *Theor. Comput. Sci.*, 287(1):101–130, 2002.

[14] B. Borchers. Branch and bound methods. In C.A. Floudas and P.M. Pardalos, editors, *Encyclopedia of Optimization*, volume 3, pages 331–335. Kluwer Press, 2001.

[15] J. Born. *Evolutionsstrategien zur numerischen Lösung von Adaptationsaufgaben.* Dissertation a, Humboldt-Universität, GDR, 1978.

[16] E.G.P. Bovenkamp, J. Dijkstra, J.G. Bosch, and J.H.C. Reiber. Multi-agent segmentation of ivus images. *Pattern Recognition*, 37(4):647–663, April 2004.

[17] E.G.P. Bovenkamp, J. Dijkstra, J.G. Bosch, and J.H.C. Reiber. User-agent cooperation in multi-agent ivus image segmentation. *IEEE Transactions on Medical Imaging*, accepted, 2008.

[18] E.G.P. Bovenkmap, J. Eggermont, R. Li, M.T.M. Emmerich T. Bäck, J. Dijkstra, and H.C. Reiber. Optimizing ivus lumen segmentations using evolutionary algorithms. In et al. I. Kakadiaris, editor, *Proceedings of the 1st International Workshop on Computer Vision for Intravascular and Intracardiac Imaging, MICCAI 2006*, Copenhaguen, Denmark, October 2006. IEEE.

[19] G.E.P. Box, W.G. Hunter, and J.S. Hunter. *Statistics for Experiments.* Wiley Press, 1978.

[20] S. Boyd, A. Ghost, and A. Magnani. Branch and bound methods. Notes for EE392o, November 2003.

[21] D.S. Broomhead and D. Lowe. Multivariable functional interpolation and adaptive networks. *Complex System*, 2(3):321–355, 1988.

[22] D. Büche, N.N. Schraudolph, and P. Koumoutsakos. Accelerating evolutionary algorithms with gaussian process fitness function models. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 35(2):183–194, May 2005.

[23] M.R. Bussieck and A. Pruessner. Mixed-integer nonlinear programming. *SIAG/OPT Views-and-News*, 14(1):19–22, February 2003.

[24] S. Cagnoni, A. B. Dobrzeniecki, R. Poli, and J. C. Yanch. Genetic algorithm-based interactive segmentation of 3d medical images. *Image and Vision Computing*, 17(12):881– 895, October 1999.

[25] B. C. Cobb, R. Rumi, and A. Salmeron. Bayesian network models with discrete and continuous variables. In P. Lucas et al., editor, *Advances in Probabilistic Graphical Models*, volume 214 of *Studies in Fuzziness and Soft Computing*, pages 81–102. Springer, 2007.

[26] R. G. Cowell, A. Philip Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. *Probabilistic Networks and Expert Systems.* Springer-Verlag New York, 1999.

[27] G.B. Dantzig. *Linear Programming and Extensions.* Princeton University Press, 1963.

[28] C.R. Darwin. *On the Origin of Species.* John Murray, 1859.

[29] K. Deb and D.E. Goldberg. An investigation of niche and species formation in genetic function optimization. In J.D. Schaffer, editor, *Proceedings of the 3rd International Conference on Genetic Algorithms (ICGA), George Mason University, Fairfax, Virginia, USA, June 1989*, pages 42–50. Morgan Kaufmann, 1989.

[30] S. Droste and D. Wiesmann. Metric based evolutionary algorithms. In *EuroGP*, LNCS 1802, pages 29–43. Springer, 2000.

[31] M.A. Duran and I.E. Grossmann. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mahtematical Programming*, 36(3):307–339, October 1986.

[32] J. Eggermont, E.G.P. Bovenkamp, R. Li, M.T.M. Emmerich, T. Bäck, J. Dijkstra, and J.H.C. Reiber. Extending nk landscapes to the mixed-integer domain. In *Proceedings of the 18th Belgium-Netherlands conference on artificial intelligence (BNAIC), Namur, Belgium*, 2006. Extended abstract.

[33] J. Eggermont, R. Li, E.G.P. Bovenkamp, H.A. Marquering, M.T.M. Emmerich, A.v.d. Lugt, T. Bäck, J. Dijkstra, and J.H.C. Reiber. Optimizing computed tomographic angiography image segmentation using fitness based partitioning. In *EvoIASP08*, volume 4974 of *LNCS*, pages 275–284. Springer, 2008.

[34] A.E. Eiben and J.E. Smith. *Introduction to Evolutionary Computing.* SpringerVerlag, 2003.

[35] M. El-Beltagy, P.B. Nair, and A.J. Keane. Metamodeling techniques for evolutionary optimization of computationally expensive problems: Promises and limitations. In Wolfgang Banzhaf, Jason M. Daida, A. E. Eiben, Max H. Garzon, Vasant Honavar, Mark J. Jakiela, and Robert E. Smith, editors, *Genetic and Evolutionary Computation Conference (GECCO 1999), Proceedings*, pages 196–203, Orlando, Florida, USA, 13 - 17, July 1999. Morgan Kaufmann.

[36] M.T.M. Emmerich. *Single- and Multi-objective Evolutionary Design Optimization Assisted by Gaussian Random Field Metamodels.* Phd thesis, University of Dortmund, Eldorado, University of Dortmund, 2005.

[37] M.T.M. Emmerich, A. Giotis, M. Özdemir, T. Bäck, and K. Giannakoglou. Metamodel-assisted evolution strategies. In J.J.M. Guervós, P. Adamidis, H.-G. Beyer, José Luis Fernández-Villacañas Martín, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature - PPSN VII, 7th International Conference*, volume 2439 of *Lecture Notes in Computer Science*, pages 361–370, Granada, Spain, 7-11, September 2002. Springer.

[38] M.T.M Emmerich, M. Grötzner, B. Groß, and M. Schütz. Mixed-integer evolution strategy for chemical plant optimization with simulators. In et al. I.C. Parmee, editor, *Evolutionary Design and Manufacture–Selected papers from ACDM'00, Plymouth, UK, April 26-28, 2000*, pages 55–67. Springer, London, 2000.

[39] M.T.M. Emmerich, R. Li, J. Eggermont, E.G.P. Bovenkamp, T. Bäck, J. Dijkstra, and J.H.C. Reiber. Optimizing parameters of coronary vessel image analysis using evolution strategies. In *Proceedings of the 18th Belgium-Netherlands conference on artificial intelligence (BNAIC), Namur, Belgium*, 2006. Extended abstract.

[40] M.T.M. Emmerich, R. Li, A.Y. Zhang, I. Flesch, and P. Lucas. Mixed-integer bayesian optimization utilizing a-priori knowledge on parameter dependences. In *Proceedings of the 20th Belgium-Netherlands conference on artificial intelligence (BNAIC), Twente, Netherlands*, 2008.

[41] R. Fletcher and S. Leyffer. Solving mixed integer nonlinear programs by outer approximation. *Mahtematical Programming*, 66(1-3):327–349, August 1994.

[42] C.A. Floudas. *Nonlinear and Mixed Integer Optimization: Fundamentals and Applications.* Oxford University Press, 1995.

