Andreas Petrow

Constraint-Handling Techniques for highly Constrained Optimization

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

INF FAKULTÄT FÜR INFORMATIK

Intelligent Cooperative Systems
Computational Intelligence

Constraint-Handling Techniques for highly Constrained Optimization

Master Thesis

Andreas Petrow

March 22, 2019

Supervisor:	Prof. Dr.-Ing. habil. Sanaz Mostaghim
Advisor:	M.Sc. Heiner Zille
Advisor:	Dr.-Ing. Markus Schori

Andreas Petrow: *Constraint-Handling Techniques for highly Constrained Optimization*
Otto-von-Guericke Universität
Intelligent Cooperative Systems
Computational Intelligence
Magdeburg, 2019.

Abstract

This thesis relates to constraint handling techniques for highly constrained optimization. Many real world optimization problems are non-linear and restricted to constraints. To solve non-linear optimization problems population-based metaheuristics are used commonly. To enable such optimization methods to fulfill the optimization with respect to the constraints modifications are required. Therefore a research field considering such constraint handling techniques establishes. However the combination of large-scale optimization problems with constraint handling techniques is rare. Therefore this thesis involves with common constraint handling techniques and evaluates these on two benchmark problems.

Contents

List of Figures

List of Tables

List of Acronyms

CHT Constraint Handling Technique

OM Optimization Method

INIT Initialization

n_x number of variables

n_c number of constraints

f objective function

$\mathcal{A}$ constraint coefficents

$\mathcal{S}$ Search Space

F Feasible Domain

QP Quadratic Programming

LP Linear Programming

RaI Ramdom Initialization

ReI Repaired Initialization

GS Gibb's Sampling

OB on boundaries

avc$_p$ *penalisation with the amount of violated constraints*

ncv$_p$ *penalisation with the norm of constraint violation*

F&W	Farmani and Wright
avc$_s$	Bi-objective optimization problem with the amount of violated constraints
ncv$_s$	Bi-objective optimization problem with the norm of constraint violation
DR	Deb's rules
RC	feasibility preserving CHT-Repair
DC	feasibility preserving CHT-Decoder
PSO	Particle Swarm Optimization
DE	Differential Evolution
GA	Genetic Algorithm
RW	Random Walk
NSGA-II	Non-dominated Sorting Genetic Algorithm II
avc	amount of violated constraints
ncv	norm of constraint violation
d2o	distance to the global optimum
obj	objective
dbs	distance between solutions
%_f	percentage of feasible solutions

1. Introduction and Motivation

As a result of growing influence of technology in everyday life our world becomes more complex. With the widespread accessible information acquisition now more complicate problems can be attacked. In particular, in the last decades developed *Optimization Method* (OM) are going to be applied to new subject areas like chemistry, physics, insurance or automotive.

After the mathematical definition, optimization is the finding of the best global solution. However by far not all problems provide a mathematical definition at which the global optimum can be calculated either directly or iteratively by following a gradient. Such problems have usually a nonlinear evaluation function.
In this cases meta-heuristics become valuable. Even if a found solution is not guaranteed to be the global optimum nevertheless it can be good enough related to some criteria. Many widespread meta-heuristics are population-based and rely mostly on *Evolutionary Algorithms* and *Particle Swarm Optimization*.

Common optimization problems with a nonlinear evaluation function come from the field of *Engineering Design*. These are mostly *Parameter Optimization*. Thereby the optimal parametrization of a product is asked. Such parameters may be the thickness of the wall of a pressure vessel or the diameter of a gearwheel. Also the ranges of values the parameters can assume are usually constrained. Constraints in such scenarios are physical limitations, maximal possible values, withstanding a certain force, the pressure for example by a pressure vessel, or preventing components leaving a restricted design space, in case the diameter of the cogwheel of a wristwatch should not be larger then the watch itself. Optimizing of all parameters of each component may span a huge search space. In addition fulfilling all constraints increases the difficulty of the optimization process.

To handle constraints and direct an optimization process towards the feasible domain with a population-based meta-heuristic diverse methods have been developed referred in the literature to as *Constraint Handling Technique* (CHT)
Over the last decades population-based meta-heuristics have become very popular

and a massive diversity of different approaches based on them arise. With them also the amount of CHT increases. Carlos A. Coello Coello collectes some of the papers addressing CHT in a list online containing over 1400 entries to the time this thesis is written[1]. Despite the seemingly great interest in this kind of research, the *holy grail of CHT* has not yet been reached, which independently of the optimizer or the problem, reaches the feasible domain and finds an optimal solution. Furthermore there is less research with bigger problems containing more constraints then decision variables. Optimizations considering such unconstrained bigger problems are referred to as large-scale optimization in the literature [60]. A combination of large-scale optimization with CHTs is a rare research topic and will be investigated by this thesis further.

1.1. Aim of this thesis

Due to the small number of investigations of high dimensional problems containing many decision variables with even more constraints, this work serves as a first investigation step towards this direction. The objective function of the considered optimization problems is uncontinuous and will be treated as a blackbox. In this respect three research questions will be examined in more detail.

- How does known and popular CHT perform with such a kind of problems?
- How does different initialization technique influence the overall optimization process?
- Which combination of CHT, initialization technique and optimization method performs best and why?

To estimate how well a CHT performs on the considered problems some representatives of different concepts of CHT will be evaluated.

In addition each analyzed CHT will be tested with different initialization methods. This may play a major role since exploration of the huge size of the search space depends either on a well distributed starting population or the well balancing of the OM between exploration and exploitation.

Furthermore the combination of CHT, OM and an initialization technique may operate either synergistically or counterproductive[47].

[1] `https://www.cs.cinvestav.mx/~constraint/`

1.2. Structure of this thesis

Finally for this chapter the structure of this thesis will be presented containing the chapter headline and a short description of their content.

Basics functions as an introduction to the subject of this thesis comprising some theoretical basic knowledge starting with constrained optimization problems in general (Section 2.1) and going on to optimization methods (Section 2.2). In particular classical (Section 2.2.1) and population-based optimization methods (Section 2.2.2) are described briefly.

Related Work summarizes different constraint handling techniques for population-based meta-heuristics into four categories (*Penalty Methods* Section 3.1, *Separatist approaches* Section 3.2 and *Hybrids & Others* Section 3.4). It also provides an overview of explicit examples associated to the categories. The overview includes the number of dimensions and constraints in the benchmark-problems of the presented papers.

Considered Class of Problems gives the exact definition of the problem including its analysis and resulting characteristics (Section 4.1). It also describes the preprocessing steps before the given problem is optimized (Section 4.2). Here two representative problem definitions are introduced which are optimized by different approaches and evaluated in the following.

Approaches contains the investigated approaches. This includes the initialization of the start-population (Section 5.1), the selected constraint handling techniques (Section 5.2) and the population-based meta-heuristics (Section 5.3).

Evaluation describes which test-cases per problem (Section 6.1.1 and 6.2.1) consisting of an initialization method, a constraint handling technique and a meta-heuristic for optimization are evaluated. In addition, Sections 6.1.2 and 6.2.2 present the results of the approaches per problem respectively in relation to some predefined metrics. The last Section 6.3 compares the results of the approaches per problem.

Conclusions and Future Work as the final chapter summarizes the results (Section 7.1) and gives a hypothetical outlook about the topic of the presented work that require further research (Section 7.2).

2. Basics

This chapter describes basic knowledge needed for a clear understanding and as an introduction of the considered topic in this thesis.

2.1. Constrained Optimization Problem

Optimization problems are the finding of an optimal best solution regarding a criterion in a predefined variable space. A general description of constrained optimization problem can be viewed in the Equation 2.1.

$$\begin{aligned} &\underset{\vec{x}}{min/max} && f(\vec{x}) && \\ &\text{subject to :} && h_i(\vec{x}) = 0 && i = 1, \ldots, n_h \\ & && g_j(\vec{x}) \leq 0 && j = 1, \ldots, n_g \end{aligned} \tag{2.1}$$

The criterion which is wished to be optimized has to be evaluable through an objective function (f), with $f : \mathbb{D}^{n_x} \rightarrow \mathbb{R}^m, n_x \geq 1, m \geq 1$. Otherwise here are no limitations for the definition of f. It can take any mathematically characteristics, e.g. linear, convex, etc. or even used as a blackbox, as long as it assigns a value out of the objective space $\mathbb{R}^m$ to a valid solution $x \in \mathbb{D}^{n_x}$.
In case of the objective space being $\mathbb{R}^m$ with $m \geq 2$ the procedure is called a multi-objective optimization and as single-objective with $m = 1$.
Valid solutions are elements in the variable space $\mathbb{D}^{n_x}$. The variable space can be either discrete, continuous or a combination of these. The dimensionality of a valid solution inside ot the variable space is ether fix or may vary depending on the problem.
The optimization problem is written down as a minimization or maximization of f. Whereby each maximization can be transfered into a minimization problem by multiplying f with -1 and vice versa. The global optimal solution $\vec{x}^*$ is found if $f(\vec{x}^*) \leq f(\vec{x}) \;\; \forall \vec{x} \in \mathbb{D}^{n_x}$ is met.

Many real world problems are not solvable by all solutions of the variable space. Rather they are restricted to constraints. Therefore the definition of the optimization problem may be expanded (see "subject to: $h_i(\vec{x})$ and $g_j(\vec{x})$" in Equation 2.1). Generally constraints are defined as equalities (h_i) and inequalities (g_i). These functions (h_i and g_i) may take any mathematically expression. Solution satisfying all constraints are denoted as feasible otherwise infeasible.
It is a common method to transfer the equalities into inequalities so that just one kind of constraints has to be handled. One common way to redefine them is described in Equation 2.2 [25].

$$h^* = |\, h(\vec{x})\, | - \epsilon \;\leq\; 0 \tag{2.2}$$

Where ϵ is a small number implemented as a acceptable tolerance. Beside the mathematically characteristics also other properties are considered for a classification.
In case the feasibility of a solution can be computed before the evaluation by f the constraints are *a priori* known. Otherwise the solutions feasibility will be checked during the evaluation. In that case the constraints are only known *posteriori*. Examples for *posteriori* evaluable constraints can be simulations or artificial intelligences playing a game and not allowed to die.

2.2. Optimization Method

To solve a given optimization problem there is a wide range of different methods and algorithms. Depending on the characteristics of the given problem several approaches may perform worse then others or even are not able to find an optimal solution.
An ideal case is if the optimal solution can be computed directly by mathematical calculation. Since most problems are not solvable straightforward many methods find optimal solution iteratively by means of an gradient of the objective function. Such optimization methods are introduced shortly in the following subsection.
In some cases a gradient for the objective function does not exist. This may appear if the objective function is not continuous or even handled as a blackbox. Nevertheless there exist a type of optimization methods which are less limited by specific characteristics of the objective function. These kind of optimization methods are meta-heuristics. In contrast to iterative methods or other optimization algorithms they do not guarantee to find the globally optimal solution. A subcategory besides

others of meta-heuristics are population-based optimization methods which are described in more detail in the subsection after next and are also investigated in this thesis regarding to the previously defined class of problems.

2.2.1. Classical optimization

Some subgroups of constrained optimization problems are already well investigated. Two common and well researched subgroups are LP and QP. Since this thesis focuses on blackboxed objective functions these subgroups are mentioned only shortly. Further information are available by the references.
In the case that the objective function and the constraints are linear the subgroup over this kind of optimization problems is called *Linear Programming* (LP) or also referred to as *Linear Optimization*. Such a kind of problems can be solved iteratively by optimization methods also called *solvers*. The most common solvers for Linear Programming (LP) are *Interior-Point* methods [59] and *Simplex Algorithms* [28].
Another well researched subgroup of optimization problems is *Quadratic Programming* (QP). Hereby the objective function is quadratic. These are either linear constrained or unconstrained. The *interior-point* method can be adapted to solve QP also. Other common solver are the *Augmented Langragian* methods [19].

2.2.2. Population-based Optimization Methods

The main idea of population-based optimization methods, also referred to as population-based meta-heuristics, is to explore the search space not by a single solution but rather with multiple candidates contained in a population. The search is be guided by evaluating the current population. The guidance can be realized by modifying, replacing or combining candidates. The number of population-based optimization methods has been increasing during the last decades[54, 4]. Many of them are nature-inspired that mimic some behavior of ants [16], bees [26], glowworms [33], fireflies [58], trees and seeds [31] or by the evolution itself [41].
Two common representatives are *particle swarm optimization* and *evolutionary algorithm*. Both are described with their general declaration in the following.

Particle Swarm Optimization

Particle Swarm Optimization (PSO) pertains to the category of *swarm intelligence*. In this category each candidate is handled as a self-organized agent. They move through the search space based on information about it gathered by themselves and the population/swarm.
Since the introduction of PSO by James Kennedy and Russell Eberhart [29] in the year 1995 many different variations have been developed [49]. This PSO is defined in Equation 2.3.

$$\begin{aligned} \vec{v}_{i+1} &= \omega\vec{v}_i + \eta_1 r_1(\vec{x}_i - \vec{x}_i^l) + \eta_2 r_2(\vec{x}_i - \vec{x}_i^g) \\ \vec{x}_{i+1} &= \vec{x}_i + \vec{v}_{i+1} \end{aligned} \tag{2.3}$$

Each agent maintains its own memory. This consists of a position in the search space $\vec{x}_i$ and a velocity $\vec{v}_i$. Both are updated per iteration i. The point $\vec{x}_i^l$ defines the position with the best objective found by the particle so far. The best solution $\vec{x}_i^g$ in a certain neighborhood attracts the particle to another in case its objective is better. This influence is also referred to as *social component*. The parameters ω, η_1 and η_2 are constants to control influences of the velocity and do not change during the optimization. The variables r_1 and r_2 are random values between 0 and 1.

Evolutionary Computation

Evolutionary Computation is a class of algorithms related to stochastic meta-heuristic optimization methods. This kind of approaches is inspired by the biological evolution. Starting with an initial population of candidate solutions also referred to as individuals in this context, a variety of biological operations are applied. These operations include mechanisms like reproduction, mutation, recombination, selection and survival of the fittest. Those are iteratively applied to population, whereby these iteration steps are also usually described as generations. Algorithm 1 shows a general scheme of a Genetic Algorithm (GA).

Algorithm 1: Generalized scheme of a genetic algorithm

```
Input: $f$ as an objective function, $S$ as a search space describing the set of
       possible solutions
Output: $\vec{x}^*$ as the best found solution
1 $pop$ ← initialize start-population in $S$
2 while $termination\ criteria$ not fulfilled do
3     $parents$ ← selection($pop$, $f$)
4     $offspring$ ← recombination($parents$)
5     $offspring$ ← mutate($offspring$)
6     $fitness$ ← evaluate($pop \cup offspring$ with $f$)
7     $pop$ ← EnvironmentSelection($pop \cup offspring$, $fitness$)
8 return $\vec{x}^*$ as best solution in $pop$
```

The first step of every genetic algorithm is the initialization of a start-population. Thereby is each individual defined by its chromosome consisting of genes. Therefore the population *pop* contains the individuals described by their genotype. After this initialization step the optimization processes as long a termination criteria is not fulfilled. This criteria can be a predefined maximal amount of generations, a minimal change of the populations mean fitness or others. To evolve the next generation some parents out of the current population have bo be chosen by a selection mechanism. *Tournament*-selection, *Roulette-Wheel* or simply selecting the better individuals according to their fitness are some examples for such a selection mechanism. The fitness is evaluated on the decoded chromosomes of the individuals. All the other operation are processed on the genotype of the individuals. This representation of the individuals is also referred to as the phenotype. An offspring is created out of the selected parents by recombination. This recombination of parents to new individuals can vary widely. It is also referred to as a *crossover* operation. Also it depends on the explicit variation of the genetic algorithm. Exemplary *Differential Evolution* (DE) takes three individuals as parents to create a single new individual for the offspring. Thereby are two individuals merged to create modification vector which is applied to the third picked parent. Detailed information about DE can be viewed in [46]. After the offspring has been created by the recombination of their parents its chromosomes are mutated by a predefined percentage. How this mutation is implemented can also vary, either by a degree how many genes are going to be mutated or an intensity how strong does a mutation change the gene. Afterwards the population and its offspring are evaluated by the objective function f. With the evaluated individuals again a selection is applied to reduce amount of individuals for the next generation to certain amount and ideally keep better individuals in the population. If the termination criteria is fulfilled the best individual $\vec{x}$ is returned as the result of the optimization process.

Process an optimization satisfying all constraints is not a trivial task. Nevertheless there exist methods tackling this issue. These are collected as *Constraint Handling Techniques* (CHT). The next chapter deals with CHTs and its subcategories in detail.

3. Related Work

Solving a given optimization problem which includes constraints increases the complexity of the optimization process with regard to ensuring satisfaction of the constraints. This chapter is meant to give an overview of different concepts of *Constraint Handling Technique* (CHT), including examples of *state-of-the-art* approaches following these concepts. In the literature CHT are categorized in different ways [44, 50, 25]. This work orients on the categorization similar to the four concepts described in [13, 1]. The following sections describe each category with some subcategories. There exists a wide range of different subcategories of CHT per chosen category. Therefore only the main represented in the literature are mentioned in subsections of the following sections.

The most common idea to handle constrained optimization problems is to transfer them into unconstrained ones. This may be done by including the constraint violation in the objective function. In this way infeasible solutions are penalized depending on a degree of constraints violation. This kind of constraint-handling technique is denoted as *penalty functions methods* and will be described in the next section.

Alternatively the infeasible solutions are processed differently in comparison to the feasible solutions. The CHT of separation of objective and constraints is referred to as *separatist method*. An overview of different approaches following this idea is given in Section 3.2.

Another concept is to keep the solutions inside the feasible domain. There are approaches which implement this concept in different methods. Section 3.3 explains the idea in detail and its subsections contain the different manners.

Beside these diverse concepts researcher developed and tested hybrid methods of these. Also some techniques are not directly correlated to one of the presented concepts. Some of these are mentioned in the section 3.4.

The last section in this chapter (Section 3.5) gives an overview of several CHT with their correlated concept and a few selected properties on the conducted benchmark tests.

3.1. Penalty Methods

The concept of *penalty methods* is to decrease the fitness value of solutions depending on their position in the search space. There exist two types of penalty functions. On the one hand there are *interior* penalty functions which intend to keep feasible solutions inside the feasible domain by penalizing them if they get closer to the borders. These are also referred to as barrier [52]. Such a type of penalty functions require a feasible initialization and are not commonly used [2].
On the other hand there is the *exterior* paradigm which penalizes infeasible solutions. This type allows starting positions outside of the feasible domain and attracts the solutions towards the feasible domain. The implementation of these CHT is realized by modifying the original cost function. Usually a penalty factor calculated by a penalty function $p(\vec{x})$ is added to the original objective value (see Equation 3.1). There are plenty ways to realize such a modification of the original cost function by $\varphi(\vec{x})$.

$$\min_{\vec{x}} \quad \varphi(\vec{x}) = f(\vec{x}) + p(\vec{x}) \tag{3.1}$$

3.1.1. Static Penalty Functions

This category chooses a fixed strategy of how to penalize an infeasible solution. A widespread static penalty function is the *death penalty*. Herein each solution violating any constraints is penalized by infinity. As a consequence infeasible solutions are more likely to be eliminated by evolutionary algorithms during the optimization process.
Another method is to penalize the solutions depending on their constraint violation. Equation 3.2 shows a possible implementation for $p\vec{x}$ with $G_i(\vec{x}) = max\,\{0, g_i(\vec{x})\}$. This may exemplarily mean to set the weights c_j in Equation 3.2 constant during the optimization process. It has proved to be hard to set the weights ideal to achieve a good performance of an optimization process.

$$p(\vec{x}) = \sum_{i=1}^{n_c} c_i G_i(\vec{x}) \tag{3.2}$$

3.1.2. Co-Evolution Penalty Methods

To get rid of the concern adjusting the weights per constraint correctly the usage of *Co-Evolution* becomes handy. By this method two different populations are evolving. One population contains the actual solutions of the constrained population with a penalty function like defined in Equation 3.2. The other population contains the weights (in this example c_i) of the penalty functions. After a certain amount ef evolved generations of the first population the second population evolves its next generations and updates the weights [11].

3.1.3. Dynamic Penalty Functions

Dynamic penalty functions include the iteration or generation number in the penalty function [39]. In this way infeasible solutions are more tolerated at the beginning of the optimization process to achieve a more explorative search and the penalty factor increases later on [22, 8].

3.1.4. Adaptive Penalty Functions

Adaptive penalty functions also include different statistics of the current population into their penalty factor. This may either be the objective value of the worst or best solutions or the mean, minimal or maximal constraint violation. The spectrum of statistical values and their influence in the penalty function is huge. Barbosa, Lemonge, and Bernardino give a detailed overview of some adaptive penalty methods [3].

3.2. Separatist approaches

Another concept of CHTs is the general distinction between the objective and the constraints. By this concept either the objective and constraint violations or feasible and infeasible solutions are separated by a differently.

3.2.1. Multi-Objective Separatist

One option to realize such a *separatist approach* is to separate the constraint violation from the original objective by adding additional objectives.
Mezura-Montes and Coello Coello summarized these CHT in two main groups [37]. The first group of methods transforms the constrained optimization problem into an unconstrained bi-objective problem which aims at minimizing the original objective and the constraint violation (e.g. sum of constraint violation like in Equation 3.2). The other techniques transforms the constrained optimization problem into a multi-objective optimization problem where the original objective and each constraint violation are handled as additional objectives. Thereby they differentiate even further between methods which use Pareto concepts as their selection criteria and methods which use non-Pareto concepts but are mainly based on multiple populations.

3.2.2. Ranking

Another variation is to prefer feasible over infeasible solutions in case of a comparison. The implementation may be done by the tournament selection of an evolutionary strategy or an adaptation of a bubble sort[1].
Many techniques have adopted and/or extended the approach by Deb [14]. This approach applies the following rules for a pairwise comparison:

- If both individuals are feasible, the individual with the better fitness value wins
- If an infeasible individual with a feasible one is compared, the feasible individual wins
- If both individuals are infeasible, the individual with the smaller constraint violation wins

These rules can also be implemented as an *adaptive penalty* function as follows:

$$\varphi(\vec{x}) = \begin{cases} f(\vec{x}) & \text{if } \vec{x} \text{ is feasible} \\ f_{worst} + \sum_{i=1}^{n_g} \max(0, G_i(\vec{x})) & \text{else} \end{cases} \tag{3.3}$$

[1]Bubble sort is a sorting algorithm. It iterates over a list comparing each entries pairwise and adjusting their order according the sort order. The iteration over the entries is repeated until no adjustments are applied

where f_{worst} is the current worst fitness value of the population and G_j are the inequality constraints(e.g. implemented in [12]).
Among other variations of Deb's rules the *stochastic ranking* is one of the more common [53]. Herein a bubble sort is applied. During its pairwise comparison Deb's rules are applied by a probability P_F. In the case that both individuals are feasible the probability is equal to 1 and the comparison is based on their fitness values only; otherwise, the probability is set to P_F if the individual with the better fitness wins or not.

3.3. Feasibility Preserving

Depending on the constraints in case these are not stochastic and can be evaluated a priori there exist methods to prevent the solutions of leaving the feasible domain. This idea can be implemented in different ways.

3.3.1. Repair Methods

Repair methods are approaches which repair infeasible solutions by moving them back into the feasible space. Methods which pull infeasible solutions on the constraint boundary of the feasible space lead a loss of population diversity. Otherwise techniques which places the infeasible solutions randomly inside the feasible space results in a loss of possible useful information gathered by this solution [47].

3.3.2. Special Operator

Slightly different are methods that use *special operators*. These need a completely feasible start population. Their main idea is to adjust the steps of the optimization process which bring up new solutions in a way that they stay inside of the feasible domain.
The *Simplex crossover* is an example of this type of CHTs. Hereby *number of variables* $+1$ parents are chosen to span a subspace by interpolating between the parents randomly and thus creating a new feasible child [56].
A necessary condition for such an operator is a single convex feasible domain.

3.3.3. Decoder

The basic idea of a *decoder*-method is to find an encoding for the genotype of solutions/individuals which represents the entire feasible domain. In such a way the decoded phenotypes are unable to leave the feasible domain. Thereby the optimization process searches on the genotype of the solutions and the evaluation is calculated on their phenotype. The usability of this approach depends on the problem constraints. As a consequence not every decoder is transferable to other problems.
One possible decoder *homomorphous mapping* was introduced by Koziel and Michalewicz in the year 1999 [32]. Herein the feasible domain is mapped on a hypercube with the range $[-1, 1]^{n_x}$.
Other techniques are based on *riemann-mapping* [30] or even on *support vector machine* [7].

3.4. Hybrids & Others

Combinations of different CHTs, also referred to as *hybrids*, are also investigated. An example of such a combinational approach is the ensemble by Mallipeddi, Das, and Suganthan [36] in which the offspring of four separated populations each evolving with another CHT are evaluated and merged into each population based on its CHT.
Another method is described in [9]. The authors use a modification on Deb's rules and a special operator for a crossover between two feasible solutions which produces only feasible solutions.

3.5. Overview

Finally for this chapter an overview of some CHT is given in the table 3.2. The table contains the name of the authors of an implemented CHT, the maximal number of variables and constraints of the benchmark problems the CHT is evaluated on.
In addition to the high amount of papers considering Constraint Handling Technique (CHT) also a couple of surveys are published. Jordehi have different CHTs adapted to PSO [25]. Mezura-Montes and Coello compared multi-objective based CHTs [37] and collected CHTs for nature-inspired optimization methods [38]. Barbosa, Lemonge, and Bernardino studied different adaptive penalty methods [3].

As shown in table 3.2 the highest dimensionality of the benchmark problem is 24 and the highest amount of constraints is 38. There have not been further research with higher dimensional problems with even more constraints. This underlines the gap of investigations with CHT in higher dimensions. This thesis tries to initiate further research by applying a reasonable amount of diverse CHTs on constrained optimization problems of higher dimensions.

Reference	n_x	n_c	LI	NI	LE	NE
penalty functions						
Parsopoulos and Vrahatis[48]	4	7	3	5	-	-
Homaifar, Qi, and Lai[21]	5	6	1	6	-	-
Mezura-Montes and Flores-Mendoza[39]	24	38	9	34	8	12
Joines and Houck[23]	7	6	2	4	-	3
Nanakorn and Meesomklin[44]	10	12	12	-	-	-
Farmani and Wright[17]	13	9	9	6	-	3
Qiao[51]	13	9	9	6	-	-
Montemurro, Vincenti, and Vannucci[42]	24	7	3	7	-	1
Wang, Cai, Zhou, and Fan[57]	20	27	9	27	-	3
Tessema and Yen[55]	20	38	9	34	3	5
Lin and Wu[35]	10	12	12	3	-	-
Coello[11]	5	7	5	3	-	-
separatists						
Krohling and dos Santos Coelho[34]	12	9	9	5	-	-
Mezura-Montes and Coello Coello[37]	10	8	6	6	-	1
Deb[14]	13	38	9	38	-	3
Poole, Allen, and Rendall[50]	24	38	9	34	8	12
Costa, Santo, and Oliveira[12]	24	38	38		19	
Karaboga and Basturk[27]	12	9	9	6	-	5
Runarsson and Yao[53]	13	9	9	6	-	3
feasibility preserving						
Kim[30]	2	3	2	2	-	-
hybirds & others						
Chehouri, Younes, Perron, and Ilinca[10]	5	7	3	6	-	-
Muñoz Zavala, Aguirre, and Villa Diharce[43]	12	9	9	6	-	5
reviews and surveys						
Michalewicz and Schoenauer[40]	13	9	9	6	-	3

Table 3.2.: Dimensionality of benchmark problems in papers CHT: n_x: number of parameters, n_c: amount of constraints, LI: linear inequalities constraints, NI: non-linear inequalities constraints, LE: linear equality constraints, NE: non-linear equality constraints. Each number is the maximal amount of a benchmark problem evaluated in the reference. Because the references may contain multiple benchmark problems the mentioned numbers do not automatically pertain to the same benchmark problem.

4. Considered Class of Problems

This chapter contains the definition of the considered class of problems in this thesis. Beside the mathematical definition, some resulting properties are mentioned. Additionally the processing of a given problem is described. How it is handled and reformulated correspondingly to its explicit declaration. Also some alternative reformulations methods not implemented are mentioned at the end of the second section.
In the last section of this chapter two explicit example problems out of the considered class of problems are described. These two problems are regarded as representatives and are used in the following for evaluation purposes of different test-cases which are described in the following chapters.

4.1. Class of Problems

The equation 4.1 shows the class of problems dealt with in this thesis. The given fitness function f has to be minimized and is treated as a blackbox. Therefore no additional information like derivatives and further information such as continuity and number of local optima is available. Furthermore the evaluation of a solution by f is computationally intensive.

$$\begin{aligned} \underset{\vec{x}}{min} \quad & \qquad f(\vec{x}) \\ s.t. \quad \vec{x}_{min} \leq & \quad \vec{x} \leq \vec{x}_{max} \\ \vec{c}_{min} \leq & A\,\vec{x} \leq \vec{c}_{max} \end{aligned} \tag{4.1}$$

The *Search Space* ($\mathcal{S}$) of this optimization problem is is continuous, $\vec{x} \in \mathbb{R}^{n_x}$, and only limited by box-constraints, $\vec{x}_{min}, \vec{x}_{max} \in \mathbb{R}^{n_x}$. Where n_x is the number of considered dimensions. In the following the box-constraints to the search space will be referred as boundaries.
The *Feasible Domain* (F) as a subspace of $\mathcal{S}$ is restricted by linear constraints

$\vec{c}_{min} \leq A\vec{x} \leq \vec{c}_{max}$ with $A \in \mathbb{R}^{n_c \times n_x}$, $n_c > n_x$. The matrix A containing the coefficients for the linear constraints is sparse. Since the constraints consist only of linear inequalities, F as a subspace of the search space satisfying all linear constraints has to be a convex polytope. Figure 4.1 illustrates the convex polygon F inside $\mathcal{S}$ for a simplified two-dimensional example.

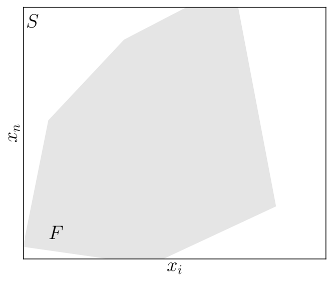


Figure 4.1.: Simplified two-dimensional $\mathcal{S}$ (box-constrained) and F (as a polygon inside of $\mathcal{S}$)

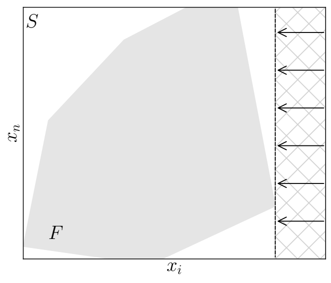


Figure 4.2.: Reduced $\mathcal{S}$ with the adjusted boundary for $x_{i,max}$

4.2. Problem Reformulation

Since a high amount of variables and constraints is considered by the problem these will be analyzed and reformulated to get a uniform formulation or even reduce some complexity in the best case.

4.2.1. Constraints Transformation

First the formulation of the constraints will be translated in a more common way used in the literature. Equation 4.2 shows the reformulation of the constraints.

$$\begin{bmatrix} A \\ -A \end{bmatrix} \vec{x} + \begin{bmatrix} -\vec{c}_{max} \\ \vec{c}_{min} \end{bmatrix} \leq 0 \tag{4.2}$$

In the following the constraints definition will be substituted and referenced as described in Equation 4.3.

$$\mathcal{A}\vec{x} + \vec{b} \leq 0 \tag{4.3}$$

Single inequalities are referenced as $g_i(\vec{x})$ defined in Equation 4.4 where $\mathcal{A}_i$ is the i-th row of $\mathcal{A}$ and b_i the i-th entry of $\vec{b}$.

$$g_i(\vec{x}) \;=\; \mathcal{A}_i\vec{x} + \vec{b}_i \;\leq\; 0 \tag{4.4}$$

Furthermore, all restrictions combined (Equation 4.5) are referenced in the following of the thesis as described in Equation 4.6.

$$\begin{bmatrix} \mathcal{A} \\ I \\ -I \end{bmatrix} \vec{x} + \begin{bmatrix} \vec{b} \\ -\vec{x}_{max} \\ \vec{x}_{min} \end{bmatrix} \;\leq\; 0 \tag{4.5}$$

$$\mathcal{A}^{\dagger}\vec{x} + \vec{b}^{\dagger} \;\leq\; 0 \tag{4.6}$$

4.2.2. Removing fixed Variables

In some cases the boundaries for a variable x_i can be equal, $\vec{x}_{max,i} = \vec{x}_{min,i}$. These variables x_i are fixed and therefore it is ignored by the optimization method.

4.2.3. Removing Infinity Inequalities

Beside fixed variables, also some constraints are always satisfied. Therefore the evaluation if these constraints are satisfied or not can be ignored saving unnecessary computational time. Always satisfied constraints occur if $\vec{b}$ contains negative infinity entries. These entries can be simply deleted with their related row in $\mathcal{A}$. Thus the number of constraints is reduced and computing effort is saved.

4.2.4. Adjust Boundaries

Since the feasible domain is restricted by constraints it may appear that F does not even touch a boundary of the search space. Therefore the boundaries restricting $\mathcal{S}$ may be adjusted to minimize $\mathcal{S}$ down to a reasonable volume. In such a case the optimization method may explore an unnecessary huge infeasible space. To counter this scenario each border will be checked, if it is reachable from the feasible domain or not. Linear Programming is used for this purpose. The algorithm to adjust the

boundaries is described in Algorithm 2. "solveLP" in line 4 and 8 solves the linear optimization problem as described in Equation 4.7 with the passed $\vec{c}$. Since $\vec{c}$ corresponds to a *unit-vector* in the direction of an axis the optimum of the LP is the maximal or minimal feasible solution in this direction depending on the sign of $\vec{c}$. The *solver* used for the LP is from the CVXOPT-package [6]. The default LP-Solver in CVXOPT uses the *interior-point* method. Since the solver moves in the direction of the gradient $\vec{c}$ and continuously nears the borders, it does not necessarily have to land exactly on the border and can abort its optimization process just before. Therefore a tolerance is defined to check if the boundary is reached or not. The tolerance *tol* is set to $1\mathrm{e}{-7}$.
Figure 4.2 illustrates the adjustment of a boundary for a $x_{max,i}$ within a two-dimensional example. As a consequence of adjusting the box-constraints alias boundaries F touches at least with a corner point a facet of the hypercube defined as $\mathcal{S}$.

Algorithm 2: Adjusting boundaries of the search space

```
Input: $\vec{x}_{min}$ and $\vec{x}_{max}$ defining the boundaries of the search space
Output: $\vec{x}_{min}$ and $\vec{x}_{max}$ adjusted boundaries
1  for $i \leftarrow 1$ to $n_x$ do
2      $\vec{c} \leftarrow$ zero-vector($n_x$)
3      $\vec{c}[i] \leftarrow 1$
4      $\vec{x} \leftarrow$ solveLP($\vec{c}$)
5      if $\vec{x}[i] - \vec{x}_{min}[i] > tol$ then
6          $\vec{x}_{min}[i] \leftarrow \vec{x}[i]$
7      $\vec{x} \leftarrow$ solveLP($-\vec{c}$)
8      if $\vec{x}_{max}[i] - \vec{x}[i] > tol$ then
9          $\vec{x}_{max}[i] \leftarrow \vec{x}[i]$
10 return $\vec{x}_{min}, \vec{x}_{max}$
```

$$
\begin{aligned}
\min_{\vec{x}} \quad & \vec{c}^{\,T}\vec{x} \\
& \mathcal{A}^{\dagger}\vec{x} + \vec{b}^{\dagger} \leq \vec{0}
\end{aligned}
\tag{4.7}
$$

4.2.5. Normalization of the constraints

For some CHT it comes handy to know the degree of violation of a constraint. To calculate the violations of the constraints evenly, these are normalized. For this purpose the normals of each hyperplane defined by the raws in $\mathcal{A}$ are normalized to a unit length of 1. Accordingly $\vec{b}$ is also normalized.
The benefit of normalizing the normals of the hyperplanes is that the violation of a constraints matches the minimal euclidean distance from the infeasible solution to the hyperplane. As an example to contrary the inequalities could be multiplied by a scalar $z(\mathcal{A}^\dagger\vec{x} + \vec{b}^\dagger) \leq \vec{0}$ which in case of violating constraints would increase the degree of violation by the factor z.

4.2.6. Rejected reformulations

Since the feasible domain has the geometrical structure of a convex polytope the search-space could be examined by an interpolation of at least $n_x + 1$ chosen corner points of the polytope. This can be realized only by a lower amount of variables to the time of preparation of this theses since the memory-usage to calculate the corner points is immense. For instance the amount of corner points of a hypercube can be computed as 2^{n_x}, whereby a problem with 50 dimensions already needs 16×10^6Gb at an accuracy of 64-bit float per value saved. Such a hypercube consists of $2 \times n_x$ faces. Hypothetical randomly chosen hyperplanes defining a polytope like defined by $\mathcal{A}$ and $\vec{b}$ can even consists more corner points than a hypercube within the same dimensions.

4.3. Considered Problems

For a more detailed investigation, two representative problems of the previously described problem class are made up artificially, P_1 and P_2. Table 4.1 shows the amount of variables and constraints per problem before and after the problem reformulation. It also shows the percentage of $\mathcal{S}$ after the adjustment of the boundaries. The problem P_1 has no fixed variables and $\mathcal{S}$ can not be reduced by the adjustment of the boundaries. Different with the second problem P_2 which has one fixed variable and $\mathcal{S}$ can be reduced down to approximately 20% of the original volume.

Equation 4.8 shows the calculation of the percentage where $\vec{\mathbf{x}}_{max}$ and $\vec{\mathbf{x}}_{min}$ are the original and $\vec{x}_{max}$ and $\vec{x}_{min}$ are the adjusted boundaries.

$$\prod_{i=1}^{n_x} \frac{\vec{x}_{max,i} - \vec{x}_{min,i}}{\vec{\mathbf{x}}_{max,i} - \vec{\mathbf{x}}_{min,i}} \tag{4.8}$$

Since these representative problems are made up artificially the global optimum for each one is known. In both cases the fitness of each optimum is 0. The Figures 4.3 and 4.4 show plots of the respectively optima. The chosen representations are parallel-coordinates. In parallel-coordinates the axis for each dimension are placed in parallel and therefore the x-axis of the plots represents the index of the corresponding dimension. The top plot represents along the x-axis the index per variable x_i inside $\mathcal{S}$ and the bottom plot has the index of per constraint $g_i(x)$ along the x-axis. The y-axis represents the value per variable or satisfaction/violation per constraint respectively. Due to the parallel placed axis a point appears as a line in parallel-coordinates.
The first plot in each case shows the optimum inside $\mathcal{S}$. The values corresponding to the y-axis are normalized from $[\vec{x}_{min}, \vec{x}_{max}]$ to $[0, 1]$ to achieve a more convenient representation filling up the whole plot. The second plot shows the optimums constraint violation as well in parallel-coordinates.
As can be seen from the plots the optimum of P_1 is rather at the edge of *Search Space* $(\mathcal{S})$ and *Feasible Domain* (F) (Figure 4.3), whereas for the second problem P_2 it is rather a little bit far away from the borders(Figure 4.4). This may play a role by the optimization process in case whole $\mathcal{S}$, the area close to the borders or the inner region of F is preferred to be explored by an approaches or not.

This kind of plots is also used in the following of this thesis as representation of solutions. As an orientation the optimum is always plotted on top as a blue dotted line. The colouring of solutions outside the optimum corresponds to the two shown colourbars, *Feasibility* for feasible solutions and *Infeasibility* for infeasible solutions. The degree of *Feasibility* is calculated as the squared root of the sum of constraint satisfaction. The degree of *Infeasibility* is calculated as the squared root of the sum of constraint violation.

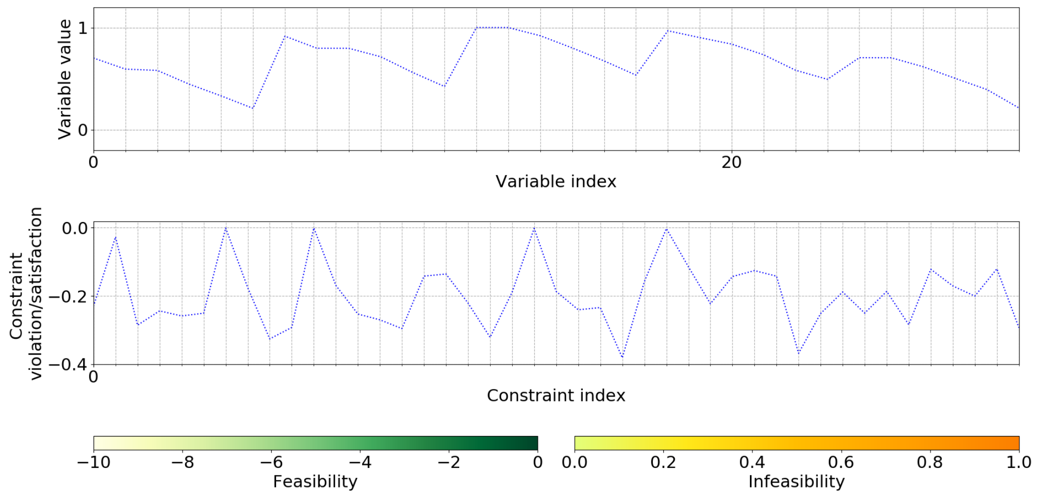


Figure 4.3.: Global optimum of problem P_1

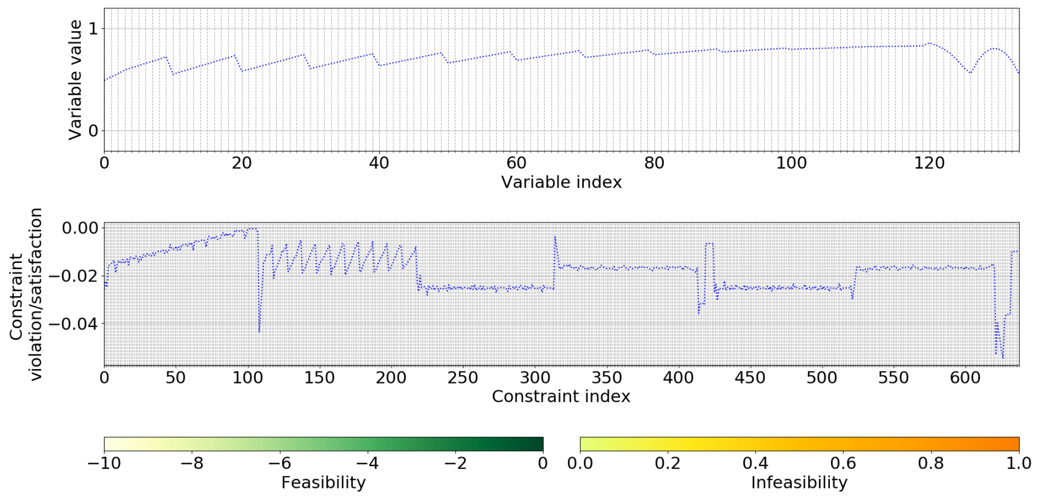


Figure 4.4.: Global optimum of problem P_2

Problem	original		reformulated		$\mathcal{S}$
	#-vars	#-cons	#-vars	#-cons	
P_1	30	43 * 2 = 86	30	43	100%
P_2	135	431 * 2 = 862	134	631	ca. 19.58%

Table 4.1.: This table contains the number of variables (*#-vars*) and constraints without the box-constraints (*#-cons*) per test-problem before (*original*) and after the reformation (*reformulated*). The last column represents the percentage of the *Search Space* ($\mathcal{S}$) after the adjustment of the boundaries.