[43] A.M. Geoffrion. Generalized benders decomposition. *Journal of Optimization Theory and Applications*, 10(4):237–260, October 1972.

[44] A.P. Giotis, M.T.M. Emmerich, B. Naujoks, K. Giannakoglou, and T. Bäck. Low cost stochastic optimisation for engineering applications. In *Proceedings Int'l Conf. Industrial Applications of Evolutionary Algorithms, EUROGEN2001, 19-21, September Athens, Greece, Barcelona, 2001. CIMNE (CD-ROM).* National Technical University of Athens and the University of Patras, 2001.

[45] A.P. Giotis, K.C. Giannakoglou, and J. Périaux. A reduced-cost multi-objective optimization method based on the pareto front technique, neural networks and pvm. In *Proceedings of European Congress on Computational*

Methods in Applied Sciences and Engineering (ECCOMAS'00), (CD-ROM), *September 11-19, Barcelona, 2000, Spain.* Center for Numerical Methods in Engineering (CIMNE), 2000.

[46] David E. Goldberg and Kalyanmoy Deb. A comparative analysis of selection schemes used in genetic algorithms. In *Foundations of Genetic Algorithms*, pages 69–93. Morgan Kaufmann, 1991.

[47] B. Groß. *Gesamtoptimierung verfahrenstechnischer Systeme mit Evolutionären Algorithmen.* Dissertation, Rheinisch – Westfälische Technische Hochschule Aachen, Lehrstuhl für Technische Thermodynamik, 1999.

[48] I.E. Grossmann and N.V. Sahinidis. Special issue on mixed-integer programming and it's application to engineering. *Optimization and Engineering*, 3(3), 2002.

[49] O.K. Gupta and A. Ravindran. Branch and bound experiments in convex nonlinear integer programming. *Management Science*, 31(12):1533–1546, 1985.

[50] S. Haykin. *Neural Networks: A Comprehensive Foundation.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 1998.

[51] D. Heckerman. A tutorial on learning with bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research, Advanced Technology Division, Redmond, WA 98052, 1995.

[52] F. Hoffmeister and T. Bäck. Genetic algorithms and evolution strategies: Similarities and differences. In H.-P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature, 1st Workshop, PPSN I, Dortmund, FRG, October 1-3, 1990, Proceedings*, volume 496 of *Lecture Notes in Computer Science*, pages 455–469. Springer, 1990.

[53] F. Hoffmeister and J. Sprave. Problem-independent handling of constraints by use of metric penalty functions. In L.J. Fogel et al., editor, *Evolutionary Programming 1996, San Diego, CA, USA*, pages 289–294. IEEE Press, 1996.

[54] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, September 1999.

[55] Y. Jin. On evolutionary optimization with approximate fitness functions. In L.D. Whitley, D.E. Goldberg, E. Cantú-Paz, L. Spector, I.C. Parmee, and H.-G. Beyer, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '00), Las Vegas, Nevada, USA, July 8-12, 2000*, pages 786–793. Morgan Kaufmann, 2000.

[56] Y. Jin. A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing Journal*, 9(1):3–12, 2005.

[57] T. Jones. *Evolutionary Algorithms, Fitness Landscapes and Search*. PhD thesis, The University of New Mexico, May 1995.

[58] K. De Jong, D. David, and D. B. Fogel H.-P. Schwefel. A history of evolutionary computation. In T. Bäck, D.B. Fogel, and Z. Michalewicz, editors, *Handbook of Evolutionary Computation*, pages 1–12. Oxford University, New York, 1997.

[59] M.I. Jordan, editor. *Learning in Graphical Models*. MIT Press, 1999.

[60] H. Kargupta and D.E. Goldberg. Blackbox optimization: Implications of search. In *In communication. Submitted in International Journal of Foundation of Computer Science*, pages 96–63, 1996.

[61] S.A. Kauffman. *The Origins of Order: Self-Organization and Selection in Evolution*. Oxford University Press, 1993.

[62] M. Keijzer, J. J. Merelo, G. Romero, and M. Schoenauer. Evolving objects: A general purpose evolutionary computation library. In P. Collet, C. Fonlupt, J.-K. Hao, E. Lutton, and M. Schoenauer, editors, *5th International Conference, Evolution Artificielle, EA 2001, Le Creusot, France, October 29-31, 2001*, volume 2310 of *Lecture Notes in Computer Science*, pages 829–888. Springer Berlin / Heidelberg, 2001.

[63] R.C. Kelahan and J.L. Gaddy. Application of the adaptive random search to discrete and mixed integer optimization. *International Journal for Numerical Methods in Engineering*, 12(2):289–298, 1978.

[64] M. Kimura. *The neutral theory of molecular evolution*. Cambridge University Press, 1985.

[65] D. Koller, U. Lerner, and D. Anguelov. A general algorithm for approximate inference and its application to hybrid bayes nets. In *Proceedings of Conference on Uncertainty in Artificial Intelligence UAI*, pages 302–313, 2007.

[66] G. Koning, J. Dijkstra, C. von Birgelen, J.C. Tuinenburg, J. Brunette, J.-C. Tardif, P.W. Oemrawsingh, C. Sieling, and S. Melsa. Advanced contour detection for three dimensional intracoronary ultrasound: validation - in vitro and in vivo. *The International Journal of Cardiovascular Imaging*, 18:235–248, 2002.

[67] S. Kotz and N.L. Johnson. *Encyclopedia of Statistical Science*, volume 2. Wiley, 1981.

[68] G. Rozenberg L. Kari. The many facets of natural computing. *Communications of ACM*, 51(10):72–83, October 2008.

[69] P. Larranaga and J.A. Lozano, editors. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*, volume 2 of *Genetic Algorithms and Evolutionary Computation*. Kluwer Academic, 2001.

[70] S.L. Lauritzen. *Graphical Models*. Clarendon Press, Oxford, 1996.

[71] E.L. Lawer and D.E. Wood. Branch-and-bound methods: A survey. *Operations Research*, 14(4):699–719, 1966.

[72] R. Li, J. Eggermont, M.T.M. Emmerich, E.G.P. Bovenkamp, T. Bäck, J. Dijkstra, and J.H.C. Reiber. Towards dynamic fitness based partitioning for intravascular ultrasound image analysis. In *Applications of Evolutinary Computing, EvoWorkshops 2007: EvoCoMnet, EvoFIN, EvoIASP,EvoINTERACTION, EvoMUSART, EvoSTOC and EvoTransLog, Valencia, Spain, April11-13, 2007, Proceedings*, volume 4448 of *Lecture Notes in Computer Science*, pages 391–398. Springer, 2007.

[73] R. Li, J. Eggermont, M.T.M. Emmerich, E.G.P. Bovenkamp, T. Bäck, J. Dijkstra, and J.H.C. Reiber. Towards dynamic fitness based partitioning for intravascular ultrasound image analysis. In *Proceedings of the 19th Belgium-Netherlands conference on artificial intelligence (BNAIC), Utrecht, Netherlands*, 2007. Extended abstract.

[74] R. Li, J. Eggermont, O.M. Shir, M.T.M. Emmerich, T. Bäck, J. Dijkstra, and J.H.C. Reiber. Mixed-integer evolution strategies with dynamic niching. In *Parallel Problem Solving from Nature - PPSN X, 10th International Conference Dortmund, Germany, September 13-17, 2008, Proceedings*, volume 5199 of *Lecture Notes in Computer Science*, pages 246–255. Springer, 2008.