5. Approaches

In this chapter several approaches to solve the problem described in the previous chapter are presented. An approach describes an optimization process which is a combination of three elements, an *Initialization* (INIT), a *Constraint Handling Technique* (CHT) and a *Optimization Method* (OM).Algorithm 3 shows the general structure of an approach containing these three parts terminating by a limit of fitness evaluations. This type of terminating the optimization process is chosen because the fitness evaluations of the defined problems are computationally expensive and their usage is limited in respective to computing time.
Each part of the optimization process is realized in different ways. The following sections describe several variations of each of these parts which are implemented and evaluated in this thesis. Before that details of the implementation are explained.

Algorithm 3: General structure of an optimization process

```
Input: $In$ - Initialization; $CHT$ - Constraint Handling Technique; $OM$ -
       Optimization Method
1 pop ← init via $In$
2 while limit of fitness evaluations reached do
3     pop ←evolve next population via $OM$ and $CHT$
```

The programming part for this thesis is realized in *python*. The packages *scipy* [24] and *numpy* [45] provide necessary datastructures for processing and storing data. To prevent a reimplementation of the chosen population-based optimization algorithms an existing framework containing them is elected. The scientific library *pagmo* [5] offers a wide range of population-based optimization algorithms. *Pagmo* is written in C++ and is built up to provide massively parallel optimization. Due to the high dimensionality of the considered problems, the efficency through the parallelization is a benefit for the calculations during the testcases. The corresponging python version of this free/libre opensource software is called *pygmo*. This library is mainly build up with wrappers arround the underlying C++ library.

5.1. Start Population

The first step of each approach is the initialization of the first population from which population-based meta-heuristic begin. The performance of the meta-heuristics can vary depending on the used start population. A main criteria is a good diversity at the beginning of the optimization process. Four out of a wide range of different initialization techniques are chosen as representatives and are described in the following. The first initialization is a naive method by sampling randomly inside $\mathcal{S}$. The other three produce feasible solutions only. The second and third create feasible solutions randomly whereby the fourth method generates the same initial solutions since this method is deterministic. Conclusively a paragraph gives a short overview of the initialization methods on two-dimensional examples.

5.1.1. Random Initialization

A naive initialization technique is to chose the individuals of the start-population randomly. Since the chosen framework *pygmo* already provides such an initialization it is adapted and not modified further. Figures 5.1 and 5.2 show one thousand randomly initialized individuals for each problem respectively. The representation of the solutions by their constraints violation/satisfaction emerges peaks. This effect appears by P_1 more intense then by P_2 since P_1 has less constraints and therefore the distance between each constraint violation is larger in the visualized plot. This peaks emerge due to the definition of the linear constraints. Because the linear equation $A\,\vec{x}$ have to yield values between $\vec{c}_{min}$ and $\vec{c}_{max}$ the solutions automatically satisfy one border by violating the other.
It is to mention that all random initialized solutions are infeasible. Even one million randomly initialized solutions does not seed a single feasible solutions. This leads to a small F in ratio to $\mathcal{S}$ since it is improbable to initiate a feasible solution randomly. This INIT is referred to as *Ramdom Initialization* (RaI) in the following of this thesis.

5.1.2. Repairing Initialization

Since initializing feasible solutions completely random is improbable it may improve the optimization process by starting with a feasible population. Therefore three different techniques are chosen for this purpose. The first presented technique

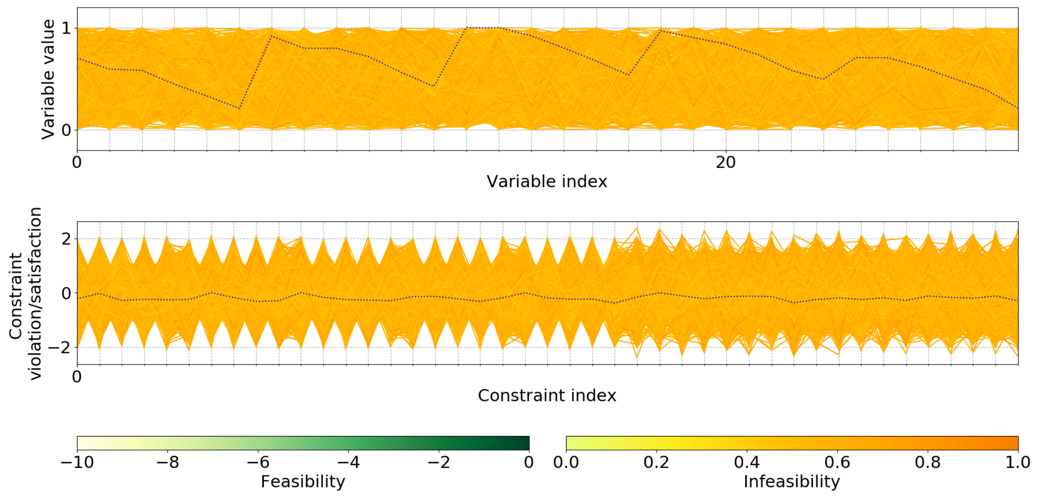


Figure 5.1.: One thousand randomly initialized solutions for P_1

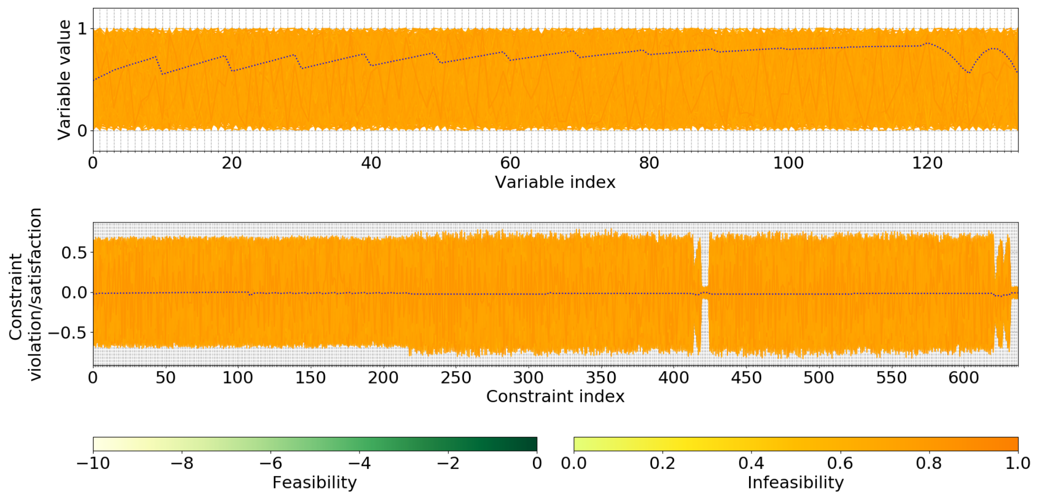


Figure 5.2.: One thousand randomly initialized solutions for P_2

initializing feasible solutions only is *Repairing Initialization*. This initialization method consists of two steps. First, random solutions like described previously are generated. Second, the infeasible solutions $\vec{x}$ will be repaired by moving them to their nearest feasible position $\vec{y}$. The point $\vec{y}$ fulfills all constraints and has the minimal euclidean distance to $\vec{x}$. This criteria of minimal distance can be defined as a quadratic optimization problem with subject to the same constraints as the origin problem. Eq. 5.1 shows the resulting problem definition. The fitness function minimizes the distance $||\vec{y} - \vec{x}||_2$. The calculation of the distance can be rewritten as $(\vec{y} - \vec{x})(\vec{y} - \vec{x})^T$ without taking the root. Omitting the root may influence the distance but not the resulting nearest feasible point $\vec{y}$. Afterwards this formulation can be written out as $\vec{y}^T\vec{y} - 2\,\vec{x}^T\vec{y} + \vec{x}^T\vec{x}$. The last summand $\vec{x}^T\vec{x}$ can be ignored for the quadratic optimization problem because it is constant and therefore does not influence the resulting optimal $\vec{y}$.

$$\begin{aligned} \underset{\vec{y}}{min} \qquad & \vec{y}^T\vec{y} - 2\,\vec{x}^T\vec{y} \\ & \mathcal{A}^{\dagger}\vec{y} + \vec{b}^{\dagger} \leq 0 \end{aligned} \tag{5.1}$$

Herein, $\vec{x}$ is the infeasible solution which needs to be repaired. $\vec{y}$ is the resulting feasible solution closest to $\vec{x}$. The linear constraints are the same like the constraints of the corresponding problem. To find a reasonable solution of this quadratic optimization problem (Equation 5.1) the default QP-Solver from *CVXOPT* is used.

For each problem one thousand solutions randomly initialized and repaired afterwards are plotted in Figures 5.3 and 5.4. As it can be seen the repaired solutions seem to concentrate more to the center of S than to the boundaries. At the problem P_2 this characteristic is much more significant.
Such an initialization seems to produce a start-population closer to the global optimum inside F. At P_1 the global optimum is nearly surrounded completely by a feasible start-population. Where at P_2 the solutions are rather distanced from the global optimum. This INIT is referred to as *Repaired Initialization* (ReI) in the following of this thesis.

5.1.3. Initialization via Gibb's sampling

The third initialization method used in this thesis is based on the *Gibbs-Sampling Algorithm* firstly introduced by Geman and Geman in the year 1987 [18]. The

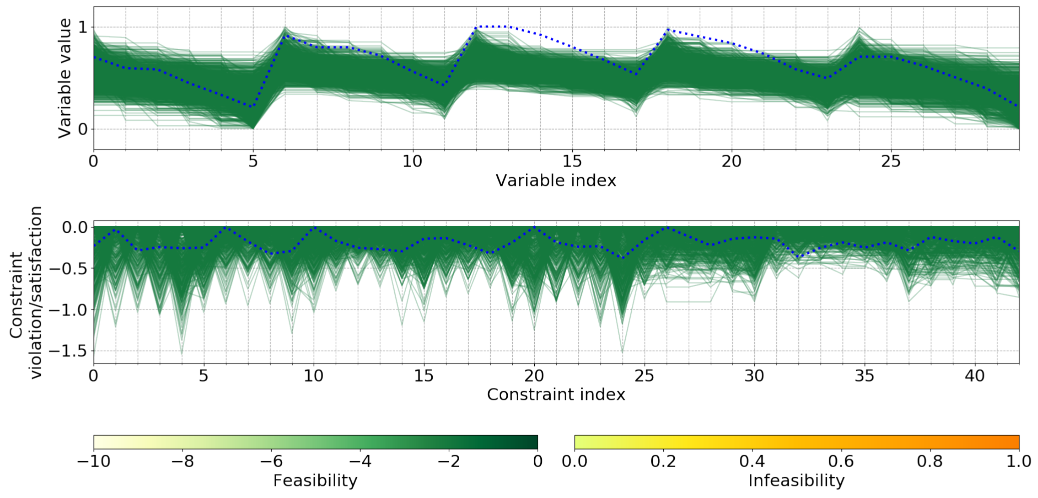


Figure 5.3.: One thousand randomly initialized solutions for P_1 and repaired afterwards.

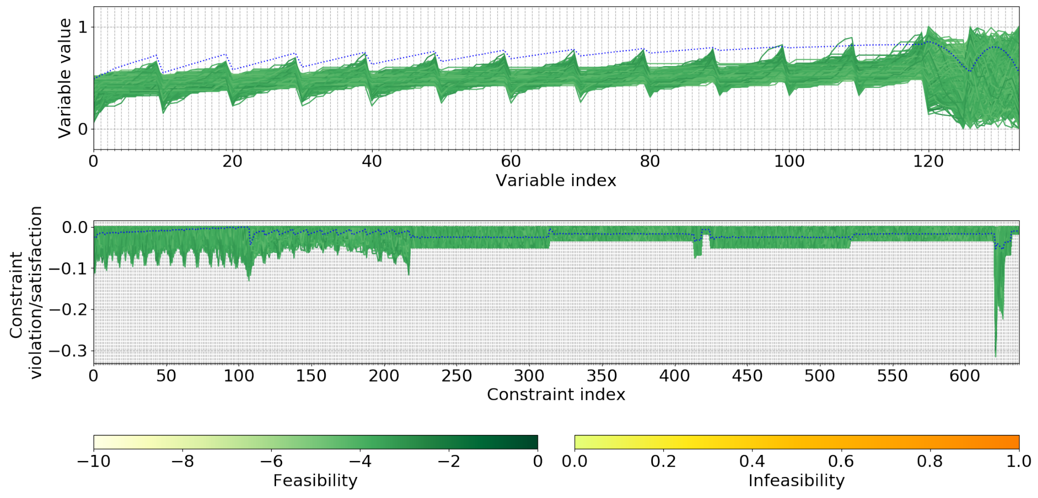


Figure 5.4.: One thousand randomly initialized solutions for P_2 and repaired afterwards.

implemented method is described as pseudo code in Algorithm 4. First a feasible solution as a starting point is needed. For this a randomly chosen solution is repaired (lines $3, 4$). This sample is then modified n_x times for each variable separately. Thereby the range how the sample can be modified according to one variable is calculated via the Algorithm 5. This calculates the furthest feasible point staring from a feasible point $\vec{s}$ into a direction $\vec{d}$. After the maximal feasible points are calculated along an axis in positive and negative direction the sample is updated by interpolating between these maxima randomly (lines $6 - 8$). When all variables in the sample are modified it is added to the start-population and the procedure is repeated until the wanted amount of feasible solutions is reached. In Figure 5.5 this initialization method is illustrated on an simplified two-dimensional example.
Figure 5.6 shows an initialization via Gibb's Sampling with 60 samples for P_1 and Figure 5.7 with 268 samples for P_2. The solutions of the resulting start-populations do not spread widely but rather close to each other. Therefore, the start-popularions appear as bands in the corresponding figures. This INIT is referred to as *Gibb's Sampling* (GS) in the following of this thesis.

Algorithm 4: Gibbs Sampling Algorithm

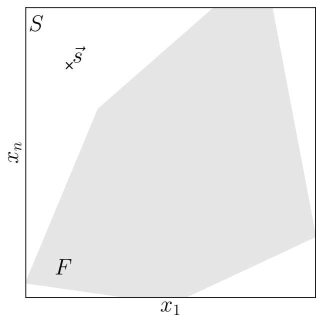
```
Input: $n_v$ amount of solutions to be initialized
Output: $pop$ feasible starting population
1  $pop \leftarrow$ empty list
2  $\vec{s} \leftarrow$random solution
3  $\vec{s} \leftarrow$repair($\vec{s}$)
4  while $pop.size < n_v$ do
5      foreach column $\vec{c}$ in $I$ do
6          $\vec{s}_{max} \leftarrow$get_furthest_feasible_point($\vec{s}, \vec{c}$)
7          $\vec{s}_{min} \leftarrow$get_furthest_feasible_point($\vec{s}, -\vec{c}$)
8          $r \leftarrow$ random$(0, 1)$
9          $\vec{s} \leftarrow \vec{s}_{min} + r(\vec{s}_{max} - \vec{s}_{min})$
10     $pop.append(sample)$
11 return $pop$
```

Algorithm 5: Get furthest feasible point

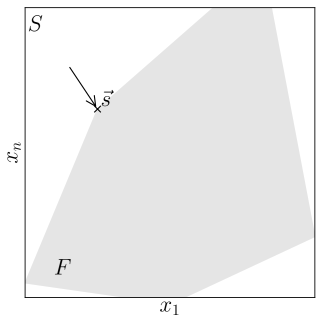
```
Input: $\vec{s}$ as a feasible reference point and $\vec{d}$ as the direction to look for
Output: $\vec{p}$ as the farest feasible point starting from $\vec{s}$ in the direction $\vec{d}$
1 $\vec{z} \leftarrow$ element-wise division $-\frac{\mathcal{A}^{\dagger}\vec{s} + \vec{b}^{\dagger}}{\mathcal{A}^{\dagger}\vec{d}}$
2 remove all values less or equal to zero in $\vec{z}$
3 $m \leftarrow$MinimumOf($\vec{z}$)
4 $\vec{p} \leftarrow \vec{s} + m\,\vec{d}$
5 return $\vec{p}$
```

5.1.4. Initialization on Boundaries

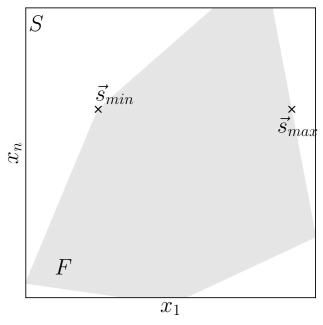
As a counterpart to the *repaired solution*-initialization method this initialization method spawns solutions on the boundaries instead on the linear constraints. This method works similar to the approach *adjusting the boundaries* (see Agorithm 2, 22). However, instead of adjusting the boundary using the maximal feasible point ($\vec{x}$ in line 4 of Algorithm 2) in one direction by a comparison, it is saved into the population (Agorithm 6). The method s̈olveLP" in the lines 4 and 5 solves the linear optimization problem as described in Equation 4.7 with the passed $\vec{c}$ like in Algorithm 2. The LP-Solver is as well the *interior-point* method provided by CVXOPT. The resulting populations for P_1 and P_2 are shown in the Figures 5.8 to 5.11. Figures 5.8 and 5.10 show solutions at the lower boundaries for P_1 and P_2 and Figures 5.9 and 5.11 at the upper boundaries respectively.
Since the LP-Solver calculates the optimum deterministic the solutions of the start-population initialized by this method are always the same. It is also to mention that the amount of initialized solutions by this methods is always two times n_x. For a varying amount of initialized solutions further research in modifying this method is necessary. This INIT is referred to as *on boundaries* (OB) in the following of this thesis.



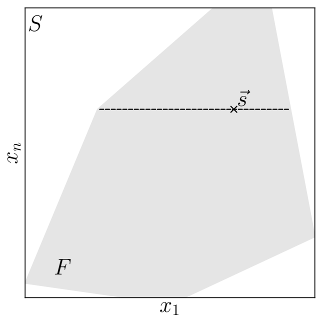
(a) randomly seeded sample $\vec{s}$



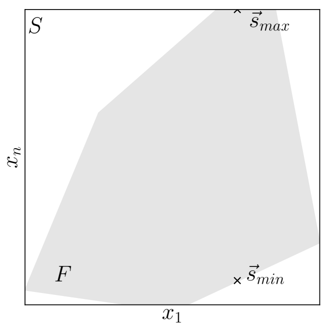
(b) repairing $\vec{s}$



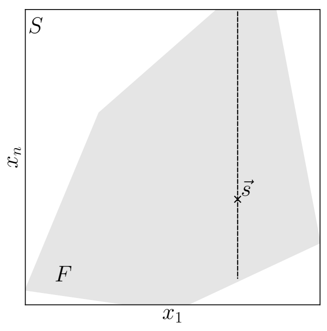
(c) get $\vec{s}_{min}$ and $\vec{s}_{max}$ along $\vec{x}_1$-axis



(d) interpolate between $\vec{s}_{min}$ and $\vec{s}_{max}$



(e) iget $\vec{s}_{min}$ and $\vec{s}_{max}$ along $\vec{x}_n$-axis



(f) interpolate between $\vec{s}_{min}$ and $\vec{s}_{max}$

Figure 5.5.: Gibb's Sampling for a two-dimensional example.

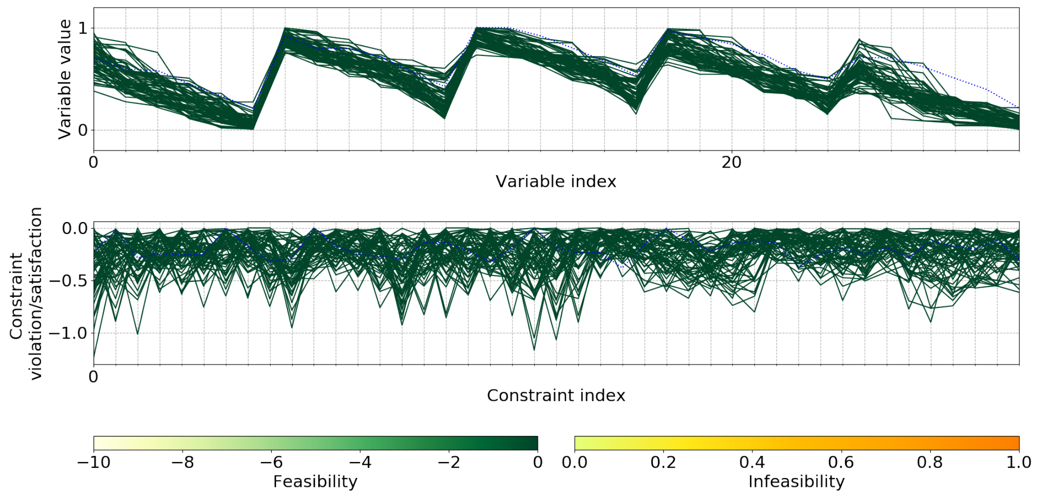


Figure 5.6.: Initialization via *Gibbs Sampling* with 60 samples for P_1

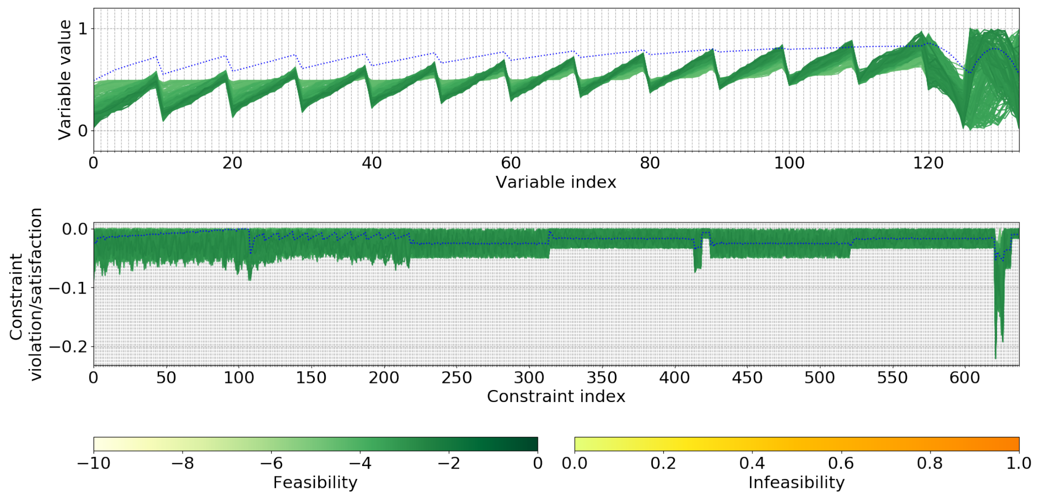


Figure 5.7.: Initialization via *Gibbs Sampling* with 268 samples for P_2

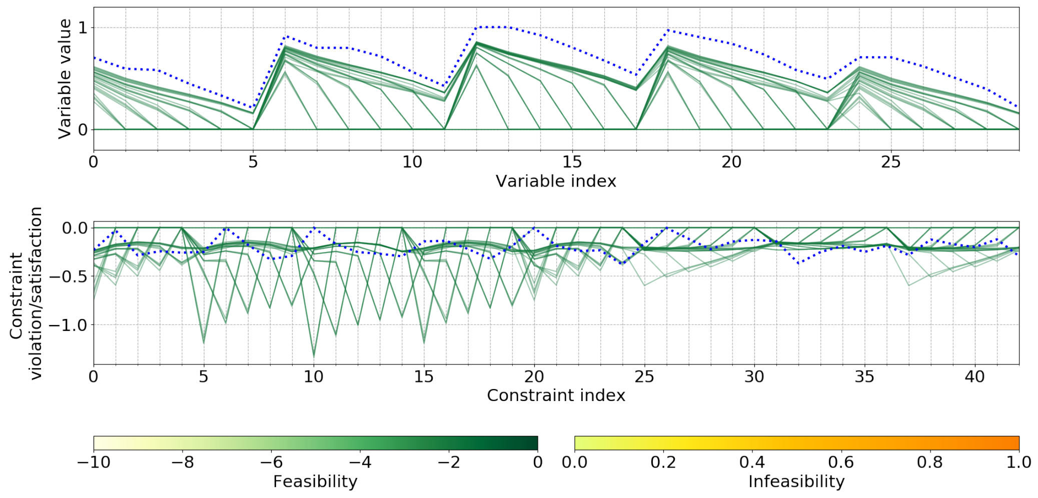


Figure 5.8.: Initialization on minimal boundaries of P_1.

Algorithm 6: Spawn feasible solutions on boundaries

```
Output: $pop$ feasible starting population on the boundaries
1 for $i \leftarrow 1$ to $n_x$ do
2     $\vec{c} \leftarrow$ zero-vector($n_x$)
3     $\vec{c}[i] \leftarrow 1$
4     $pop.append$(solveLP($\vec{c}$))
5     $pop.append$(solveLP($-\vec{c}$))
6 return $pop$
```

Overview Initialization

The first initialization shows the difficulty entering F randomly. This allows the suggestion that F seems to be small in ratio to $\mathcal{S}$. Since the boundaries of $\mathcal{S}$ are adjusted to be minimized and enclose F (see Adjust Boundaries 4.2.4) each boundary of $\mathcal{S}$ is touched by F by at least a corner point. This leads to the conclusion that the shape of the convex polygon F is to narrow and diagonally aligned.

Figure 5.12 illustrates two-dimensional examples of the presented initialization methods. Thereby the same effects like in the problems are observable. The randomly initialized solutions are infeasible, the repaired solutions concentrate to the center per axis and the Gibb's sampling results in adjacent solutions.

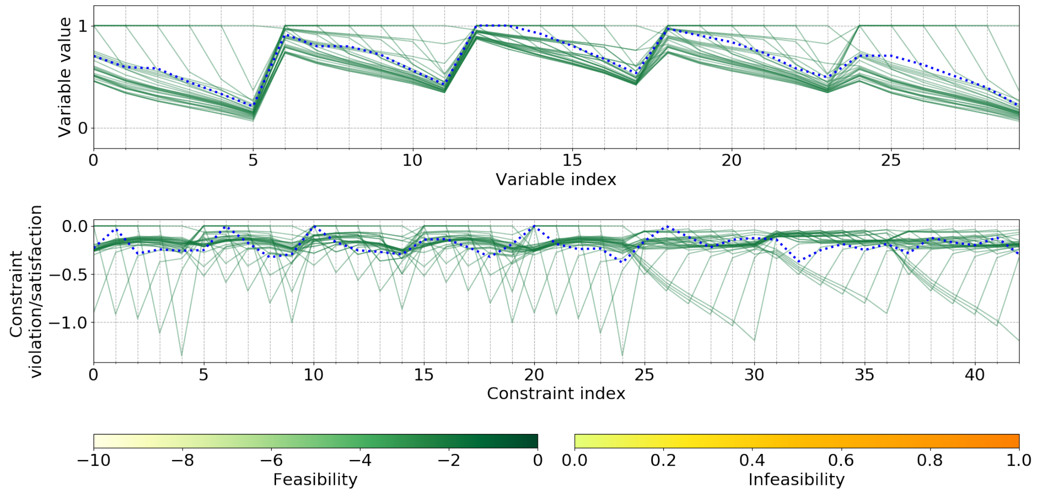


Figure 5.9.: Initialization on maximal boundaries of P_1.

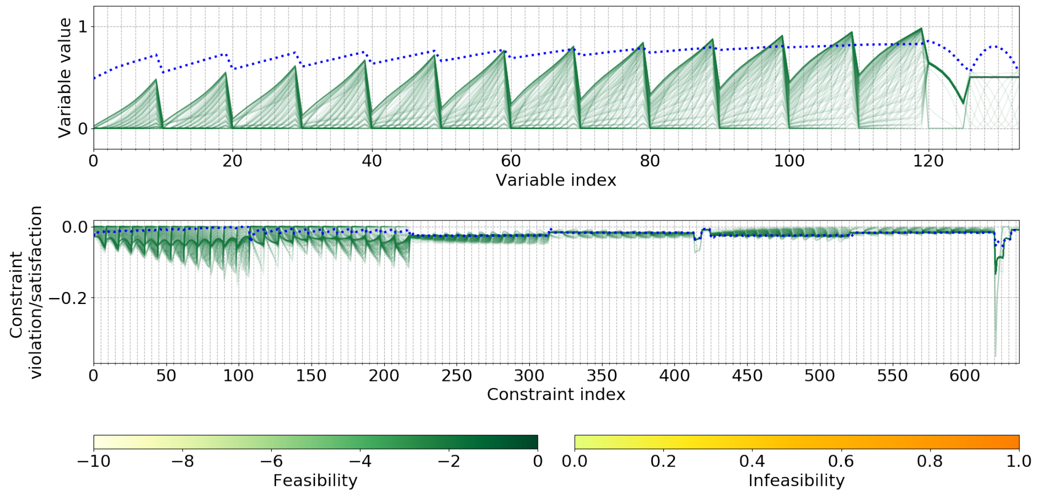


Figure 5.10.: Initialization on minimal boundaries of P_2.

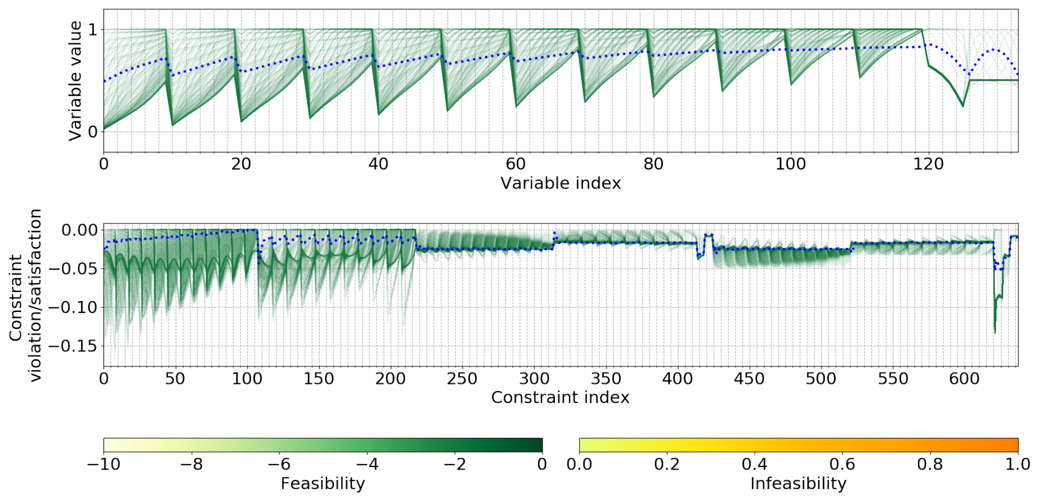


Figure 5.11.: Initialization on maximal boundaries of P_2.

5.2. Constraint-Handling Technique

In this section some CHTs for each category of CHT presented in Chapter 3 are explained in detail. They are also implemented and evaluated for P_1 and P_2 in the following chapter. It is also explained how they interact with *Optimization Method* (OM) described in last section of this chapter.

5.2.1. Used Penalty Approaches

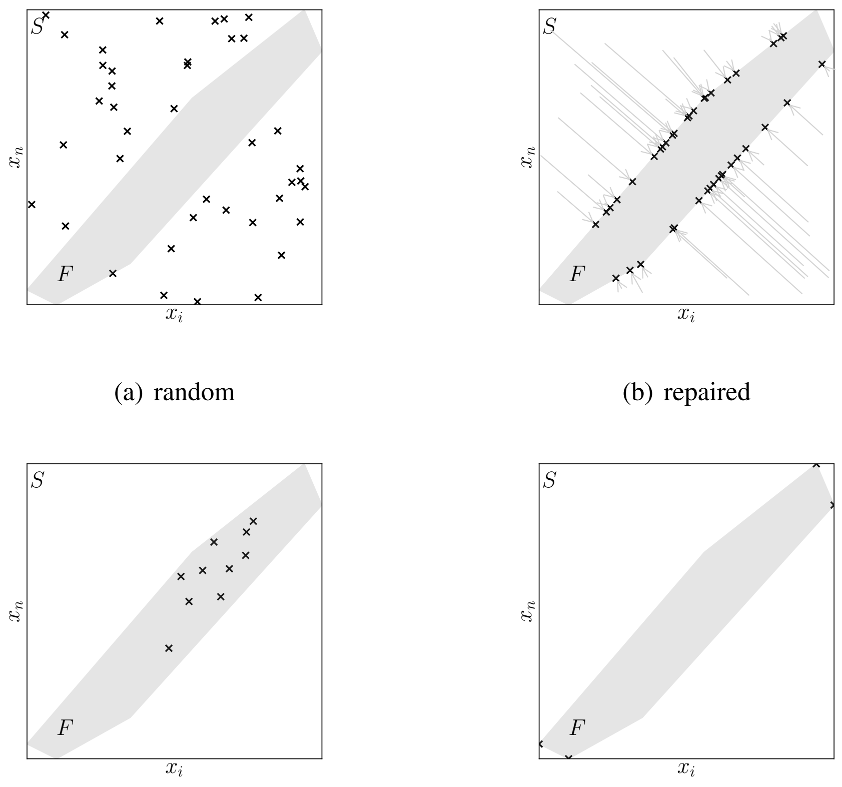
Two static and one adaptive penalty methods are chosen as representatives of penalty methods.

5.2.1.1. Static Penalty

Both static penalty methods extend the objective function f by p (Equation 5.2). Hereby the OMs have to minimize φ.

$$\varphi(\vec{x}) = f(\vec{x}) + p(\vec{x}) \tag{5.2}$$

One static *static* penalty method adds the amount of violated constraints. The calculation of the penalty is shown in Equation 5.3. The function $g_i(\vec{x})$ calculates the



(a) random

(b) repaired

(c) gibb's sampling

(d) on boundaries

Figure 5.12.: Overview of the described initialization methods with a simplified two-dimensional examples

violation/satisfaction of the i-th constraint (see Equation 4.4). Since the penalty factor is the sum of violated constraints the resulting landscape emerges with plateaus containing whole numbers as penalty (see Figure 5.13). In the following this CHT is referred to as avc$_p$.

$$p(\vec{x}) = \sum_{i=1}^{n_c} \begin{cases} 1 & g_i(x) > 0 \\ 0 & else \end{cases} \tag{5.3}$$

The other *static* penalty methods adds the norm of constraint violations defined in Equation 5.4. The landscape of this penalty is continuous inside the infeasible space (see Figure 5.14). It may be interpreted as the shortest distance to F mistakenly. This CHT will be referred to as ncv$_p$ in the following of this thesis.

$$p(\vec{x}) = \sqrt{\sum_{i=1}^{n_c} \begin{cases} g_i(x)^2 & g_i(x) > 0 \\ 0 & else \end{cases}} \tag{5.4}$$

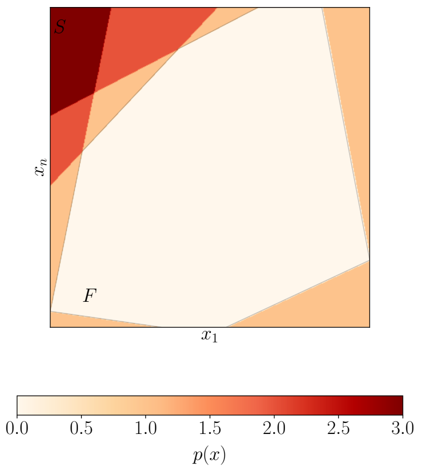


Figure 5.13.: Landscape of $p(x)$ with the amount of violated constraints

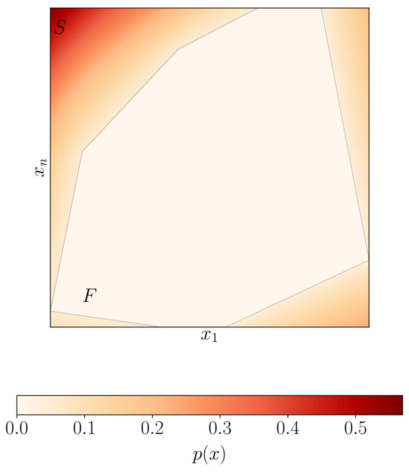


Figure 5.14.: Landscape of $p(x)$ with the norm of constraint violations

5.2.1.2. Adaptive Penalty

Since *adaptive* penalty methods are common as CHTs one of these is also evaluated. The framework *pygmo* offers such a technique developed by Farmani and Wright [17]. Thereby factors like the fitness value of best and worst solutions influence the penalty for infeasible solutions. In addition they introduce a measurement for infeasibility $\iota(\vec{x})$ (see Equation 5.5) which also take part at the penalization.

$$\iota(\vec{x}) = \frac{1}{n_c} \sum_{i=1}^{n_c} \frac{max(0, g_i(\vec{x}))}{g_{i,max}} \tag{5.5}$$

Hereby the infeasibility is calculated by the solution's constraint violation ($max(0, g_i(\vec{x})))$ which is normalized according to maximal violation occurred in the population ($g_{i,max}$).
The modified objective function φ is defined as shown in Equation 5.6 in case the population holds infeasible solutions. Otherwise the optimization processes on the original fitness function.

$$\varphi(\vec{x}) = f(\vec{x}) + \acute{\iota}(\vec{x})(f(\vec{x}_h) - f(\vec{x}_{worst})) \tag{5.6}$$

Where $\vec{x}_h$ is the highest objective of the population. $\vec{x}_{worst}$ is the solution with the highest infeasibility value. $\acute{\iota}(\vec{x})$ is calculated as follows:

$$\acute{\iota}(\vec{x}) = \frac{\iota(\vec{x}) - \iota(\vec{x}_{best})}{\iota(\vec{x}_{worst}) - \iota(\vec{x}_{best})} \tag{5.7}$$

Where $\vec{x}_{best}$ is the "best" solution in the current population. If there exist feasible solutions it is the solution with the lowest objective. In case all solutions in the population are infeasible the "best" solution is the solution with the lowest infeasibility value.
For further information on how this *adaptive* penalty method works with the factors influencing the penalty and how it is implemented to work with an OM, refer to read their paper and the source code of *pygmo*. In the following this CHT is referred to as *Farmani and Wright* (F&W).

5.2.2. Used Separatist Approaches

As representative of the CHT-category *separatist* three techniques are chosen. Twice the constrained optimization problem is reformulated in multi-objective unconstrained optimization problems and other time *Deb's rules* are implemented.

5.2.2.1. Multi-Objective Separatist

When reformulating in unconstrained multi-objective problems both times it is described as bi-objective problems (see Equation 5.8).

$$\min_{\vec{x}} \qquad \varphi(\vec{x}) = (\ f(\vec{x}),\ p(\vec{x})\ ) \tag{5.8}$$

One bi-objective transformation uses the amount of violated constraints ($p(\vec{x})$ like in Equation 5.3) as the second criterion and the other uses the norm of constraint violation ($p(\vec{x})$ like in Equation 5.4). These are referred to as avc$_s$ and ncv$_s$ respectively in the following.

5.2.2.2. Deb's rules

The third *separatist* method is the technique introduced by Deb [14]. The ranking is implemented via an adaptive penalty fuction like introduced in Equation 3.3 where f_{worst} is updated before the OM evolves the next population. Algorithm 7 shows the implementation of evolving the next population by an OM in combination with this CHT (Agorithm 3, line 3). This CHT is referred to as *Deb's rules* (DR) in the following.

Algorithm 7: Optimization process with Deb's rules

```
Input: $pop$ containing solutions; $OM$ optimization method
Output: $pop$ next population
1 $f_{worst} \leftarrow$ calc worst
2 $\varphi(\vec{x}) \leftarrow$ update objective function with $f_{worst}$
3 $pop \leftarrow$ evolve next population with $OM$ and updated $\varphi(\vec{x})$ as fitness-function
4 return $pop$
```

5.2.3. Used Feasibility Preserving Approaches

As representatives for the category *Feasibility Preserving* are two CHTs selected. These two CHTs are described in the following.

5.2.3.1. Repair

The first *Feasibility Preserving*-CHT is a *repair*-technique. Herein any infeasible solution evolved in the next population by an OM is mapped back to the nearest point in the *Feasible Domain* (F). The repair technique is the same as the method by the initialization process Repairing Initialization 5.1.2. The implementation of this repair method in combination with an OM is shown in Algorithm 8. This kind of implementation may differ from other *repair*-technique from the literature since the repairing operation is realized afterwards the

Algorithm 8: Optimization process with repair method

```
Input: $pop$ containing solutions; $OM$ optimization method
Output: $pop$ next population
1 $pop \leftarrow$ evolve next population with $OM$
2 foreach solution $x$ in $pop$ do
3     if $x$ is infeasible then
4         $x \leftarrow$ repair($x$)
5 return $pop$
```

5.2.3.2. Decoder

The other selected *Feasibility Preserving* method is a *Decoder*. Hereby, like previously mentioned in Chapter 3, the solutions are encoded in such a way that they are not able to leave F. The method of Koziel and Michalewicz is implemented for this purpose [32]. Thereby a hypercube (H) $[-1,1]^{n_x}$ is transformed on F. Consequently the transformation T maps each point $\vec{x}_0$ in F to a point $\vec{y}_0$ in H. The OM searches inside the hypercube instead of whole *Search Space* (S). The origin $\vec{\mathbf{0}}$ of H corresponds to a chosen *center point* $\vec{r}_0$ in F. As the *center point* the *Chebyshev Center* is selected. It describes the circle center of the largest possible hypersphere inside of a polytope which F is. Figure 5.15 shows the schema of T. The transformation calculates the direction from $\vec{\mathbf{0}}$ to the point $\vec{y}_0$ and its maximal possible distance to the border of H. The relative distance from $\vec{\mathbf{0}}$ to $\vec{y}_0$ in relation to the maximal possible is transfered to the relative distance in F from $\vec{r}_0$ in the same direction to its maximal possible distance.

To integrate this CHT into an OM the search space of the OM is H. By the fitness evaluation each solution is decoded and then evaluated by the origin f.

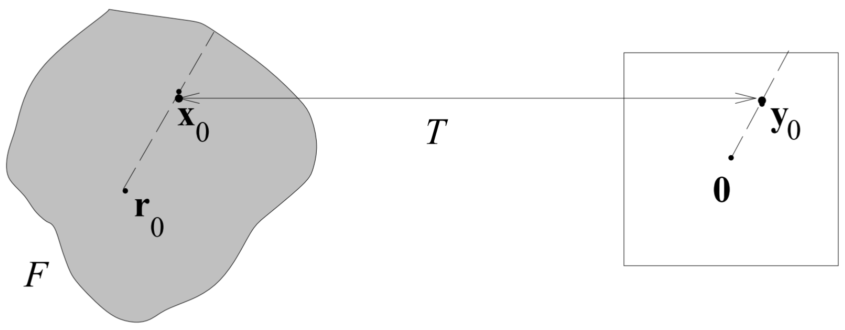


Figure 5.15.: Homomorphous Mapping[32, p. 6]: left the decoded solution $\vec{x}_0$ inside the feasible domain and right the encoded counterpart $\vec{y}_0$ in the hypercube

5.3. Used Optimization Methods

In this last section of this chapter considers the selected OMs. Different single objective optimization methods are chosen since the most CHTs transform the constrained optimization into a single objective unconstrained optimization problem. For the two CHT transforming the problem into bi-objective unconstrained optimization problems is one multi-objective optimization method selected. All chosen optimization methods are used from the framework *pagmo* and are described in the following briefly.