[75] R. Li, M.E.M. Emmerich, E.G.P. Bovenkamp, J. Eggermont, T. Bäck, J. Dijkstra, and J.H.C. Reiber. Mixed-integer evolution strategies and their application to intravascular ultrasound image analysis. In Franz Rothlauf et al., editor, *Applications of Evolutionary Computing, EvoWorkshops 2006: EvoBIO, EvoCOMNET, EvoHOT, EvoIASP, EvoINTERACTION, EvoMUSART, and EvoSTOC, Budapest, Hungary, April 10-12, 2006, Proceedings*, volume 3907 of *Lecture Notes in Computer Science*, pages 415–426. Springer, 2006.

[76] R. Li, M.T.M. Emmerich, T. Bäck, E.G.P. Bovenkamp, J. Eggermont, J. Dijkstra, and J.H.C. Reiber. Optimizing parameters of intravascular ultrasound image analysis using mixed-integer evolution strategies. In I.C. Parmee et al, editor, *Proceedings of adaptive computing in design and manufacture (ACDM), April 25 - 27, 2006, Bristol, UK*, 2006.

[77] R. Li, M.T.M. Emmerich, J. Eggermont, and E.G.P. Bovenkamp. Mixed-integer optimization of coronary vessel image analysis using evolution strate-

gies. In M. Cattolico, editor, *Genetic and Evolutionary Computation Conference, GECCO 2006, Proceedings, Seattle, Washington, USA, July 8-12, 2006*, pages 1645–1652. ACM Press, 2006.

[78] R. Li, M.T.M. Emmerich, J. Eggermont, E.G.P. Bovenkamp, T. Bäck, J. Dijkstra, and J.H.C. Reiber. Mixed-integer nk landscapes. In T.P. Runarsson et al., editor, *Parallel Problem Solving from Nature - PPSN IX, 9th International Conference, Reykjavik, Iceland, September 9-13, 2006, Procedings*, volume 4193 of *Lecture Notes in Computer Science*, pages 42–51. Springer, 2006.

[79] R. Li, M.T.M. Emmerich, J. Eggermont, E.G.P. Bovenkamp, T. Bäck, J. Dijkstra, and J.H.C. Reiber. Optimizing a medical image analysis system using mixed-integer evolution strategies. In S. Cagnoni et al., editor, *Evolutionary Image Analysis and Signal Processing*, Studies in Computational Intelligence, pages 91–112. Springer, 2009.

[80] R. Li, M.T.M. Emmerich, E.G.P.Bovenkamp, J. Eggermont, T. Bäck, J. Dijkstra, and J.H.C. Reiber. Metamodel-assisted mixed integer evolution strategies and their application to intravascular ultrasound image analysis. In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2005, 1-6 June, 2008, Hong Kong, China*. IEEE, 2008.

[81] J.T. Linderoth and M.W.P Savelsbergh. A computational study of search strategies for mixed integer programming. Technical report, Georgia Institute of Technology, 1997.

[82] R. Lohmann. Structure evolution and incomplete induction. In et al. R. Männer, editor, *Parallel Problem Solving from Nature 2, PPSN-II, Brussels, Belgium, September 28-30, 1992*, pages 177–188. Elsevier, 1992.

[83] S.W. Mahfoud. *Niching Methods for Genetic Algorithms*. PhD thesis, University of Illinois at Urbana Champaign, 1995.

[84] H.A. Marquering, J. Dijkstra, P.J.H. de Koning, B.C. Stoel, and J.H.C. Reiber. Towards quantitative analysis of coronary cta. *Int J Cardiovasc Imaging*, 21(1):73–84, 2005.

[85] B.L. Miller and M.J. Shaw. Genetic algorithms with dynamic niche sharing for multimodal function optimization. In *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation (ICEC'96)*, pages 786–791, 1996.

[86] K.P. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, UC Berkeley, 2002.

[87] A. Newell. *Unified Theories of Cognition*. Harvard University Press, Cambridge, MA, USA, 1990.

[88] Ahmet Irfan Oyman, H.-G. Beyer, and H.-P. Schwefel. Analysis of the (1, lambda)-es on the parabolic ridge. *Evolutionary Computation*, 8(3):249–265, 2000.

[89] M. Papadogiorgaki, V. Mezaris, Y.S. Chatzizisis, G.D. Giannoglou, and I. Kompatsiaris. Automated ivus contour detection using intesity features and radial basis function approximation. In *CBMS '07: Proceedings of the Twentieth IEEE International Symposium on Computer-Based Medical Systems*, pages 183–188, Washington, DC, USA, 2007. IEEE Computer Society.

[90] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kauffman, San Francisco, CA, 1988.

[91] M. Pelikan, D.E. Goldberg, and E. Cantú-Paz. BOA: The Bayesian optimization algorithm. In W. Banzhaf et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99*, volume I, pages 525–532, Orlando, FL, 13-17 1999. Morgan Kaufmann Publishers, San Fransisco, CA.

[92] M. Preuss, L. Schönemann, and M.T.M. Emmerich. Counteracting genetic drift and disruptive recombination in $(\mu + /, \lambda)$-ea on multimodal fitness landscapes. In H.-G. Beyer and U.-M. O'Reilly, editors, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2005*, pages 865–872. ACM, 2005.

[93] S.S. Rao. *Engineering Optimization: Theory and Practice*. Wiley, 1996. 3rd Edition.

[94] A. Ratle. Accelerating the convergence of evolutionary algorithms by fitness landscape approximation. In A.E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature - PPSN V, 5th International Conference, Amsterdam, The Netherlands, September 27-30, 1998, Proceedings*, volume 1498 of *Lecture Notes in Computer Science*, pages 87–96. Springer, 1998.

[95] I. Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Phd dissertation, Technical University of Berlin, 1971.

[96] I. Rechenberg. *Evolutionstrategie: Optimierung Technisher Systeme nach Prinzipien des Biologischen Evolution*. Fromman-Hozlboog Verlag, Stuttgart, Germany, 1973.

[97] M.E. Roberts and E. Claridge. Cooperative coevolution of image feature construction and object detection. In X. Yao, E.K. Burke, J.A. Lozano, J. Smith, J.J. M. Guervós, J.A. Bullinaria, J.E. Rowe, P. Tiño, A. Kabán, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature - PPSN VIII, 8th International Conference, Birmingham, UK, September 18-22,*

2004, Proceedings, volume 3242 of *Lecture Notes in Computer Science*, pages 902–911. Springer, 2004.

[98] W. Rosamond, K. Flegal, G. Friday, K. Furie, A. Go, K. Greenlund, N. Haase, M. Ho, V. Howard, B. Kissela, S. Kittner, D. Lloyd-Jones, M. McDermott, J. Meigs, C. Moy, G. Nichol, C. J. O'Donnell, V. Roger, J. Rumsfeld, P. Sorlie, J. Steinberger, T. Thom, S. Wasserthiel-Smoller, and Y. and Hong. Heart disease and stroke statistics–2007 update: a report from the american heart association statistics committee and stroke statistics subcommittee. *Circulation*, 115(5), February 2007.