5.3.1. Single-Objective Optimization Methods

Because most selected CHTs work with single-objective OM the most common population-based meta-heuristics are chosen. This includes *Particle Swarm Optimization* (PSO), GA and DE. For comparison *Random Walk* (RW) is also included if the other optimization methods fulfill their purpose to guide the population towards a optimum or a simple randomly guided local search outperform them.

The offered PSO by *pygmo*[1] does not remember the particles local memory and the global best by interrupting the optimization process after every single generation. Therefore this PSO optimizes as a fresh start again and again. The particles memory about its own best and the global best loses its influence by becoming initialized

[1]`https://esa.github.io/pagmo2/index.html`

each generation freshly.
Therefore this PSO can not be used for this purpose and should be reimplemented. Considering the limited time for this thesis and the amount of other comparable OMs no PSO will not be tested with the implemented CHTs.

The scheme of *Genetic Algorithm* (GA) is originally established by Holland [20]. It equals the description in Algorithm 1 in Section 2.2.2. Since this OM do not require any information about the previous generation except the solutions itself it can be used in this thesis without any modifications. The framework *pygmo* offers an implementation of such an optimization method which is used in this thesis. The framework names it *simple Genetic Algorithm*.

The *Differential Evolution* (DE) offered by *pygmo* fulfills the criteria to work properly evolving single generations. Therefore this OM is used in combination with the introduced CHT. Its implementation matches the original described by Otieno, Adeyemo, Abbass, Sarker, Amir, Fisher, Rakesh, Babu, Babu, Jehan, et al. [46]. DE bases on *Evolutionary Algorithms* similar to the GA. The characteristic of DE is its special crossover technique. Thereby are three parents required. A difference vector is computed out of two of the selected parents. Then this difference vector is multiplied by a predefined weight and added onto the third selected parent. Afterwards a mutation on the newly created solution is applied. Otherwise different selection methods can be used in combination with the DE.

The last single-objective OM used in this thesis is *Random Walk* (RW). This technique takes a solution, randomly modifies it and compares it with the other solutions. In case that the newly produced solution is better than the previously found solutions it will replace it. This is equivalent to an uncontrolled local search since the modifications applied to a selected solutions do not create a completely new one but rather changes just a part. The implementation of *pygmo*'s *Simple Evolutionary Algorithm* matches a RW. Thereby per generation the current best solution out of the population is selected, modified and compared with the current worst solution. In case the new solution is better than the worst solution of the population the new solution replaces the worse solution.

5.3.2. Multi-Objective Optimization Method

As a multi-objective OM the *Non-dominated Sorting Genetic Algorithm II* (NSGA-II) of *pygmo* is selected. This OM is adapted from by Deb et. al [15].

This OM follows the scheme of a *Genetic Algorithm*. The only characteristic is by the environment selection since simply comparing the fitness value is not practicable with multiple objective anymore. Therefore pareto fronts are generated. In case that individuals from the same pareto front are compared the individual with the larger crowding distance wins. Further information what a mentioned step means and does in detail can be read in the related paper [15].

6. Evaluation

This chapter involves the description of the conducted experiments and their results. The first section deals with the setup of the experiments and their results of problem P_1 and the next section with P_2. Finally the results of the CHTs are summarized in the last section of this chapter.

6.1. Evaluation of P_1

In this section the experimental setup for the problem P_1 and their results are described in detail.

6.1.1. Experimental Setup for P_1

This subsection describes the experimental setup. This consists out of the combinations of the elements of an approach (the following subsection 6.1.1.1) and the parametrization of the chosen OMs. Also the metrics for the evaluation of the experiments are described and how the logging of these is implemented.

6.1.1.1. Experimental combinations

An experiment is equivalent to an approach described in the previous chapter. It consists of a combination of an *Initialization* (INIT), a *Constraint Handling Technique* (CHT) and an *Optimization Method* (OM). The combinatorial approaches are visualized in Figure 6.1. Every INIT is combined with every CHT and the appropriate OM, except ReI with the RC. This combination is excluded since it does not differ from the RaI because in both cases the same repairing technique is applied to the population.
Obviously the CHTs transforming the single objective constrained optimization

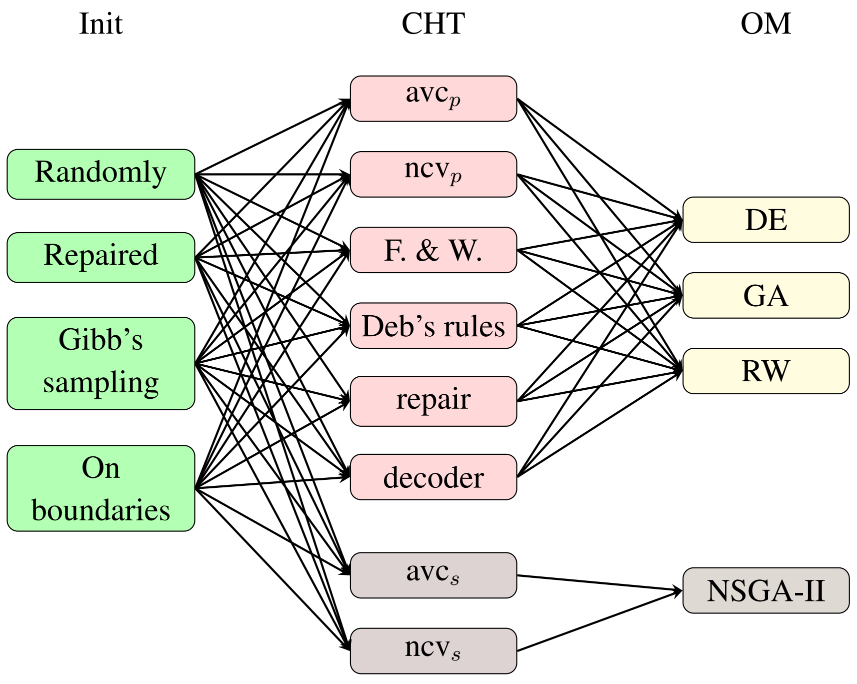


Figure 6.1.: Combinatorial components of the test-cases for P_1

problem to a unconstrained multi-objective problem require a multi-objective optimization method. Therefore these CHT are combined with multi-objective optimization method and the others are tested with the single objective optimization methods.
All combinations together equals a total an amount of 77 approches to be evaluated. To estimate the performance of an approach each one is conducted 21 times.

6.1.1.2. Parametrization of the OMs

The population size for each combination is twice n_x after the preprocessing of the problem. For P_1 it is 60. The introduced OMs from the previous chapter are all provided by the framework *pygmo*. Since this thesis concentrates on CHTs the default configuration of the OMs by *pygmo* are used and are not further modified.
The *Differential Evolution* (DE) offered by *pygmo* matches the original introduced one in [46]. The default values for *weight coefficient* is 0.8, the *crossover probability* is 90% and the *mutation variant* is $rand/1/exp$. Further information, what these settings mean and how they influence the optimization process, are available on the

homepage of *pygmo*[1] and can be read in [46].
The *simple Genetic Algorithm* by *pygmo* follows the scheme of the originally established *genetic algorithm* by Holland [20]. The default in *pygmo* for *crossover type* is *exponential* with a probability of 90%, for the *selection schemes* is a *tournament selection* with two randomly chosen individuals and the *mutation type* is *polynomial*. For further information, the reader is also referred to the homepage of the framework.
The last single-objective OM used in this thesis is *Random Walk* (RW). The OM *simple Evolutionary Algorithm* provided by *pygmo* corresponds to RW. Its implementation is equivalent to a (N+1)-*Evolutionary Algorithm*. Thereby the best solution of the current generation is chosen, uniformly mutated and inserted back into the population whereby the worst individual will be replaced. Since this implementation matches with a Random Walk, whereby the local area is explored and in case of discovering a better solution it is memorized.
As a multi-objective OM the *Non-dominated Sorting Genetic Algorithm II* (NSGA-II) of *pygmo* is selected. This OM is adapted from by Deb, Pratap, Agarwal, Meyarivan, and Fast [15]. As with the other OMs, the default settings are used for this one as well. Another special case is the INIT OB in combination with NSGA-II. Thereby $\mathcal{S}$ defined by box-constraints is shrunken by 0.1% along every axis. The shrunken boundaries are moved by 0.05% of the range towards the center of $\mathcal{S}$. In this way the Initialization on boundaries spawns solutions not on the boundaries but rather short before these. The optimization process itself performs on the original $\mathcal{S}$. For clarification *Figure* 6.2 shows such an initialization in a shrunken $\mathcal{S}$ by a factor of 10% so that the effect is easier to recognize.

6.1.1.3. Metrics

The following paragraphs describe six chosen metrics to estimate the behavior and performance of the test-cases. For each of the metrics different statistical representation per population are evaluated. There the mean, median or minimal values of the metric per corresponding population are recorded during the experiments. The value of a metric is recorded after at least one thousand fitness evaluations. This can vary per test-case. For example the OM RW requires the fitness evaluation of a single individual to evolve the next generation. Therefore is the loop checking the termination criteria (see lines 2 and 3 in Algorithm 3) per fitness evaluation. It is

[1] `https://esa.github.io/pagmo2/index.html`

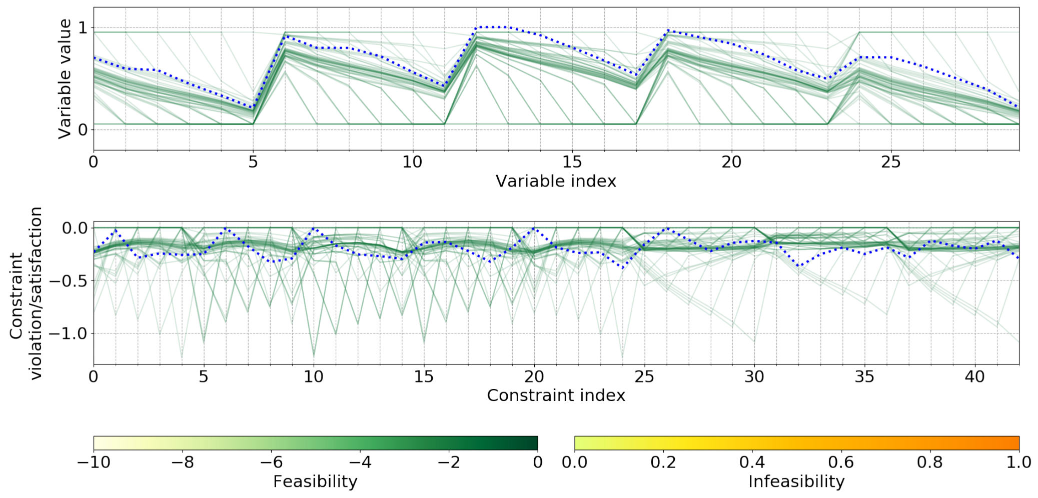


Figure 6.2.: INIT on boundaries within a shrunken $\mathcal{S}$ by 10% of P_1

different with the combination of the GA with the Hereby this OM requires n_x fitness evaluations to generate the next population plus the fitness evaluations in case of infeasible solutions with are repaired afterwards.

Amount of violated constraints (avc)

In the case of infeasible solutions it will be useful to have a measurement to compare the infeasibility of the population. A simple metric to fulfill this information gap is the amount of violated constraints. It is calculated like described in Equation 5.3. Hereby the median of the respective population recorded. In contrast to the mean the median is more likely stable against outliers.

Norm of constraint violations (ncv)

Because the metric avc captures just how many constraints are violated and not how strong the constraints are violated avc can be misleading in term of evaluating infeasibility. Therefore a second metric to measure infeasibility is chosen. Consequently the euclidean norm of the constraint violations will be recorded. It is calculated like described previously in Equation 5.4. As a representational value for the population also the median is recorded during the optimization process.

Objective (obj)

To observe how well a test-case performs the objective of the original problem will be also tracked. Hereby is the mean objective of the solutions of the corresponding population logged. In this way the overall performance of an approach can be monitored. If it converges towards minima (may it be local or global) or not.

Distance to the optimum (d2o)

Because the toy-problems are artificially made up the global optimum is known. To monitor if a test-case reaches the global optimum by a lucky initialization close to the optimum or if it is able to converge towards the global optimum by its own the euclidean distance of the population to the global optimum is recorded. Since reaching the global optimum is the main goal to achieve the minimal distance of a solution to the global optimum is logged. In this way it is evaluable if the approach is appropriate for the considered problems.

Distance between the solutions (dbs)

In case of identical solutions in the population valuable fitness evaluations are conducted unnecessarily. To observe this situation the mean distance between the individuals of the population is also tracked. This metric is also an indicator for the diversity of the current population.

Percentage of feasible solutions inside the population(%_f)

To track if an the specific testcase leaves the feasible domain with some individuals or is it possible to get them back into the feasible domain the percentage of feasible solution inside the current population will be captured.

6.1.2. Results for P_1

This section considers the results of the approaches mentioned in the previous section. An analyzed approach consisting of the three elements INIT, CHT and OM is referenced as such a triple. With respect to the high amount of experiments the results are analyzed grouped by their CHT.
Concerning the 21 runs per approach only a representative run is analyzed and compared with the other approaches. Therefore the runs of an approach are sorted regarding to the minimal d2o in the final population. Afterwards the median of this

sorting is chosen as a representative for its approach. The progress for each metric of the representatives is plotted. Along the x-axis is the amount of function evaluations of f.

Static Penalty: amount of violated constraints (avc$_p$)

Figure 6.3 shows the progress of the optimization processes with the CHT avc$_p$. The found results of the mean of objective values vary below 1 (Figure 6.3, plot obj_mean). Since avc$_p$ penalizes by full numbers only any infeasible solution becomes automatically worse than a feasible one.
According to the avc which is in this case also the penalty factor it is recognizable that it is continuously decreasing. Also the obj decreases in nearly all approaches. Only by RaI initialized approaches with RW and GA as their OM seem to increase the mean obj (Figure 6.3, plot obj_mean). This effect appears simultaneously when avc has a jump to a smaller value (Figure 6.3, plot avc_median). Since obj does not differ between feasible and infeasible solutions it means that these approaches found worse solutions compared by their obj but with a smaller amount of violated constraints (Figure 6.3, plot obj_mean).
Another interesting observation is that this CHT keeps nearly all feasible solution in case it is initialized with them (Figure 6.3, plot %_f). Only DE reaches F by initialization via RaI (Figure 6.3, plot %_f).
DE keeps independent of the INIT the highest diversity between the solutions (Figure 6.3, plot dbs_mean). But this CHT does not converge towards the global optimum whereby the approaches with DE become closest (Figure 6.3, plot d2o_min).

Static Penalty: norm of constraint violation (ncv$_p$)

The other static penalty is ncv$_p$. Figure 6.4 shows the appropriate plots of its optimization processes. Similar to avc$_p$ also the approaches with DE keep the highest diversity and become closest to the optimum (Figure 6.4, plot dbs_mean and d2o_mean). But this CHT converges faster towards F than avc$_p$. This is observable by the sharply decreasing ncv and avc (Figure 6.4, plot avc_median and ncv_median).
In consideration to %_f all randomly initialized approaches reach F (Figure 6.4, plot obj_mean). Furthermore some feasible initialized approaches in combination with RW and GA leave F with the whole population (Figure 6.4, plot obj_mean). This can be explained that infeasible areas close to F result in such a small penalty

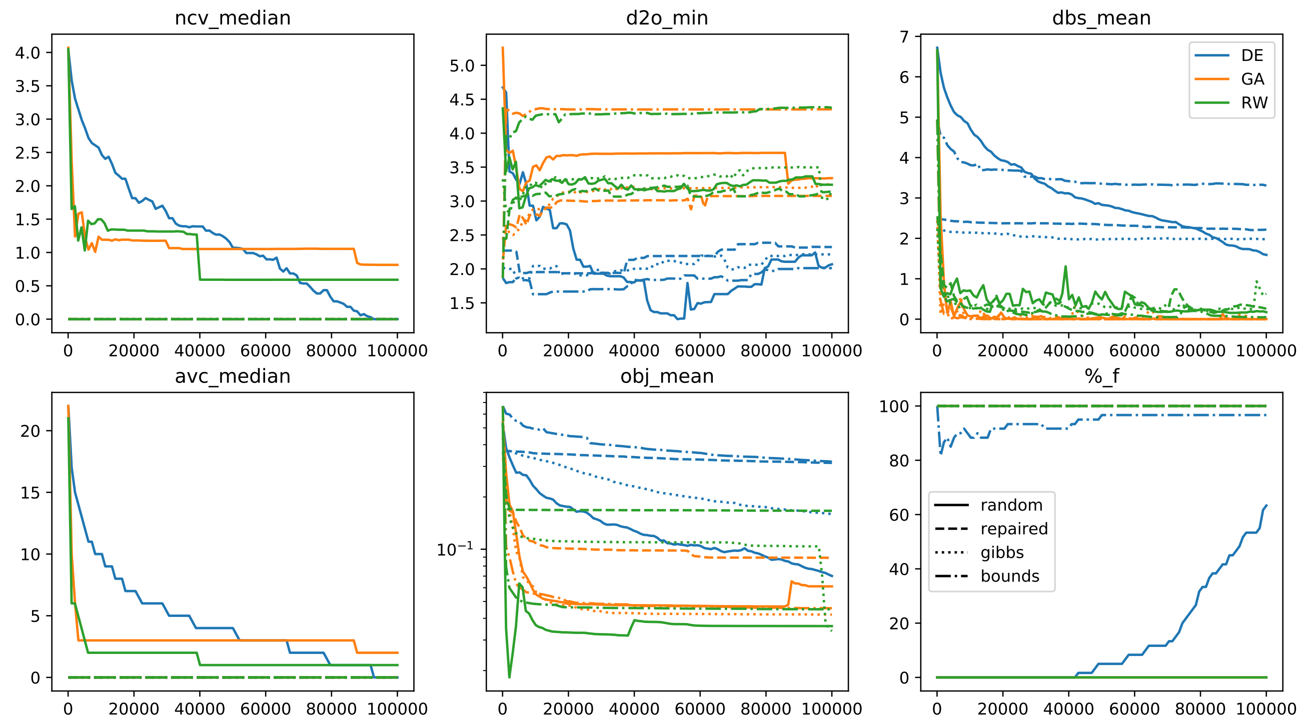


Figure 6.3.: Static Penalty-CHT avc$_p$ for the problem P_1; the color of the lines indicates the OM and the linestyle the used INIT

factor thus the penalized fitness value stays better than the objective of feasible solutions.

Adaptive Penalty: Farmani and Wright (F&W)

By F&W not any randomly initialized approach reaches F (Figure 6.5, plot obj_mean). All approaches with infeasible populations stay in the infeasible space and do not find F. Furthermore only approaches with RW stay mostly feasible by a initialization inside of F (Figure 6.5, plot obj_mean).
Although DE keeps in combination with this CHT also the highest diversity the mean dbs is smaller than by the static penalties (Figure 6.5, plot dbs_mean).
The best objective is reached by RaI-F&W-GA (Figure 6.5, plot obj_mean). However, it must be taken into account that the measured obj is the mean of the population and this approach compared by its dbs seem to result with similar solutions in the population (Figure 6.5, plot dbs_mean). Also this approach has the highest avc and ncv (Figure 6.5, plot avc_median and ncv_median) Compared by d2o this approach is the second worst. Therefore it can be assumed that this approach emerges similar or even several equal infeasible solutions. This effect is let assume that this CHT can stuck in a local optimum in the infeasible space.
Also this CHT does not converge towards the global optimum (Figure 6.5, plot d2o_min).

Separatist: Deb's rules (DR)

The representative progresses of *Deb's rules* (DR) implemented as an adaptive penalty is plotted in Figure 6.6. The approaches combined with DE differ from the others by d2o, obj and dbs (Figure 6.6, plot dbs_mean, d2o_min and obj_mean). Although these result the worst obj they get closer to the global optimum compared with the others.
All approaches with this CHT are not able to keep their population inside of F (Figure 6.6, plot %_f). %_f falls by all combination below 50%. This may be due to the implementation of DR. Thereby after each generation f is updated according to current worst found objective in the population. By this way only new found solutions get the modified objective and not the already evaluated solutions from the last generation which are kept in the population.
Furthermore it is observable that ncv fluctuates compared to the continuous decreasing values by the previous CHTs (Figure 6.6, plot ncv_median).