[99] P.S. Rosenbloom, A. Newell, and J.E. Laird, editors. *Soar Papers: Research on Integrated Intelligence*. MIT Press, Cambridge, MA, USA, 1993.

[100] G. Rudolph. On correlated mutations in evolution strategies. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature 2, PPSN-II, Brussels, Belgium, September 28-30, 1992*, pages 107–116. Elsevier, 1992.

[101] G. Rudolph. An evolutionary algorithm for integer programming. In Y. Davidor et al., editor, *Parallel Problem Solving from Nature - PPSN III, International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature, Jerusalem, Israel, October 9-14, 1994, Proceedings*, volume 866 of *Lecture Notes in Computer Science*, pages 139–148. Springer, 1994.

[102] G. Rudolph. *Convergence Properties of Evolutionary Algorithms*. Schriftenreihe Forschungsergebnisse zur Informatik. Kovac Press, 1997.

[103] R. Sanz-Requenaa, D. Moratala, D. Ramón García-Sáncheza, V. Bodíb, J.J. Rietaa, and J.M. Sanchis. Automatic segmentation and 3d reconstruction of intravascular ultrasound images for a fast preliminar evaluation of vessel pathologies. *Computerized Medical Imaging and Graphics*, 31(2):71–80, March 2007.

[104] L. Schönemann, M.T.M. Emmerich, and M. Preuss. On the extinction of evolutionary algorithms sub-populations on multimodal landscapes. *Informatica - Special Issue on Bioinspired Optimization*, 28(4):345–351, 2004.

[105] M. Schütz and J. Sprave. Application of parallel mixed-integer evolution strategies with mutation rate pooling. In *Evolution Programming*, pages 345–354, 1996.

[106] H.-P. Schwefel. *Evolutionsstrategie und numerische optimierung*. Dissertation, TU-Berlin, Germany, 1975.

[107] H.-P. Schwefel. *Evolution and Optimum Seeking: The Sixth Generation Computer Technonogy Series*. John Wiley & Sons, Inc., New York, NY, USA, 1993.

[108] M. Sebag and A. Ducoulombier. Extending population-based incremental learning to continuous search spaces. In *PPSN V: Proceedings of the 5th International Conference on Parallel Problem Solving from Nature*, pages 418–427, London, UK, 1998. Springer-Verlag.

[109] O.M. Shir and T. Bäck. Dynamic niching in evolution strategies with covariance matrix adaptation. In *Congress on Evolutionary Computation*, pages 2584–2591, 2005.

[110] O.M. Shir and T. Bäck. Niching in Evolution Strategies. Technical report, Leiden Institute of Advanced Computer Science, Leiden University, Leiden, The Netherlands, 2005.

[111] O.M. Shir and T. Bäck. Niching with Derandomized Evolution Strategies in Artificial and Real-World Landscapes. *Natural Computing*, 2008.

[112] G. Singh and K. Deb. Comparison of multi-modal optimization algorithms based on evolutionary algorithms. In M.Cattolico, editor, *Genetic and Evolutionary Computation Conference, GECCO 2006, Proceedings, Seattle, Washington, USA, July 8-12, 2006*, pages 1305–1312. ACM, 2006.

[113] R.E. Smith and J. Smith. An examination of tunable, random search landscapes. In W. Banzhaf and C.R. Reeves, editors, *Proceedings of the Fifth Workshop on Foundations of Genetic Algorithms (FOGA), Madison, WI, USA, 22-25 July 1998*, pages 165–181. Morgan Kaufmann, 1998.

[114] J.C. Spall. *Introduction to Stochastic Search and Optimization – Estimation, Simulation, and Control*. Wiley Press, 2003.

[115] B.M.R. Stadler and P.F. Stadler. The topology of evolutionary biology. In G. Ciobanu, editor, *Modeling in Molecular Biology*, Natural Computing Series, pages 267–286. Springer Verlag, 2004.

[116] C. Stoean, M. Preuss, R. Gorunescu, and D. Dumitrescu. Elitist generational genetic chromodynamics - a new radii-based evolutionary algorithm for multimodal optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2005, 2-4 September 2005, Edinburgh, UK*, pages 1839–1846. IEEE, 2005.

[117] F. Streichert, G. Stein, H. Ulmer, and A. Zell. A clustering based niching method for evolutionary algorithms. In E. Cantú-Paz et al., editor, *GECCO '03: Proceedings of the 2003 conference on Genetic and evolutionary computation*, pages 644–645, Berlin, 2003. Springer.

[118] H. Ulmer, F. Streichert, and A. Zell. Evolution strategies assisted by gaussian processes with improved pre-selection criterion. In *Proceedings of Congress on Evolutionary Computation (CEC'2003), Canberra, Australia, Piscataway, NY, December 8-12, 2003*, pages 692–699. IEEE, 2003.

[119] U. Utecht and K. Trint. Mutation operators for structure evolutionof neural networks. In et al. Y. Davidor, editor, *Parallel Problem Solving from Nature - PPSN III, International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature, Jerusalem, Israel, October 9-14, 1994, Proceedings*, volume 866 of *Lecture Notes in Computer Science*, pages 492–501. Springer, 1994.

[120] L. Vanneschi, G. Mauri, A. Valsecchi, and S. Cagnoni. Heterogeneous cooperative coevolution: strategies of integration between gp and ga. In M. Cattolico, editor, *Genetic and Evolutionary Computation Conference, GECCO 2006, Proceedings*, pages 361–368, Seattle, Washington, USA, 8 - 12, July 2006. ACM.

[121] E.D. Weinberger. NP Completeness of Kauffman's N-K Model, A Tuneable Rugged Fitness Landscape. Working Papers 96-02-003, Santa Fe Institute, Feb 1996.

[122] D.R. Wilson and T.R. Martinez. Improved heterogeneous distance functions. *Journal of Artificiall Intelligence Research*, 6(1):1–34, 1997.

[123] W.L. Winston. *Operation Research: Applications and Algorithms.* Wadsworth Publishing Company, 1997.

[124] S. Wright. The roles of mutation, inbreeding, crossbreeding and selection in evolution. In D.F. Jones, editor, *Proceedings of the sixth international congress on genetics.*, volume 1, pages 356–366, 1932.

[125] A.Y. Zhang. Bayesian mixed integer optimization using a-priori knowledge on variable dependences. Internal report, LIACS, Leiden University, Leiden, NL, August 2008.

[126] J. Ziegler and W. Banzhaf. Decreasing the number of evaluations in evolutionary algorithms by using a meta-model of the fitness function. In C. Ryan, T. Soule, M. Keijzer, E.P.K. Tsang, R. Poli, and E. Costa, editors, *Genetic Programming, 6th European Conference, EuroGP 2003, Essex, UK, April 14-16, 2003. Proceedings*, volume 2610 of *Lecture Notes in Computer Science*, pages 264–275. Springer, 2003.

Samenvatting

In natuurlijke systemen is het nastreven van een optimale toestand een heel belangrijk verschijnsel. Atomen, bijvoorbeeld, proberen optimale bindingen aan te gaan waardoor ze in een toestand van laagste energie geraken, levende mieren zijn in staat zich aan de veranderende omgeving aan te passen en de kortste weg te vinden van nest naar voedselbron, en de gezamenlijke beweging van een vogelzwerm doet de kans toenemen dat hun waakzaamheid tot succes leidt. Deze verbazingwekkende oplossingen van de hand van de natuur zijn voor wetenschappers en technici altijd al een bron van inspiratie geweest bij het aangaan van allerlei toepassingsuitdagingen in onze leefwereld. Natural computing is een onderzoeksgebied waarin gebruik wordt gemaakt van op de natuur geïnspireerde berekeningstechnieken en waarin algoritmen worden ontwikkeld voor het oplossen van problemen uit de reële wereld. Wij richten onze aandacht op evolutionary computation, op dit moment een van de gebieden in de informatica waarin de meeste onderzoeksaktiviteit gaande is, met een enorm aantal succesvolle toepassingen op problemen uit de reële wereld en met, voor sommige technieken, heel ver ontwikkelde theoretische onderbouwingen. In plaats van kenmerkende eigenschappen van afzonderlijke biologische organismen precies na te maken, ontleent evolutionary computing zijn inspiratie aan de dynamiek van hele populaties van organismen. Daarbij wordt gebruik gemaakt van begrippen als *mutatie*, *recombinatie* en *selectie*, om het *organische evolutie*proces na te bootsen, waarin *survival of the fittest, van de meest geschikte,* en *fenotypische variatie*principes een belangrijke rol spelen en leiden tot een een betere aanpassing van een populatie van individuen aan een gegeven evolutioniare omgeving. Dit betekent dat individuen met een grotere geschiktheid, fitness, dan ook betere kansen hebben op overleven en op nakomelingen. Het is in de litteratuur gebruikelijk om de hele verzameling van algoritmen die van dit *organische evolutie*proces zijn afgeleid, aan te duiden met evolutionaire algoritmen (EAs).

Het oorspronkelijke idee achter ons werk is, de canonieke EvolutieStrategieën (ESsen) uit het traditionele domein van optimalisering met reële parameters, uit te breiden naar het optimaliseringsdomein met mixed-integer parameters. Dit is nodig, omdat in het bedrijfsleven zich talrijke op de praktijk gerichte optimaliseringsproblemen voordoen waarbij de verzameling van beslissingsvariabelen continue, integerwaardige en anderszins discrete variabelen omvat. Bovendien zouden

doelfuncties voor dit type probleem gebaseerd kunnen worden op grootschalige simulatiemodellen, of ook zou de structuur van de doelfuncties te ingewikkeld kunnen zijn om in zo'n model op te nemen. Vanwege deze mogelijke complicaties wordt dit type optimaliseringsproblemen gecatalogiseerd als de categorie van *black-box*-optimaliseringen. Hierop kunnen de klassieke optimaliseringstechnieken, afkomstig uit het onderzoeksgebied van de Mathematische Programmering (MP), niet zo maar worden toegepast, omdat deze gebaseerd zijn op de eigenschap, dat met een *verdeel-en-heers*-aanpak de zoekruimte altijd efficiënt doorlopen kan worden. Daarentegen is het nieuwe algoritme dat wij voorstellen, de zogenoemde Mixed-Integer Evolution Strategies (MIES), heel wel in staat tot goede oplossingen te komen voor deze uitdagende *black-box*-optimaliseringsproblemen, namelijk door gebruikmaking van daartoe ontwikkelde variatie-operatoren toegespitst op klassen van mixed-integer parameters.

Binnen onze onderzoeksaktiviteiten hebben we niet alleen MIES geïtroduceerd en vanuit theoretisch standpunt diepgaand bestudeerd, maar we hebben ook een raamwerk ontwikkeld voor het toepassen van MIES op de optimaliseringsproblematiek uit de reële wereld van het medisch onderzoek. Meer in het bijzonder passen we MIES daar toe op de optimalisering van besturingsparameters van een semi-automatisch beeldanalysesysteem voor IntraVasculaire UltraSoundbeelden (IVUS). Dit zijn real-time, hoge-resolutie-tomografiebeelden die de binnenkant van een kransslagader laten zien of van andere slagaders. IVUS-beelden zijn lastig te interpreteren, wat er weer toe leidt dat handmatige segmentering in hoge mate gevoelig is voor geringe veranderingen door toedoen van een enkele waarnemer of door toedoen van het samenspel der waarnemers. Aldus heeft de ontwikkeling van een systeem voor het opsporen van karakteristieken in IVUS-beelden veel aandacht gekregen in het medisch onderzoek en in het informatica-onderzoek. De performance van de meeste systemen hangt echter af van een groot aantal besturingsparameters, die met de hand lastig te optimaliseren zijn, en die mede afhankelijk kunnen zijn van verschil in interpretatiecontext. Deze parameters zijn bovendien onderhevig aan verandering, als er in het registratieproces van de beelden iets wijzigt. Vergeleken met andere aanpakken kan er met MIES door de systeemontwikkelaar geautomatiseerd worden gezocht naar optimale parameterinstellingen, waarbij de kans groot is op het vinden van een parameterinstelling die resulteert in een significant hogere nauwkeurigheid bij het opsporen van de karakteristieken.

De inhoud van dit proefschrif bestaat uit drie delen: (1) de inleiding, en het theoretisch onderzoek aan het nieuwvoorgestelde optimaliseringsalgoritme; (2) het gebruik ervan bij toepassingen uit de reële wereld, en wel bij parameteroptimalisering in medische beeldanalyse; (3) geavanceerde onderwerpen zoals *Nichetechnieken*. Meer in het bijzonder worden in het theoriedeel de state-of-the-art MIES-algoritmen geïntroduceerd en vervolgens worden ze getest op verschillende, zorgvuldig ontworpen artificial landscapes, bijvoorbeeld op gegeneraliseerde NK landscapes. Het deel van de toepassingen uit de reële wereld gaat voornamelijk in op parameteroptimaliseringsproblemen uit het medisch onderzoeksgebied. De

door ons voorgestelde MIES-algoritmen worden toegepast, om een multi-agent-systeem te optimaliseren dat ontwikkeld was voor het opsporen van karakteristieken in medische beelden. Tevens worden enkele belangrijke waarnemingen uit de experimenten vermeld. Ten einde de performance van onze algoritmen nog meer te verbeteren, worden in het derde deel enkele geavanceerde technieken onderzocht die in combinatie met MIES kunnen worden gebruikt, bijvoorbeeld de technieken *Metamodel-Assisted Optimalisatie*, *Niche-Technieken* and *Bayesian Learning*.

In meer detail kan het proefschrift als volgt worden samengevat.

Hoofdstuk 2 geeft eerst een kort overzicht van de essentiële terminologie voor globale optimalisering en in het bijzonder introduceert het het mixed-integer parameteroptimalisatieprobleem. Verschillende klassieke algoritmen uit de traditionele Mathematische Programmering (MP) worden er besproken, zoals Branch-en-Bound-methoden (BB) and Outer-Approximation-methoden (OA). Tegenovergesteld aan deze *white-box* optimaliseringsaanpak wordt het raamwerk voor mixed-integer parameteroptimalisering binnen het *black-box* scenario besproken, en wel heel gedetailleerd. Ook worden twee representatieve toepassingen uit de reële wereld – *ontwerpen van optische filters* en *optimalisering van chemische fabrieken* – gegeven, als voorbeelden ter motivatie.