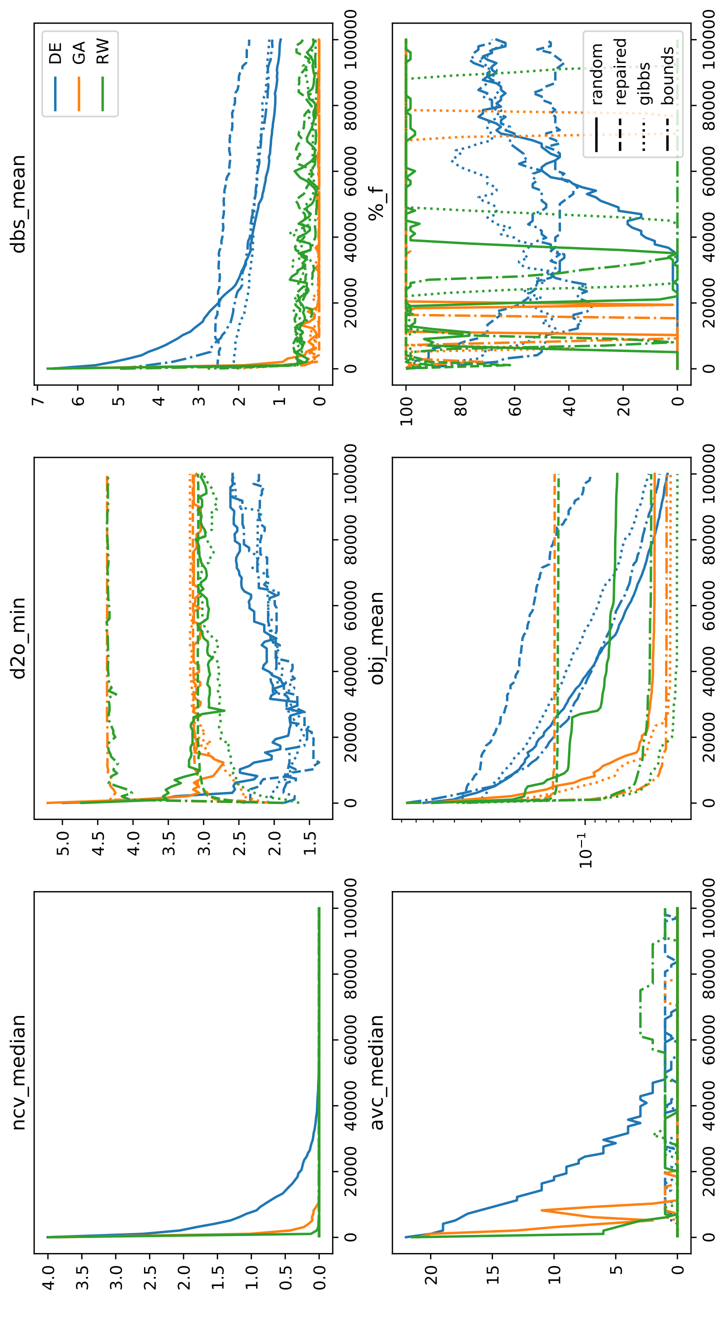


Figure 6.4.: Static Penalty-CHT ncv$_p$ for the problem P_1; the color of the lines indicates the OM and the linestyle the used INIT

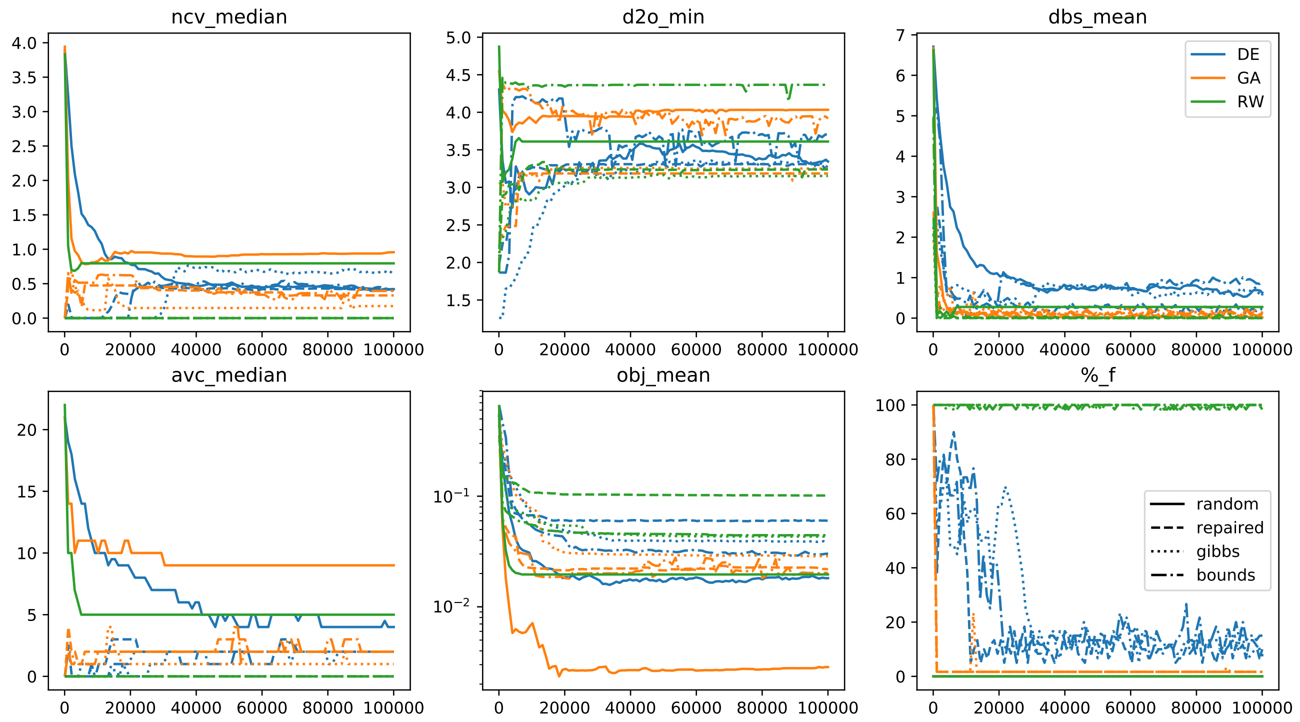


Figure 6.5.: Adaptive Penalty-CHT F&W for the problem P_1; the color of the lines indicates the OM and the linestyle the used INIT

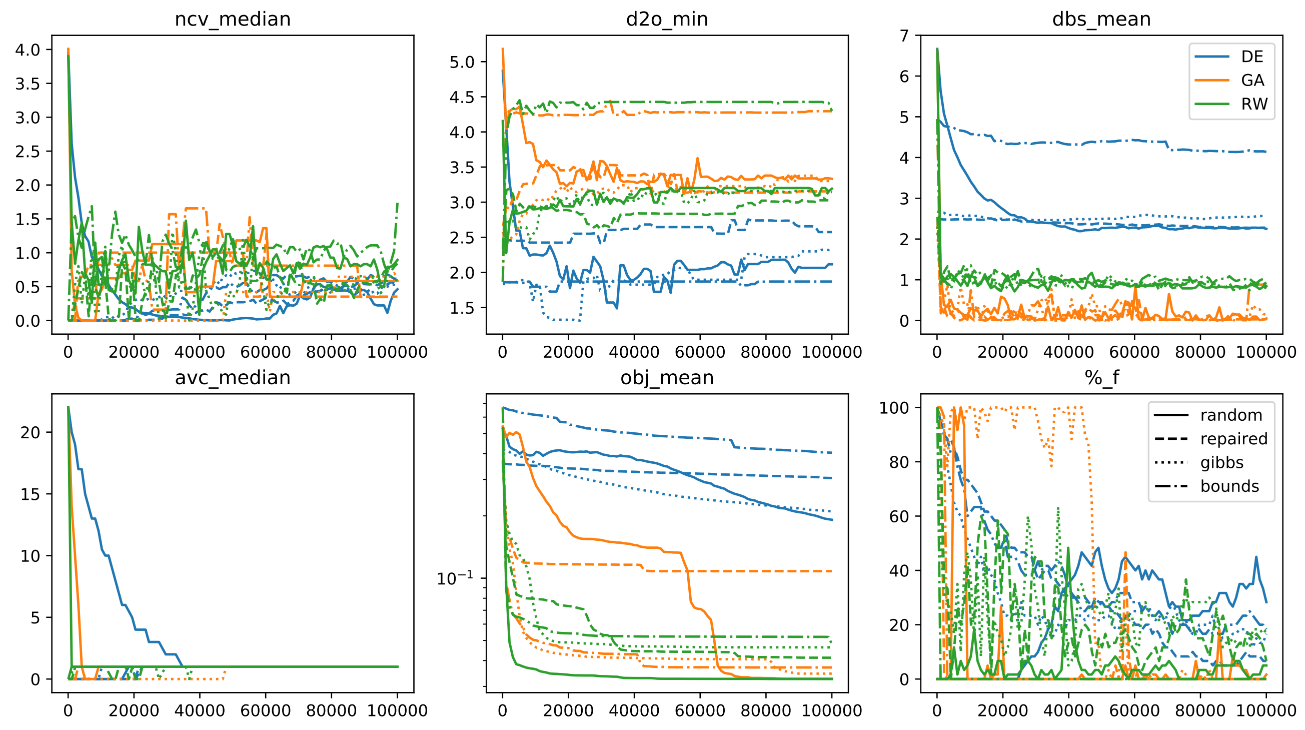


Figure 6.6.: Separatist-CHT DR for the problem P_1; the color of the lines indicates the OM and the linestyle the used INIT

Separatist: Bi-objective (avc$_s$ and ncv$_s$)

Unlike the other CHT the separatist-CHTs which transfers the constrained optimization problem into bi-objective unconstrained problems are plotted together shown in Figure 6.7. Since only one OM, NSGA-II, is evaluated with these CHTs avc$_s$ and ncv$_s$ are combined in one figure distinguished by color.
According to the second objective these two separatist approaches differ by avc and ncv (Figure 6.7, plot ncv_median and avc_median). Both approaches lose feasible solutions (Figure 6.7, plot %_f). This is due to the fact that NSGA-II tries to reach a population balancing both objectives. Since the second objective requires infeasible solutions to result a diversity the OM evolves a combination of feasible and infeasible solutions inside the population as long as infeasible solutions have a better obj than feasible solutions.
The approaches initialized by RaI do not reach F (Figure 6.7, plot %_f). Feasible initialized approaches are able to keep feasible solutions whereby the percentage by avc$_s$ is higher than by ncv$_s$.
The d2o seems to be stuck after ca. 50.000 objective evaluations (Figure 6.7, plot d2o_min).

Feasibility Preserving: repair (RC)

Compared to the other CHTs the measured metrics of RC do not fluctuate that much. Since the this CHT is listed to feasibility preserving avc and ncv are constant zero and %_f is fix by 100% (Figure 6.8, plot avc_median, avc_median and %_f).
The approaches consisting DE keep a higher diversity and seem to perform a bit better according to d2o (Figure 6.8, plot d2o_min and dbs_mean). Nevertheless the resulting obj is worse than by the other OMs. However this could also be induced by the mean of obj and the lower diversity (Figure 6.8, plot dbs_mean).

The results of *feasibility preserving CHT-Decoder* (DC) are not presented because they are invalid. The optimization processes yielded infeasible solution which should not be possible by its definition. Which is due to the assumption that the implementation resulted in inaccuracy or consisted a bug. However due to the restricted processing time for this master thesis this CHT will not be analyzed further and therefore can be conducted for future work. Nevertheless the invalid results of DC can be viewed in the appendix.

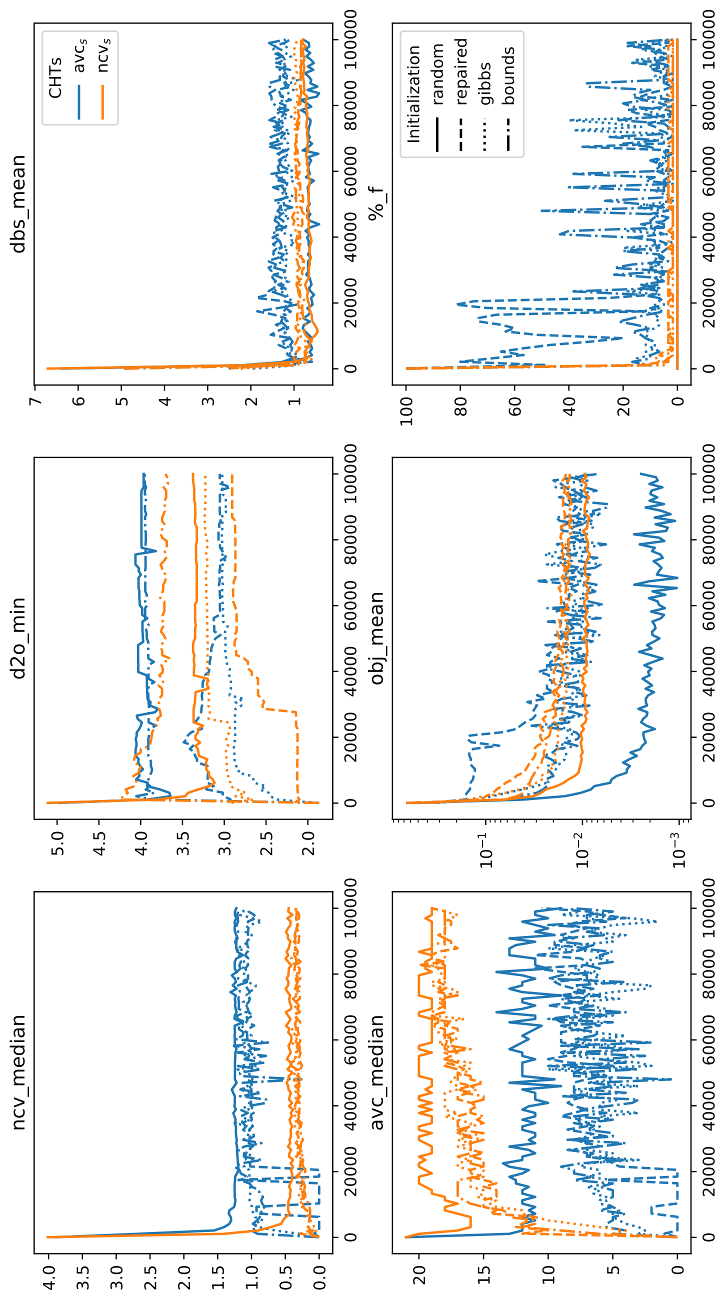


Figure 6.7.: Separatist-CHT avc$_s$ and ncv$_s$ for the problem P_1; the color of the lines indicates the CHT and the linestyle the used INIT

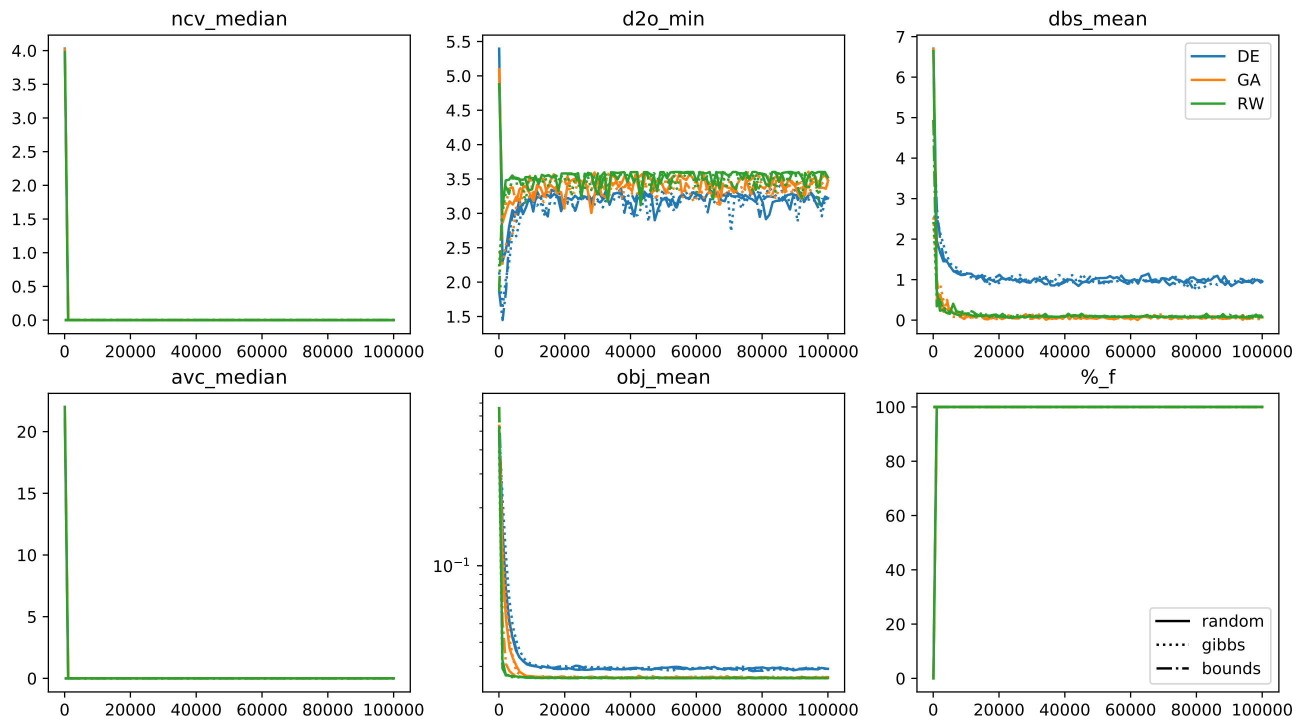


Figure 6.8.: Feasibility Preserving-CHT RC for the problem P_1; the color of the lines indicates the OM and the linestyle the used INIT

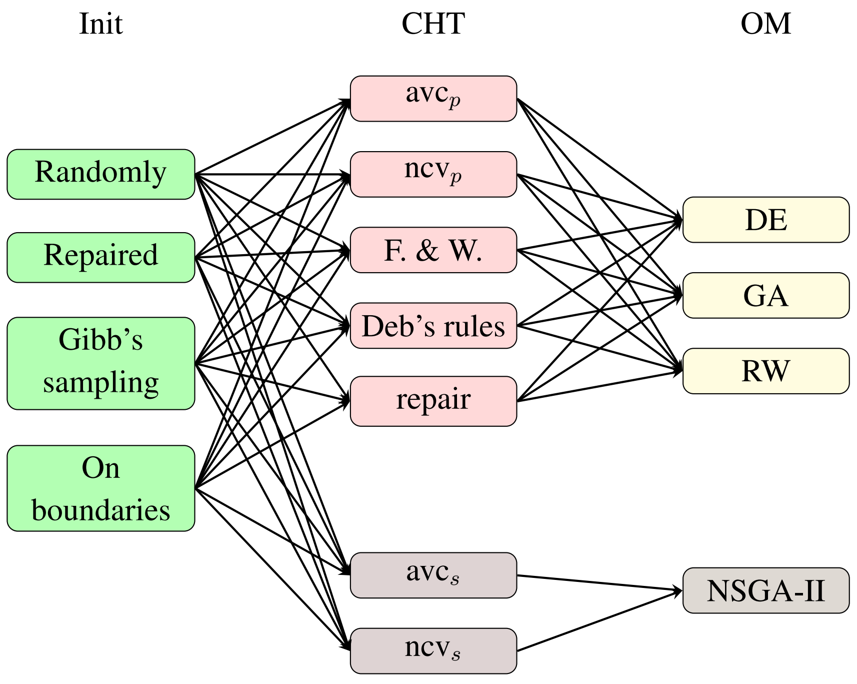


Figure 6.9.: Combinatorial components of the test-cases for P_2

6.2. Evaluation of P_2

6.2.1. Experimental Setup for P_2

The experimental setup of the approaches for P_2 is similar to the combination for P_1 except DC. This CHT is omitted completely. Figure 6.9 visualizes the evaluated approaches for P_2. The parametrization of the OMs is the same like by P_1. Only the population size is increased twice n_x up to 268. The logging process is also the same with the equal metrics. The termination criteria of the optimization processes is also reaching 100.000 function evaluations of f.

6.2.2. Results for P_2

The analyses of the results of the approaches with P_2 are equal to the analyses with P_1. The results are grouped by their CHT except the bi-objective separatist CHT which are plotted together since they are combined only with the only chosen multi-objective OM.

Static Penalty: amount of violated constraints (avc$_p$)

Figure 6.10 shows the progress of the approaches with avc$_p$ as their CHT for P_2 along the amount of function evaluations of f. Equal to the smaller problem the mean objective per population of the found results varies below 1.
This CHT also keeps feasible solutions in case it is initialized with feasible solutions (Figure 6.10, plot %_f). If avc$_p$ is initialized via RaI it is not able to reach F though it moves towards F. This can be observed by the decreasing of avc and ncv (Figure 6.10, plot avc_median and ncv_median). Reaching F may be possible in case the termination criterion would be a higher amount of function evaluations of f.
The approaches uses DE as OM seem to stuck in combination with INITs which produces feasible solutions only (Figure 6.10, plot d2o_min, dbs_mean and obj_mean). This effect can be explained that in case of producing solutions these are worse than the already found. Since creating also feasible solutions with DE requires that the modification vector applied on the third parent (see Section 5.3.1 describing DE) does not point outwards F.
Compared by the reached obj the approaches using RW perform best with this CHT (Figure 6.10, plot obj_mean). But they also have the lowest diversity (Figure 6.10, plot dbs_mean).

Static Penalty: norm of constraint violation (ncv$_p$)

According to the penalty factor ncv decreases sharply by the approaches which are initialized via RaI (Figure 6.11, plot ncv_median).
The approaches with DE seem to be stuck like by avc$_p$ and change over the amount of function evaluations of f only slightly (Figure 6.11, plot d2o_min, dbs_mean and obj_mean). This effect can also be explained only by the technique DE uses to create new solutions. Thereby the new solutions have to be rejected since their objective is worse than the objective of the current population.
No approach initialized via RaI reaches F (Figure 6.11, plot %_f). The other approaches do not necessary their feasible solutions and accept infeasible solutions in exchange. Nevertheless overall the mean of obj decreases continuously (Figure 6.11, plot obj_mean). This observation can be explained like already described by P_1 with the same CHT. Infeasible solutions with a low penalty factor may be preferred by the OM with the modified objective function over feasible solutions.