In Hoofdstuk 3 introduceren we eerst het algemene raamwerk van EAs. Daarna geven we een expliciete uitleg van de grondslagen van de canonieke ESsen, welke op hun beurt de kernen vormen van de algoritmen van de door ons voorgestelde aanpak MIES, gericht op mixed-integer-parameteroptimalisering. Vervolgens worden in detail de filosofie achter het ontwerp van MIES en verschillende belangrijke eigenschappen ervan besproken.

In Hoofdstuk 4 stellen we twee innovatieve, geconstrueerde testproblemen voor, *Barrier Functies* en *Mixed-Integer NK landscapes* (*MINKLs*). De barrier functies worden aangemaakt door een multimodale probleemgenerator die integeroptimaliseringsproblemen produceert met een schaalbare onregelmatigheidsgraad maar zonder interactie tussen de variabelen. MINKLs zijn uitbreidingen van standaard NK landscapes (NKLs), die zelf weer stochastisch gegenereerde pseudo-boolean functies zijn van N bits (de genen) en met K interacties tussen de genen. Deze twee kunstmatige testproblemen worden zorgvuldig ontworpen en de experimentele resultaten laten zien dat zij bijzonder nuttig zijn voor het begrijpen van de dynamiek van evolutionair zoeken binnen de mixed-integer toestandsruimte.

MIES toegepast op parameteroptimalisering van IVUS-beeldanalyse, wordt in Hoofdstuk 5 besproken. Er wordt een geavanceerd multi-agent-systeem geïntroduceerd dat bestemd is voor het opsporen van karakteristieken van IVUS-beelden, van lumen-karakteristieken in het bijzonder, en het raamwerk voor het optimaliseren ervan met behulp van MIES wordt uitgelegd samen met enkele veelbelovende experimentele resultaten.

In Hoofdstuk 6 onderzoeken we het gebruik van indelingen naar fitness, om groepen van Computed-Tomographic-Angiography-beelden (CTA) te kunnen vinden waarvoor een vergelijkbare parameterinstelling vereist is ten behoeve van

het segmenteringsalgoritme, terwijl deze parameterinstellingen voor de groepen tegelijkertijd blijven evolueren.

Hoofdstuk 7 bespreekt hoe metamodellen moeten worden gebruikt, radial-basis-function-netwerken (RBFN) met name, om MIES te ondersteunen bij het uitvoeren van optimaliseringstaken met tijdrovend gebruik van evaluatiefuncties, zoals analyse van IVUS-beelden.

Hoofdstuk 8 bespreekt een dynamische niche-techniek voor MIES die is gebaseerd op een bestaande ES niche-aanpak en die kort geleden ontwikkeld is en succesvol is toegepast op continuous landscapes. De nieuwe techniek is gebaseerd op de heterogene afstandsmaat die rekening houdt met overeenkomsten tussen toestandsruimten, en die in zekere zin consistent is met de mutatie-operatoren van MIES.

Hoofdstuk 9 introduceert een nieuw algoritme voor het schatten van verdelingen, dat een uitbreiding is van de toepasbaarheid van het Baysiaanse optimaliseringsalgoritme (met een vaste netwerkstructuur) en wel van binaire naar mixed-integer-optimaliseringsproblemen. Experimentele resultaten laten zien, dat door het hier voorgestelde algoritme a-priori-kennis van afhankelijkheden tussen beslissingsvariabelen ingezet kan worden ter verbetering van convergentiesnelheid en betrouwbaarheid. Het is binnen dit algoritme dat MIES als subalgoritme zijn werk doet in het zelf-organiserende clustering-proces.

Curriculum Vitae

Rui Li was born on November 29th 1978 in Urumqi, Xinjiang, People's Republic of China. He received his undergraduate academic degree in Software Engineering from Beijing Institute of Technology (BIT) in July 2000. He was appointed as a school counselor in the computer science department of BIT till the July 2002. In September 2002, he decided to go to the Netherlands to pursue his Master degree in Delft University of Technology (TU-Delft). He finished his graduate studies successfully and received his MSc in Media and Knowledge Engineering in July 2004. After that he joined the research group[1] of Prof. Thomas Bäck, and worked towards his PhD. His research project is funded by the Netherlands Organization for Scientific (NWO) and is carried out through co-operation between LIACS and LKEB[2]. During his four years PhD research, Rui Li has co-authored several papers in peer-reviewed conferences. He got the Best Paper Award Nomination in the 8th European workshop on evolutionary computation in image analysis and signal processing (EvoIASP) for the work on "Mixed-Integer Evolution Strategies and Their Application to Intravascular Ultrasound Image Analysis". In April of 2008, his work on "Optimizing Computed Tomographic Angiography Image Segmentation using Fitness Based Partitioning" received the Best Paper Award in the 10th European workshop on evolutionary computation in image analysis and signal processing (EvoIASP). Besides his research, he was also a teaching assistant for several courses in LIACS, including, databases and computer architecture.

[1] Natural Computing Group in Leiden Institute of Advanced Computer Science (LIACS), Leiden University.

[2] The Division of Image Processing, the Department of Radiology, Leiden University Medical Center.

Acknowledgement

During my four years PhD research, there are many people whom I would like to take this opportunity to thank.

Let me begin by thanking Prof. Thomas Bäck and Dr. Michael Emmerich. I own a deep gratitude to them, for their excellent inspiration and supervision all along my research.

Particularly, I would like to thank my "SAVAGE" team-mates, Dr. Ernst Bovenkamp at TNO and Dr. Jeroen Eggermont at LUMC. I really enjoyed the cooperation with them during these years and I learned a lot about Medical Image Processing from them.

I thank Dr. Peter Lucas at Radboud University Nijmegen and Dr. Ildikó Flesch at Tilburg University for their fruitful discussion on Bayesian Network issues.

I would also like to thank Dr. Luuk Groenewegen for his help with compiling the Dutch Samenvatting.

Research is only a part of my life, I appreciate the cultural diversity at LIACS and enjoyed a lot casual chatting with my colleagues at LIACS. Special thanks go to my ex-roommates Julia Dmitrieva, Yun Bei, and my current roommate Johannes Kruisselbrink. I am also grateful to my friends who helped me all the time.

Last but not least, I would like to thank my beloved family, especially my wife Bin. It is impossible for me to finish this thesis without their endless love and support.

Titles in the IPA Dissertation Series since 2005

E. Ábrahám. *An Assertional Proof System for Multithreaded Java - Theory and Tool Support-* . Faculty of Mathematics and Natural Sciences, UL. 2005-01

R. Ruimerman. *Modeling and Remodeling in Bone Tissue.* Faculty of Biomedical Engineering, TU/e. 2005-02

C.N. Chong. *Experiments in Rights Control - Expression and Enforcement.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-03

H. Gao. *Design and Verification of Lock-free Parallel Algorithms.* Faculty of Mathematics and Computing Sciences, RUG. 2005-04

H.M.A. van Beek. *Specification and Analysis of Internet Applications.* Faculty of Mathematics and Computer Science, TU/e. 2005-05

M.T. Ionita. *Scenario-Based System Architecting - A Systematic Approach to Developing Future-Proof System Architectures.* Faculty of Mathematics and Computing Sciences, TU/e. 2005-06

G. Lenzini. *Integration of Analysis Techniques in Security and Fault-Tolerance.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-07

I. Kurtev. *Adaptability of Model Transformations.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-08

T. Wolle. *Computational Aspects of Treewidth - Lower Bounds and Network Reliability.* Faculty of Science, UU. 2005-09

O. Tveretina. *Decision Procedures for Equality Logic with Uninterpreted Functions.* Faculty of Mathematics and Computer Science, TU/e. 2005-10

A.M.L. Liekens. *Evolution of Finite Populations in Dynamic Environments.* Faculty of Biomedical Engineering, TU/e. 2005-11

J. Eggermont. *Data Mining using Genetic Programming: Classification and Symbolic Regression.* Faculty of Mathematics and Natural Sciences, UL. 2005-12

B.J. Heeren. *Top Quality Type Error Messages.* Faculty of Science, UU. 2005-13

G.F. Frehse. *Compositional Verification of Hybrid Systems using Simulation Relations.* Faculty of Science, Mathematics and Computer Science, RU. 2005-14

M.R. Mousavi. *Structuring Structural Operational Semantics.* Faculty of Mathematics and Computer Science, TU/e. 2005-15

A. Sokolova. *Coalgebraic Analysis of Probabilistic Systems.* Faculty of Mathematics and Computer Science, TU/e. 2005-16

T. Gelsema. *Effective Models for the Structure of pi-Calculus Processes with Replication.* Faculty of Mathematics and Natural Sciences, UL. 2005-17

P. Zoeteweij. *Composing Constraint Solvers.* Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2005-18

J.J. Vinju. *Analysis and Transformation of Source Code by Parsing and Rewriting.* Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2005-19

M.Valero Espada. *Modal Abstraction and Replication of Processes with Data.* Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2005-20

A. Dijkstra. *Stepping through Haskell.* Faculty of Science, UU. 2005-21

Y.W. Law. *Key management and link-layer security of wireless sensor networks: energy-efficient attack and defense.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-22

E. Dolstra. *The Purely Functional Software Deployment Model.* Faculty of Science, UU. 2006-01

R.J. Corin. *Analysis Models for Security Protocols.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2006-02

P.R.A. Verbaan. *The Computational Complexity of Evolving Systems.* Faculty of Science, UU. 2006-03

K.L. Man and R.R.H. Schiffelers. *Formal Specification and Analysis of Hybrid Systems.* Faculty of Mathematics and Computer Science and Faculty of Mechanical Engineering, TU/e. 2006-04

M. Kyas. *Verifying OCL Specifications of UML Models: Tool Support and Compositionality.* Faculty of Mathematics and Natural Sciences, UL. 2006-05

M. Hendriks. *Model Checking Timed Automata - Techniques and Applications.* Faculty of Science, Mathematics and Computer Science, RU. 2006-06

J. Ketema. *Böhm-Like Trees for Rewriting.* Faculty of Sciences, VUA. 2006-07

C.-B. Breunesse. *On JML: topics in tool-assisted verification of JML programs.* Faculty of Science, Mathematics and Computer Science, RU. 2006-08

B. Markvoort. *Towards Hybrid Molecular Simulations.* Faculty of Biomedical Engineering, TU/e. 2006-09

S.G.R. Nijssen. *Mining Structured Data.* Faculty of Mathematics and Natural Sciences, UL. 2006-10

G. Russello. *Separation and Adaptation of Concerns in a Shared Data Space.* Faculty of Mathematics and Computer Science, TU/e. 2006-11

L. Cheung. *Reconciling Nondeterministic and Probabilistic Choices.* Faculty of Science, Mathematics and Computer Science, RU. 2006-12

B. Badban. *Verification techniques for Extensions of Equality Logic.* Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2006-13

A.J. Mooij. *Constructive formal methods and protocol standardization.* Faculty of Mathematics and Computer Science, TU/e. 2006-14

T. Krilavicius. *Hybrid Techniques for Hybrid Systems*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2006-15

M.E. Warnier. *Language Based Security for Java and JML*. Faculty of Science, Mathematics and Computer Science, RU. 2006-16

V. Sundramoorthy. *At Home In Service Discovery*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2006-17

B. Gebremichael. *Expressivity of Timed Automata Models*. Faculty of Science, Mathematics and Computer Science, RU. 2006-18

L.C.M. van Gool. *Formalising Interface Specifications*. Faculty of Mathematics and Computer Science, TU/e. 2006-19

C.J.F. Cremers. *Scyther - Semantics and Verification of Security Protocols*. Faculty of Mathematics and Computer Science, TU/e. 2006-20

J.V. Guillen Scholten. *Mobile Channels for Exogenous Coordination of Distributed Systems: Semantics, Implementation and Composition*. Faculty of Mathematics and Natural Sciences, UL. 2006-21

H.A. de Jong. *Flexible Heterogeneous Software Systems*. Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2007-01

N.K. Kavaldjiev. *A run-time reconfigurable Network-on-Chip for streaming DSP applications*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2007-02

M. van Veelen. *Considerations on Modeling for Early Detection of Abnormalities in Locally Autonomous Distributed Systems*. Faculty of Mathematics and Computing Sciences, RUG. 2007-03

T.D. Vu. *Semantics and Applications of Process and Program Algebra*. Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2007-04

L. Brandán Briones. *Theories for Model-based Testing: Real-time and Coverage*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2007-05

I. Loeb. *Natural Deduction: Sharing by Presentation*. Faculty of Science, Mathematics and Computer Science, RU. 2007-06

M.W.A. Streppel. *Multifunctional Geometric Data Structures*. Faculty of Mathematics and Computer Science, TU/e. 2007-07

N. Trčka. *Silent Steps in Transition Systems and Markov Chains*. Faculty of Mathematics and Computer Science, TU/e. 2007-08

R. Brinkman. *Searching in encrypted data*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2007-09

A. van Weelden. *Putting types to good use*. Faculty of Science, Mathematics and Computer Science, RU. 2007-10

J.A.R. Noppen. *Imperfect Information in Software Development Processes*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2007-11

R. Boumen. *Integration and Test plans for Complex Manufacturing Systems*. Faculty of Mechanical Engineering, TU/e. 2007-12

A.J. Wijs. *What to do Next?: Analysing and Optimising System Behaviour in Time*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2007-13

C.F.J. Lange. *Assessing and Improving the Quality of Modeling: A Series of Empirical Studies about the UML*. Faculty of Mathematics and Computer Science, TU/e. 2007-14

T. van der Storm. *Component-based Configuration, Integration and Delivery*. Faculty of Natural Sciences, Mathematics, and Computer Science,UvA. 2007-15

B.S. Graaf. *Model-Driven Evolution of Software Architectures*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2007-16

A.H.J. Mathijssen. *Logical Calculi for Reasoning with Binding*. Faculty of Mathematics and Computer Science, TU/e. 2007-17