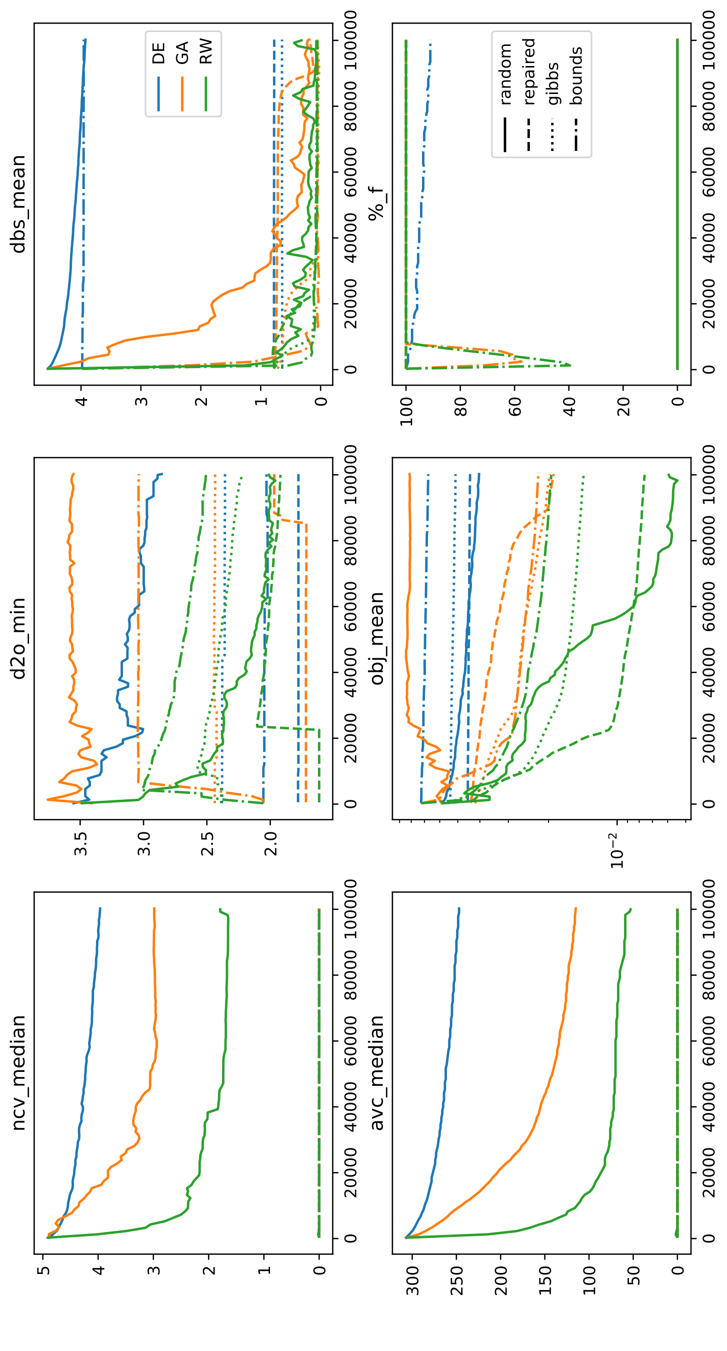


Figure 6.10.: Static Penalty-CHT avc$_p$ for the problem P_2; the color of the lines indicates the OM and the linestyle the used INIT

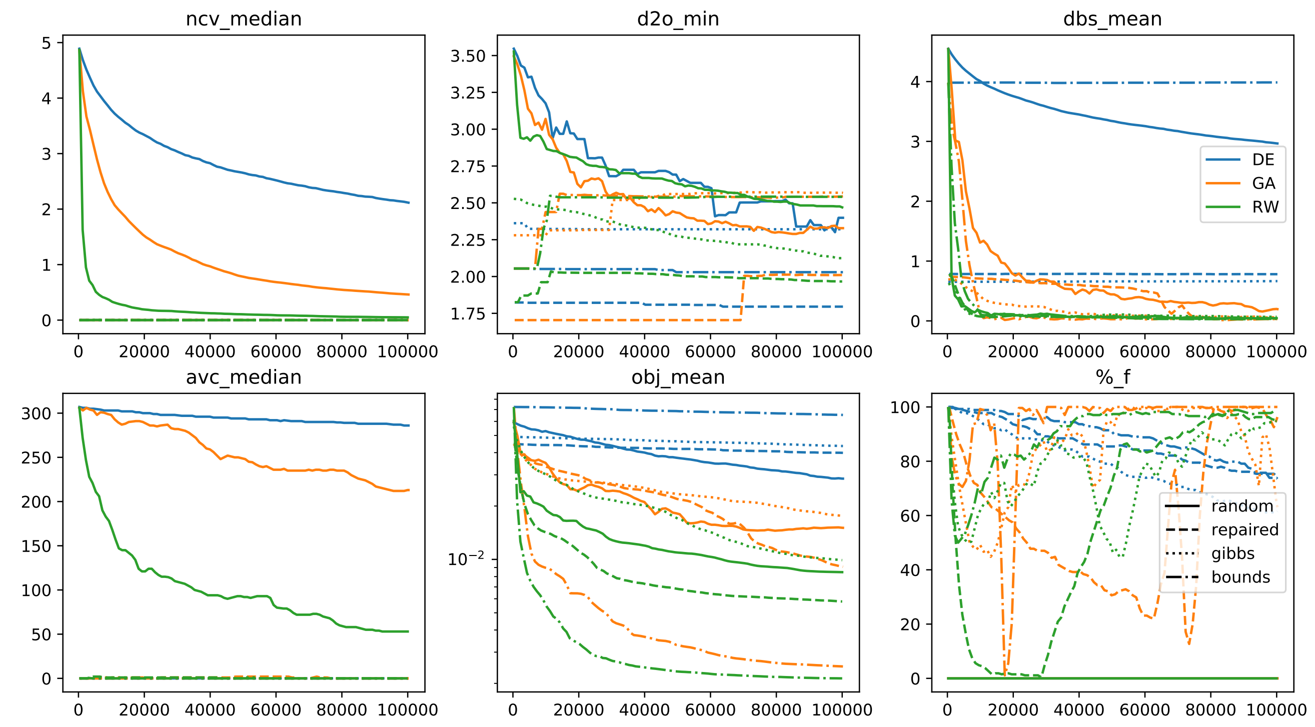


Figure 6.11.: Static Penalty-CHT ncv_p for the problem P_2; the color of the lines indicates the OM and the linestyle the used INIT

Adaptive Penalty: Farmani and Wright (F&W)

The approaches with this CHT leave F except for these which are initialized with feasible solutions only and use RW as OM (Figure 6.12, plot %_f). This is expectable since F&W is also attracted by infeasible local optima.
F&W in combination with RW and RaI does not reaches F (Figure 6.12, plot %_f). Nevertheless it performs as the best according to its d2o and obj (Figure 6.12, plot d2o_min and obj_median). This approach could reach the global optima if it could optimize with a higher amount of function evaluations of f. Also DE in combination with RaI seems to converge towards the global optima (Figure 6.12, plot d2o_min).
Unexpected the GA with RaI converges away from F (Figure 6.12, plot avc_median and ncv_median). This behavior suggests that P_2 has preferable local optima outside of F.

Separatist: Deb's rules (DR)

By this CHT the approaches seem to be stuck similar like avc$_p$ and ncv$_p$ (Figure 6.13, plot d2o_min, dbs_mean and obj_mean).
But other than this CHT with P_1 can keep feasible solution and the percentage of them in the population does not fluctuate like by P_1 (Figure 6.13, plot %_f).
RW performs according to obj better than the other approaches (Figure 6.13, plot obj_mean). Only GA initialized via OB can keep up. Whereby the diversities of these are also the smallest (Figure 6.13, plot dbs_mean).

Separatist: Bi-objective (avc$_s$ and ncv$_s$)

The bi-objective separatist approaches do not perform in such clustered structure with P_2 like with P_1 according to avc and ncv (Figure 6.7 and 6.14, plot ncv_median and avc_median). The outlier is RaI-avc$_s$-NSGA-II which performs as the worst according to avc.
Apart from that the random initialized approaches perform well according to their obj and d2o (Figure 6.14, plot obj_mean and d2o_min). Whereby these two approaches do not reach F (Figure 6.14, plot %_f).

Feasibility Preserving: repair (RC)

In comparison to RC with P_1 the results of this CHT with P_2 are more diverse (Figure 6.8 and 6.15, plot d2o_min and obj_mean).
According to obj the approaches using DE performs worst (Figure 6.15,

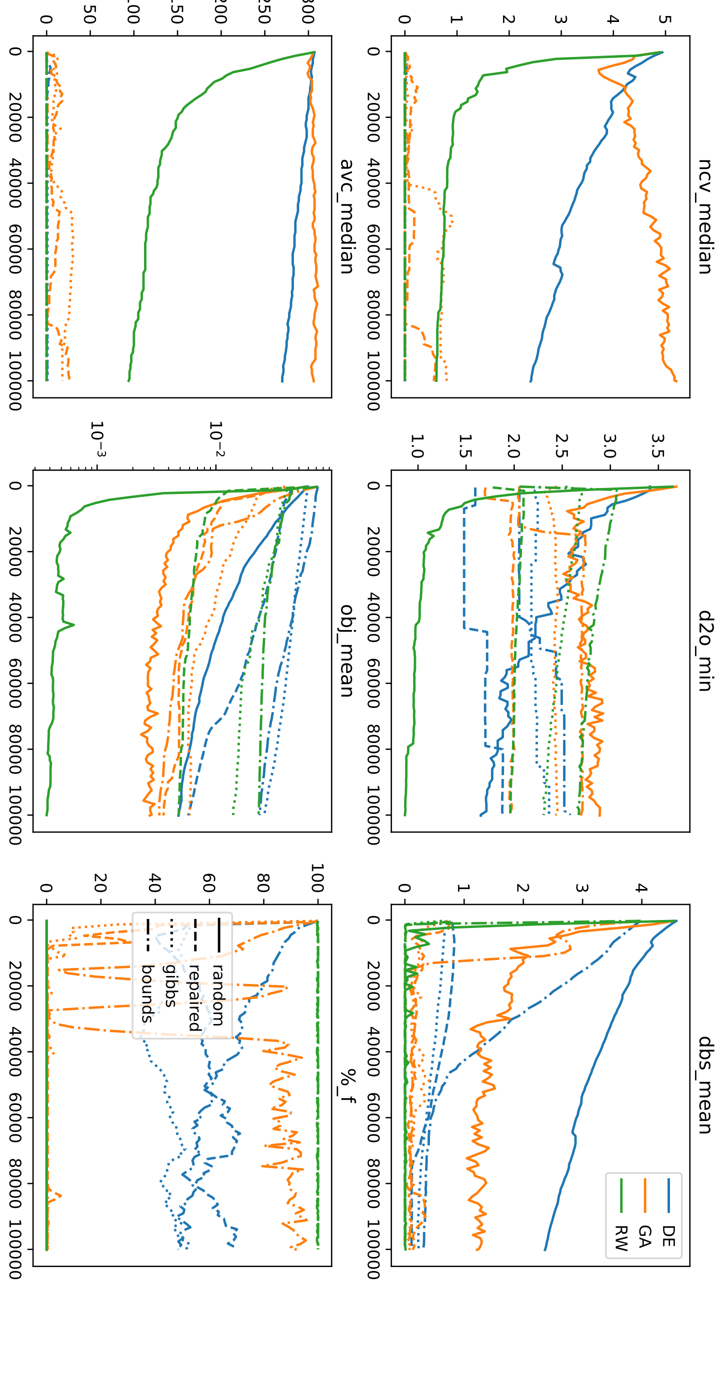


Figure 6.12.: Adaptive Penalty-CHT F&W for the problem P_2; the color of the lines indicates the OM and the linestyle the used INIT

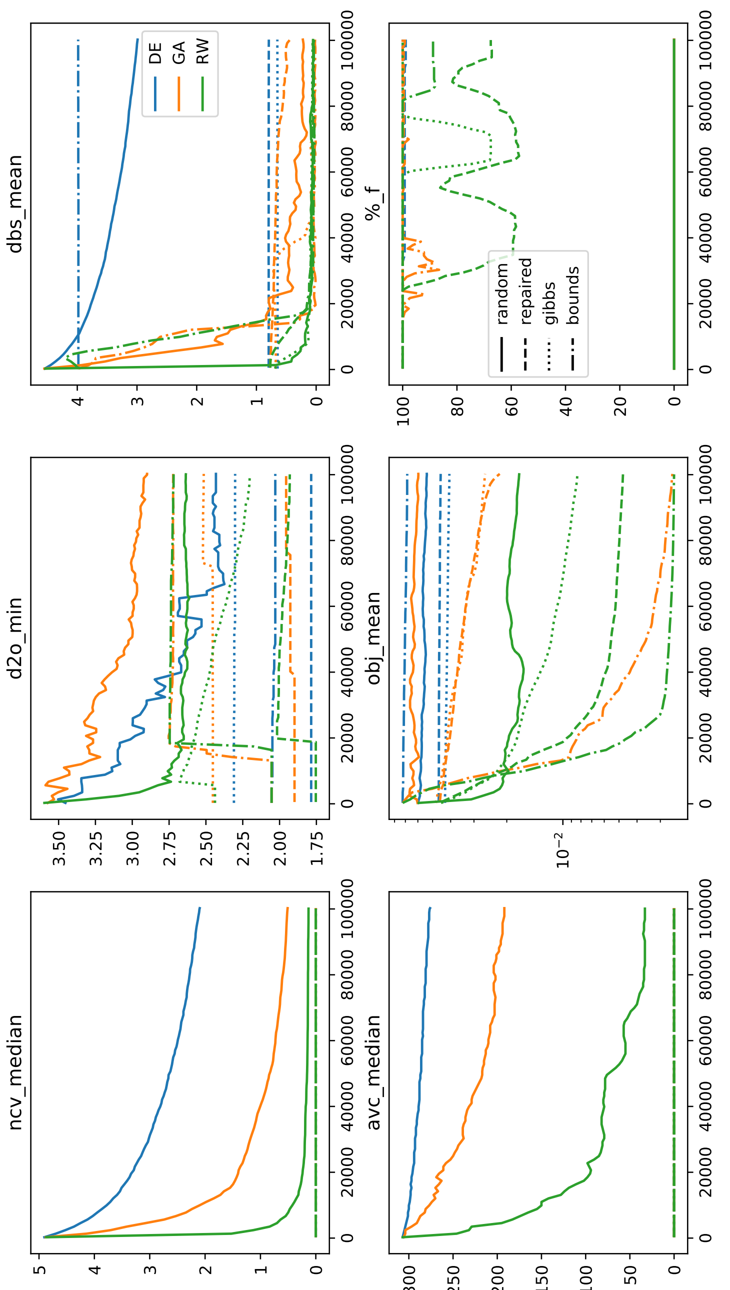


Figure 6.13.: Separatist-CHT DR for the problem P_2; the color of the lines indicates the OM and the linestyle the used INIT

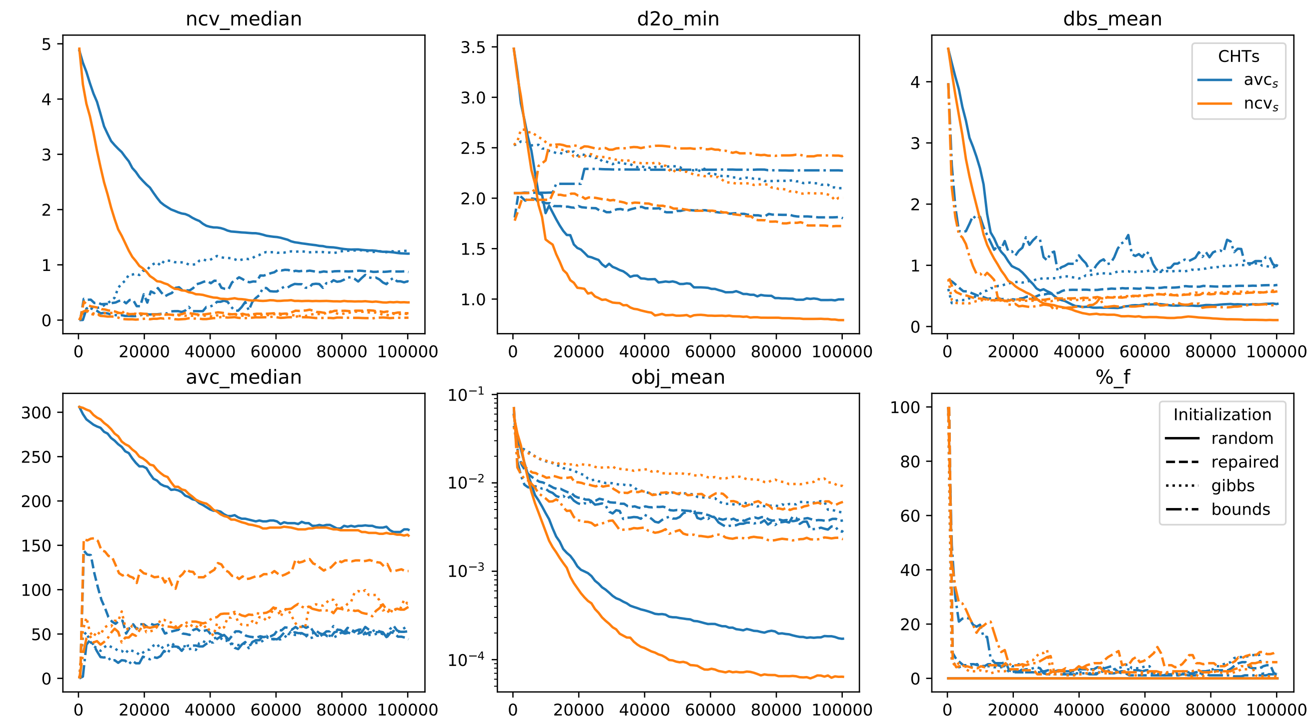


Figure 6.14.: Separatist-CHT avc_s and ncv_s for the problem P_2; the color of the lines indicates the CHT and the linestyle the used INIT

plot obj_mean). However it is to mention that obj is the mean of the population and can be misleading by a high diversity which in fact is the case (Figure 6.15, plot dbs_mean). It is also recognizable that the approach OB-RC-DE performs with RW initialized via RaI and GS as the best with this CHT regarding to d2o (Figure 6.15, plot d2o_min).

6.3. Discussion

This section discusses the results as in their entirety. All approaches can be compared according their minimal distance to the global optimum in the final population. The Tables 6.3 and 6.3 show d2o_min for each approach. In addition the related interquartile ranges are also listed in the tables. The best approach with respect to d2o for P_1 is OB-DR-DE. Closest to the optimum by P_2 is OB-RC-DE. Even if for both problems the initialization OB and the optimization method DE performs best, other approaches get close to it also.

As a main point is to mention that the metric measuring the distance to the global optimum can be realized just with the artificial problem where these are well known. Comparing the performance of the approaches by obj instead of d2o leads to another valuation because in some cases the approaches perform only on one metric well and not always on both simultaneously.
Another weakness of the metrics regarding a detailed interpretation is the chosen statistical measurement of a metric. As an example it would be more helpful to know the minimal instead of mean of obj. Also the ranges of reached avc and ncv (minimal and maximal reached value in the corresponding population) could be useful to estimate an infeasibility of the population.