D. Jarnikov. *QoS framework for Video Streaming in Home Networks*. Faculty of Mathematics and Computer Science, TU/e. 2007-18

M. A. Abam. *New Data Structures and Algorithms for Mobile Data*. Faculty of Mathematics and Computer Science, TU/e. 2007-19

W. Pieters. *La Volonté Machinale: Understanding the Electronic Voting Controversy*. Faculty of Science, Mathematics and Computer Science, RU. 2008-01

A.L. de Groot. *Practical Automaton Proofs in PVS*. Faculty of Science, Mathematics and Computer Science, RU. 2008-02

M. Bruntink. *Renovation of Idiomatic Crosscutting Concerns in Embedded Systems*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2008-03

A.M. Marin. *An Integrated System to Manage Crosscutting Concerns in Source Code*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2008-04

N.C.W.M. Braspenning. *Model-based Integration and Testing of High-tech Multi-disciplinary Systems*. Faculty of Mechanical Engineering, TU/e. 2008-05

M. Bravenboer. *Exercises in Free Syntax: Syntax Definition, Parsing, and Assimilation of Language Conglomerates*. Faculty of Science, UU. 2008-06

M. Torabi Dashti. *Keeping Fairness Alive: Design and Formal Verification of Optimistic Fair Exchange Protocols*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2008-07

I.S.M. de Jong. *Integration and Test Strategies for Complex Manufacturing Machines*. Faculty of Mechanical Engineering, TU/e. 2008-08

I. Hasuo. *Tracing Anonymity with Coalgebras*. Faculty of Science, Mathematics and Computer Science, RU. 2008-09

L.G.W.A. Cleophas. *Tree Algorithms: Two Taxonomies and a Toolkit*. Faculty of Mathematics and Computer Science, TU/e. 2008-10

I.S. Zapreev. *Model Checking Markov Chains: Techniques and Tools.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-11

M. Farshi. *A Theoretical and Experimental Study of Geometric Networks.* Faculty of Mathematics and Computer Science, TU/e. 2008-12

G. Gulesir. *Evolvable Behavior Specifications Using Context-Sensitive Wildcards.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-13

F.D. Garcia. *Formal and Computational Cryptography: Protocols, Hashes and Commitments.* Faculty of Science, Mathematics and Computer Science, RU. 2008-14

P. E. A. Dürr. *Resource-based Verification for Robust Composition of Aspects.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-15

E.M. Bortnik. *Formal Methods in Support of SMC Design.* Faculty of Mechanical Engineering, TU/e. 2008-16

R.H. Mak. *Design and Performance Analysis of Data-Independent Stream Processing Systems.* Faculty of Mathematics and Computer Science, TU/e. 2008-17

M. van der Horst. *Scalable Block Processing Algorithms.* Faculty of Mathematics and Computer Science, TU/e. 2008-18

C.M. Gray. *Algorithms for Fat Objects: Decompositions and Applications.* Faculty of Mathematics and Computer Science, TU/e. 2008-19

J.R. Calamé. *Testing Reactive Systems with Data - Enumerative Methods and Constraint Solving.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-20

E. Mumford. *Drawing Graphs for Cartographic Applications.* Faculty of Mathematics and Computer Science, TU/e. 2008-21

E.H. de Graaf. *Mining Semi-structured Data, Theoretical and Experimental Aspects of Pattern Evaluation.* Faculty of Mathematics and Natural Sciences, UL. 2008-22

R. Brijder. *Models of Natural Computation: Gene Assembly and Membrane Systems.* Faculty of Mathematics and Natural Sciences, UL. 2008-23

A. Koprowski. *Termination of Rewriting and Its Certification.* Faculty of Mathematics and Computer Science, TU/e. 2008-24

U. Khadim. *Process Algebras for Hybrid Systems: Comparison and Development.* Faculty of Mathematics and Computer Science, TU/e. 2008-25

J. Markovski. *Real and Stochastic Time in Process Algebras for Performance Evaluation.* Faculty of Mathematics and Computer Science, TU/e. 2008-26

H. Kastenberg. *Graph-Based Software Specification and Verification.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-27

I.R. Buhan. *Cryptographic Keys from Noisy Data Theory and Applications.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-28

R.S. Marin-Perianu. *Wireless Sensor Networks in Motion: Clustering Algorithms for Service Discovery and Provisioning*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-29

M.H.G. Verhoef. *Modeling and Validating Distributed Embedded Real-Time Control Systems*. Faculty of Science, Mathematics and Computer Science, RU. 2009-01

M. de Mol. *Reasoning about Functional Programs: Sparkle, a proof assistant for Clean*. Faculty of Science, Mathematics and Computer Science, RU. 2009-02

M. Lormans. *Managing Requirements Evolution*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2009-03

M.P.W.J. van Osch. *Automated Model-based Testing of Hybrid Systems*. Faculty of Mathematics and Computer Science, TU/e. 2009-04

H. Sozer. *Architecting Fault-Tolerant Software Systems*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2009-05

M.J. van Weerdenburg. *Efficient Rewriting Techniques*. Faculty of Mathematics and Computer Science, TU/e. 2009-06

H.H. Hansen. *Coalgebraic Modelling: Applications in Automata Theory and Modal Logic*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2009-07

A. Mesbah. *Analysis and Testing of Ajax-based Single-page Web Applications*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2009-08

A.L. Rodriguez Yakushev. *Towards Getting Generic Programming Ready for Prime Time*. Faculty of Science, UU. 2009-9

K.R. Olmos Joffré. *Strategies for Context Sensitive Program Transformation*. Faculty of Science, UU. 2009-10

J.A.G.M. van den Berg. *Reasoning about Java programs in PVS using JML*. Faculty of Science, Mathematics and Computer Science, RU. 2009-11

M.G. Khatib. *MEMS-Based Storage Devices. Integration in Energy-Constrained Mobile Systems*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2009-12

S.G.M. Cornelissen. *Evaluating Dynamic Analysis Techniques for Program Comprehension*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2009-13

D. Bolzoni. *Revisiting Anomaly-based Network Intrusion Detection Systems*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2009-14

H.L. Jonker. *Security Matters: Privacy in Voting and Fairness in Digital Exchange*. Faculty of Mathematics and Computer Science, TU/e. 2009-15

M.R. Czenko. *TuLiP - Reshaping Trust Management*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2009-16

T. Chen. *Clocks, Dice and Processes*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2009-17

C. Kaliszyk. *Correctness and Availability: Building Computer Algebra on*

top of Proof Assistants and making Proof Assistants available over the Web. Faculty of Science, Mathematics and Computer Science, RU. 2009-18

R.S.S. O'Connor. *Incompleteness & Completeness: Formalizing Logic and Analysis in Type Theory*. Faculty of Science, Mathematics and Computer Science, RU. 2009-19

B. Ploeger. *Improved Verification Methods for Concurrent Systems*. Faculty of Mathematics and Computer Science, TU/e. 2009-20

T. Han. *Diagnosis, Synthesis and Analysis of Probabilistic Models*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2009-21

R. Li. *Mixed-Integer Evolution Strategies for Parameter Optimization and Their Applications to Medical Image Analysis*. Faculty of Mathematics and Natural Sciences, UL. 2009-22