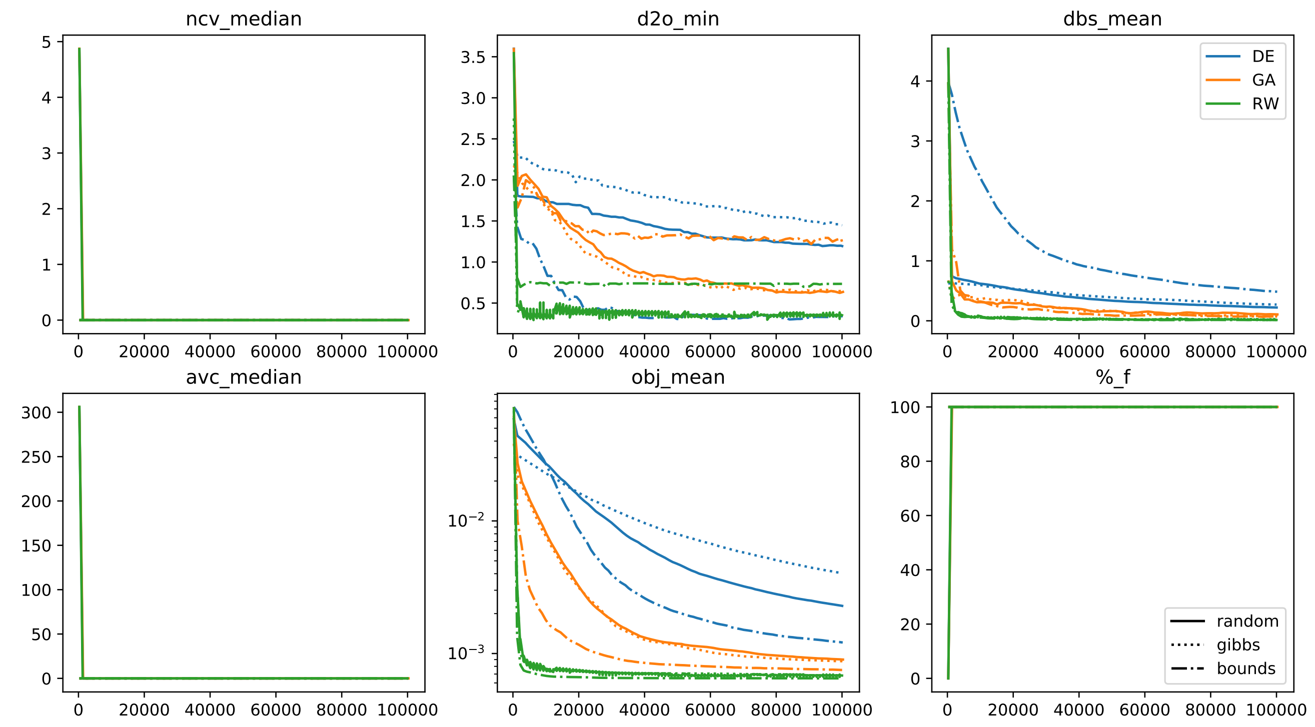


Figure 6.15.: Feasibility Preserving-CHT RC for the problem P_2; the color of the lines indicates the OM and the linestyle the used INIT

	INIT	OB		GS		RaI		ReI	
		median	(iqr)	median	(iqr)	median	(iqr)	median	(iqr)
CHT	**OM**								
	DE	1.870156	(0.169954)	2.330155	(0.199864)	2.115940	(0.161415)	2.574563	(0.311259)
Deb's rule	GA	4.294767	(0.128412)	3.300492	(0.393890)	3.332714	(0.714005)	3.153674	(0.394895)
	RW	4.297686	(0.099889)	3.139116	(0.221705)	3.192407	(0.345274)	3.027026	(0.318638)
	DE	3.695380	(0.364233)	3.350555	(0.273938)	3.369001	(0.233995)	3.309202	(0.485816)
F. & W.	GA	3.917548	(0.210818)	3.250433	(0.294835)	4.032919	(0.273619)	3.184976	(0.257554)
	RW	4.366054	(0.193001)	3.151881	(0.403630)	3.610285	(0.401857)	3.238251	(0.521583)
	DE	2.011611	(0.109959)	2.213998	(0.216002)	2.068443	(0.270744)	2.322658	(0.417265)
avc$_p$	GA	4.351348	(0.048871)	3.245890	(0.274313)	3.337875	(0.494168)	3.076622	(0.629833)
	RW	4.373237	(0.175876)	3.173634	(0.490469)	3.240757	(0.492786)	3.145980	(0.798206)
	DE	2.591299	(0.158101)	2.584288	(0.243631)	2.595641	(0.166314)	2.216237	(0.165777)
ncv$_p$	GA	4.367195	(0.098893)	3.192956	(0.316287)	3.100893	(0.356695)	3.152999	(0.374363)
	RW	4.354097	(0.121592)	2.995953	(0.546249)	3.018735	(0.501634)	3.079392	(0.360221)
	DE	3.219966	(0.166493)	3.230011	(0.042632)	3.220210	(0.069773)	/	(/)
repair	GA	3.473521	(0.207684)	3.436151	(0.130645)	3.471572	(0.105667)	/	(/)
	RW	3.523204	(0.159955)	3.570538	(0.148446)	3.523210	(0.148992)	/	(/)
ncv$_s$	NSGA-II	3.702016	(0.072450)	3.225980	(0.192829)	3.373823	(0.136364)	2.904464	(0.213291)
avc$_s$	NSGA-II	3.950870	(0.101732)	3.023598	(0.242568)	3.964708	(0.220457)	3.053741	(0.314362)

Table 6.1.: In this table are the median over the runs of the minimal *distance to the global optimum* (d2o) in the final population for each approach with P_1 (which are a combination out of *Initialization* (INIT), *Constraint Handling Technique* (CHT) and *Optimization Method* (OM))

	INIT	OB		GS		RaI		ReI	
		median	(iqr)	median	(iqr)	median	(iqr)	median	(iqr)
CHT	**OM**								
	DE	2.030231	(0.013528)	2.301663	(0.252272)	2.426906	(0.125487)	1.783722	(0.096033)
Deb's rule	GA	2.722293	(0.070727)	2.501691	(0.346559)	2.897723	(0.329872)	1.954037	(0.322937)
	RW	2.717994	(0.014745)	2.201443	(0.249164)	2.636207	(0.444659)	1.920535	(0.247987)
	DE	2.604283	(0.213890)	2.359871	(0.530070)	1.638078	(0.182085)	1.876712	(0.202442)
F. & W.	GA	2.695234	(0.227511)	2.429077	(0.369383)	2.888929	(0.566553)	1.958166	(0.260840)
	RW	2.672183	(0.025045)	2.338320	(0.299585)	0.860458	(0.165476)	1.974788	(0.259091)
	DE	2.029736	(0.013822)	2.357182	(0.332791)	2.856477	(0.105896)	1.777750	(0.061697)
avc$_p$	GA	3.037586	(0.009410)	2.448514	(0.315756)	3.553626	(0.510720)	2.052433	(0.239424)
	RW	2.510802	(0.073231)	2.227708	(0.428315)	2.043080	(0.459468)	1.929993	(0.165972)
	DE	2.029468	(0.033474)	2.320798	(0.485104)	2.394165	(0.171982)	1.795515	(0.112706)
ncv$_p$	GA	2.540535	(0.133500)	2.567751	(0.320107)	2.328379	(0.705556)	2.034269	(0.330933)
	RW	2.539735	(0.013252)	2.124747	(0.318995)	2.442881	(0.467947)	1.979082	(0.247898)
	DE	0.341344	(0.020819)	1.447894	(0.233602)	1.194498	(0.055515)	/	(/)
repair	GA	1.241550	(0.095377)	0.669446	(0.062972)	0.630981	(0.096590)	/	(/)
	RW	0.731198	(0.150310)	0.360903	(0.029524)	0.362036	(0.033121)	/	(/)
ncv$_s$	NSGA-II	2.422386	(0.090795)	2.041589	(0.310443)	0.787285	(0.066565)	1.724724	(0.175936)
avc$_s$	NSGA-II	2.273643	(0.152831)	2.096857	(0.243741)	0.996765	(0.101500)	1.801495	(0.173736)

Table 6.2.: In this table are the median over the runs of the minimal *distance to the global optimum* (d2o) in the final population for each approach with P_2 (which are a combination out of *Initialization* (INIT), *Constraint Handling Technique* (CHT) and *Optimization Method* (OM))

7. Conclusions and Future Work

As the final chapter of this thesis the conclusion out of the experiments answering and possible further research in Future Work are presented.

7.1. Conclusions

A couple of common *Constraint Handling Technique* (CHT) from the literature are implemented, run and analyzed on two artificial large-scale problems. They all performed differently depending on the combination with an *Initialization* (INIT) and *Optimization Method* (OM). Some get close to the global optimum whereby others kept their distance or even increased it. Therefore to answer how does the chosen CHT perform with the introduced problems is not trivial. It may happen that with a higher amount of allowed function evaluations of f more approaches could reach the neighborhood of the global optimum or even find it. Therefore are more experiments required with different positions of the global optimum regarding its distance to the box-constraints and linear constraints. Also a more detailed analysis of the runs are needed. It could be helpful to find out in which scenarios which approach get stuck or is mislead and processes wrongly.
The second research question dedicates to the influence of the initialization method to the performance of an approach. It can be answered with yes. Depending on the position of the global optimum the ideal chosen INIT can give be a benefit. But with the addition that the OM and CHT take advantage of the INIT.
The last question which approach consisting of an INIT, a CHT and OM performs best and why can not be answered clearly because the approaches perform differently regarding the considered problem. The best approach for P_1 OB-DR-DE performs moderate on P_2. The best approach for P_2 OB-RC-DE performs also moderate on P_1. Since a real given problem should not be optimized by several approaches and evaluated further research considering this topic is necessary to find an approach which performs on several problems similar well.

7.2. Future Work

Since the topic of this thesis combining large-scale optimization with Constraint Handling Technique is a rather unexplored field this thesis serves as a first step towards it. Although different CHT are chosen many more are already developed. Therefore several other in the literature introduced can be applied on the mentioned problems.
Also the used CHT in this thesis can be investigated further. For example they could recombined since all CHTs are evaluated separately. Furthermore they can be tweaked to reduce the amount of function evaluations of the original f. It could be only evaluated if feasibility is guaranteed. Thus some approaches could process a couple of generations longer and consequently maybe terminate with better results. Another technique to reduce the amount of function evaluations of f is to use surrogate models. Thereby is a model trained which approximates the original f and evaluate only after a predefined amount of function evaluation on the original f again which also updates the surrogate modal.
Furthermore the geometry could be investigated in more detail. To that *abstract polytopes* may be used. With a larger knowledge about F the optimization methods could be guided to explore F more efficiently.

Bibliography

[1] Bahriye Akay and Dervis Karaboga. Artificial bee colony algorithm for large-scale problems and engineering design optimization. *Journal of Intelligent Manufacturing*, 23(4):1001–1014, August 2012. ISSN: 1572-8145.

[2] Thomas Bäck, David B Fogel, and Zbigniew Michalewicz. *Handbook of evolutionary computation*. CRC Press, 1997.

[3] Helio J. C. Barbosa, Afonso C. C. Lemonge, and Heder S. Bernardino. *A critical review of adaptive penalty techniques in evolutionary computation*. In *Evolutionary Constrained Optimization*. Rituparna Datta and Kalyanmoy Deb, editors. Springer India, New Delhi, 2015, pages 1–27.

[4] Zahra Beheshti and Siti Mariyam Shamsuddin. A review of population-based meta-heuristic algorithm. *International Journal of Advances in Soft Computing and Its Applications*, 5:1–35, March 2013.

[5] Francesco Biscani, Dario Izzo, Wenzel Jakob, Marcus Märtens, Alessio Mereta, Cord Kaldemeyer, Sergey Lyskov, Sylvain Corlay, Benjamin Pritchard, Kishan Manani, and et al. Esa/pagmo2: pagmo 2.10, January 2019. DOI: `10.5281/zenodo.2529931`.

[6] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, March 2004.

[7] Jörg Bremer and Michael Sonnenschein. Constraint-handling with support vector decoders. In *International Conference on Agents and Artificial Intelligence*, pages 228–244. Springer, 2013.

[8] Susan E Carlson and Ron Shonkwiler. Annealing a genetic algorithm over constraints. In *Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on*, volume 4, pages 3931–3936. IEEE, 1998.

[9] Adam Chehouri, Rafic Younes, Jean Perron, and Adrian Ilinca. A constraint-handling technique for genetic algorithms using a violation factor. *arXiv preprint arXiv:1610.00976*, 2016.

[10] Adam Chehouri, Rafic Younes, Jean Perron, and Adrian Ilinca. A constraint-handling technique for genetic algorithms using a violation factor. *arXiv preprint arXiv:1610.00976*, 2016.

[11] Carlos A Coello Coello. Use of a self-adaptive penalty approach for engineering optimization problems. *Computers in Industry*, 41(2):113–127, 2000.

[12] Lino Costa, Isabel Espıérito Santo, and Pedro Oliveira. An adaptive constraint handling technique for evolutionary algorithms. *Optimization*, 62(2):241–253, 2013.

[13] Rituparna Datta and Kalyanmoy Deb. *Evolutionary constrained optimization*. Springer, 2014.

[14] Kalyanmoy Deb. An efficient constraint handling method for genetic algorithms, 2000.

[15] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, T Meyarivan, and A Fast. Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[16] M. Dorigo, M. Birattari, and T. Stutzle. Ant colony optimization. *IEEE Computational Intelligence Magazine*, 1(4):28–39, November 2006. ISSN: 1556-603X.

[17] Raziyeh Farmani and Jonathan A Wright. Self-adaptive fitness formulation for constrained optimization. *IEEE Transactions on Evolutionary Computation*, 7(5):445–455, 2003.

[18] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. In *Readings in computer vision*, pages 564–584. Elsevier, 1987.

[19] Ronald Glowinski and Patrick Le Tallec. *Augmented Lagrangian and operator-splitting methods in nonlinear mechanics*, volume 9. SIAM, 1989.

[20] John H Holland. Genetic algorithms. *Scientific american*, 267(1):66–73, 1992.

[21] Abdollah Homaifar, Charlene X Qi, and Steven H Lai. Constrained optimization via genetic algorithms. *Simulation*, 62(4):242–253, 1994.

[22] Jeffrey A Joines and Christopher R Houck. On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with ga's. In *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, pages 579–584. IEEE, 1994.

[23] Jeffrey A Joines and Christopher R Houck. On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with ga's. In *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, pages 579–584. IEEE, 1994.

[24] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: open source scientific tools for Python, 2001–. URL: `http://www.scipy.org/`.

[25] A Rezaee Jordehi. A review on constraint handling strategies in particle swarm optimisation. *Neural Computing and Applications*, 26(6):1265–1275, 2015.

[26] Dervis Karaboga. An idea based on honey bee swarm for numerical optimization, technical report - tr06. *Technical Report, Erciyes University*, January 2005.

[27] Dervis Karaboga and Bahriye Basturk. Artificial bee colony (abc) optimization algorithm for solving constrained optimization problems. In *International fuzzy systems association world congress*, pages 789–798. Springer, 2007.

[28] Howard Karloff. *The simplex algorithm*. In *Linear Programming*. Birkhäuser Boston, Boston, MA, 1991, pages 23–47.

[29] J. Kennedy and R. Eberhart. Particle swarm optimization. 4:1942–1948 vol.4, November 1995. DOI: `10.1109/ICNN.1995.488968`.

[30] Dae Gyu Kim. Riemann mapping based constraint handling for evolutionary search. In *Proceedings of the 1998 ACM symposium on Applied Computing*, pages 379–385. ACM, 1998.

[31] Mustafa Kıran. *An implementation of tree-seed algorithm (tsa) for constrained optimization*. In volume 5. January 2016, pages 189–197.

[32] Slawomir Koziel and Zbigniew Michalewicz. Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization. *Evolutionary computation*, 7(1):19–44, 1999.

[33] KN Krishnanand and Debasish Ghose. Glowworm swarm based optimization algorithm for multimodal functions with collective robotics applications. *Multiagent and Grid Systems*, 2(3):209–222, 2006.

[34] Renato A Krohling and Leandro dos Santos Coelho. Coevolutionary particle swarm optimization using gaussian distribution for solving constrained optimization problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 36(6):1407–1416, 2006.

[35] C-Y Lin and W-H Wu. Self-organizing adaptive penalty strategy in constrained genetic search. *Structural and Multidisciplinary Optimization*, 26(6):417–428, 2004.

[36] Rammohan Mallipeddi, Swagatam Das, and Ponnuthurai Nagaratnam Suganthan. Ensemble of constraint handling techniques for single objective constrained optimization. In *Evolutionary Constrained Optimization*, pages 231–248. Springer, 2015.

[37] Efrén Mezura-Montes and Carlos A. Coello Coello. *Constrained optimization via multiobjective evolutionary algorithms*. In *Multiobjective Problem Solving from Nature: From Concepts to Applications*. Joshua Knowles, David Corne, Kalyanmoy Deb, and Deva Raj Chair, editors. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pages 53–75.

[38] Efrén Mezura-Montes and Carlos A Coello Coello. Constraint-handling in nature-inspired numerical optimization: past, present and future. *Swarm and Evolutionary Computation*, 1(4):173–194, 2011.

[39] Efrén Mezura-Montes and Jorge Isacc Flores-Mendoza. Improved particle swarm optimization in constrained numerical search spaces. In *Nature-inspired algorithms for optimisation*, pages 299–332. Springer, 2009.

[40] Zbigniew Michalewicz and Marc Schoenauer. Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary computation*, 4(1):1–32, 1996.

[41] Melanie Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, USA, 1998. ISBN: 0262631857.

[42] Marco Montemurro, Angela Vincenti, and Paolo Vannucci. The automatic dynamic penalisation method (adp) for handling constraints with genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, 256:70–87, 2013.

[43] Angel E Muñoz Zavala, Arturo Hernández Aguirre, and Enrique R Villa Diharce. Constrained optimization via particle evolutionary swarm optimization algorithm (peso). In *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, pages 209–216. ACM, 2005.

[44] Pruettha Nanakorn and Konlakarn Meesomklin. An adaptive penalty function in genetic algorithms for structural design optimization. *Computers & Structures*, 79(29-30):2527–2539, 2001.

[45] Travis Oliphant. NumPy: a guide to NumPy. USA: Trelgol Publishing, 2006–. URL: `http://www.numpy.org/`.

[46] FAO Otieno, JA Adeyemo, HA Abbass, R Sarker, I Amir, FM Fisher, A Rakesh, BV Babu, BV Babu, MM Jehan, et al. Differential evolution: a simple and efficient adaptive scheme for global optimization over continuous spaces. *Trends in Applied Sciences Research*, 5(1):531–552, 2002.

[47] Nikhil Padhye, Pulkit Mittal, and Kalyanmoy Deb. Feasibility preserving constraint-handling strategies for real parameter evolutionary optimization. *Computational Optimization and Applications*, 62(3):851–890, 2015.

[48] Konstantinos E Parsopoulos and Michael N Vrahatis. Unified particle swarm optimization for solving constrained engineering optimization problems, Springer, 2005.

[49] Riccardo Poli, James Kennedy, and Tim Blackwell. Particle swarm optimization. *Swarm Intelligence*, 1(1):33–57, June 2007.

[50] Daniel J Poole, Christian B Allen, and Thomas CS Rendall. A generic framework for handling constraints with agent-based optimization algorithms and application to aerodynamic design. *Optimization and Engineering*, 18(3):659–691, 2017.

[51] Bingqin Qiao. Hybrid particle swarm algorithm for solving nonlinear constraint optimization problems. In 2012.

[52] Singiresu S Rao. *Engineering optimization: theory and practice*. John Wiley & Sons, 2009.

[53] Thomas P. Runarsson and Xin Yao. Stochastic ranking for constrained evolutionary optimization. *IEEE Transactions on evolutionary computation*, 4(3):284–294, 2000.

[54] Nazmul Siddique and Hojjat Adeli. Nature inspired computing: an overview and some future directions. *Cognitive Computation*, 7(6):706–714, December 2015.

[55] Biruk Tessema and Gary G Yen. An adaptive penalty formulation for constrained evolutionary optimization. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 39(3):565–578, 2009.

[56] Shigeyoshi Tsutsui, Masayuki Yamamura, and Takahide Higuchi. Multi-parent recombination with simplex crossover in real coded genetic algorithms. In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 1*, pages 657–664. Morgan Kaufmann Publishers Inc., 1999.

[57] Yong Wang, Zixing Cai, Yuren Zhou, and Zhun Fan. Constrained optimization based on hybrid evolutionary algorithm and adaptive constraint-handling technique. *Structural and Multidisciplinary Optimization*, 37(4):395–413, 2009.

[58] Xin-She Yang. Firefly algorithm, stochastic test functions and design optimisation. *International Journal of Bio-inspired Computation*, 2, March 2010. DOI: 10.1504/IJBIC.2010.032124.

[59] Yinyu Ye. *Interior Point Algorithms: Theory and Analysis*. John Wiley & Sons, Inc., New York, NY, USA, 1997.

[60] Heiner Zille, Hisao Ishibuchi, Sanaz Mostaghim, and Yusuke Nojima. A framework for large-scale multiobjective optimization based on problem transformation. *IEEE Transactions on Evolutionary Computation*, 22(2):260–275, 2018.

Declaration of Authorship

I hereby declare that this thesis was created by me and me alone using only the stated sources and tools.

Andreas Petrow   Magdeburg, March 22, 2019

A. Appendix

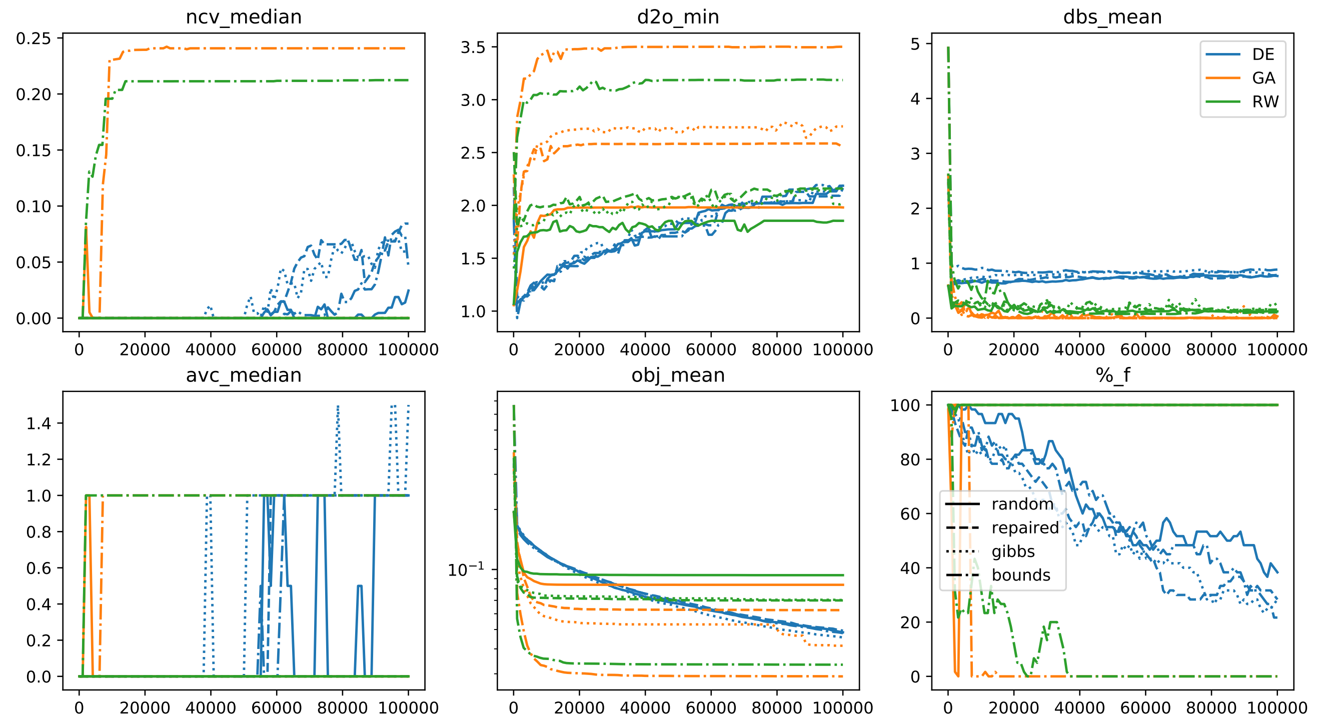


Figure A.1.: Feasibility Preserving *feasibility preserving CHT-Decoder* (DC) applied to P_1; coloring of the lines implies the used OM and the linestyle the INIT