Universidad Veracruzana

Instituto de Investigaciones en Inteligencia Artificial

Doctoral Thesis

A Study on Metaheuristic Approaches for Single- and Multi-Objective Bilevel Optimization

Jesús-Adolfo Mejía-de-Dios

supervised by

PhD. Efrén Mezura-Montes

October, 2022

Acknowledgments

First of all, I would like to thank my advisor Dr. Efrén Mezura-Montes for his personal and professional support, without his guidance, it would not have been possible to complete my research works.

I would like to express my gratitude to Rocío S.G. for giving me the courage to accomplish academic activities that I would not dare without her support. Thank you for understanding me without asking, for supporting me without asking, and for loving me as I love you.

I would like to thank my family for their constant moral support in these difficult times.

I want to thank the reviewers of this document, who kindly reviewed and suggested changes that helped improve this work.

I want to thank all my professors who helped me achieve this work.

This research work has been possible thanks to the support from CONACyT through a scholarship to pursue my Ph.D. studies.

Abstract

Bilevel Optimization (BO) is an emergent research area due to its important property to model instances where two optimization problems are nested in a hierarchical structure. That is, a BO problem is an optimization problem with another optimization problem as a constraint. The problem used as the constraint is known as the lower-level problem, whilst the main optimization task is at the upper level. Moreover, the goal of the BO problem is governed by the upper-level problem but considers the lower-level optimality as part of the constraints.

In BO, a feasible solution is obtained by optimal solutions at the lower-level considering also the corresponding equality and inequality constraints. Thus, a BO problem is solved when the upper-level finds an optimal and feasible solution for the upper level.

Different variants of BO problems emerge when the number of conflicting objective functions are considered at each level. For instance, if the upper or lower contains a multi-objective problem, then a multi-objective bilevel optimization task is obtained. On the other hand, a single-objective bilevel task is defined if two single-objective problems are nested, being this last one of the most studied so far.

Although different solution approaches have been proposed from classical to metaheuristic methods, BO still requires studies to improve the current state-of-the-art. Those broadly include theoretical studies and efficient solution approaches. This thesis is used to address the following three studies, (1) proposing a bilevel metaheuristic able to solve optimization problems (considering accuracy, feasibility, and efficiency) at the upper level, (2) a study and solution proposal for the automated parameter tuning problem as the bilevel task to show how convenient becomes the application of bilevel models, and (3) the application of a novel representation of solution for multi-objective bilevel optimization. The studies are evaluated through multiple experiments by performing comparative studies against state-of-the-art methods and using statistical tests to discuss the results.

Contents

List of Figures

List of Tables

List of abbreviations

Abbreviation	Definition
BCA	Bilevel Centers Algorithm
BLEAQ	Bilevel Evolutionary Algorithm with Quadratic Approximations
BLEMO	Bilevel Evolutionary Multi-objective Optimization
BO	Bilevel Optimization
BOP	Bilevel Optimization Problem
EA	Evolutionary Algorithm
EC	Evolutionary Computing
ECA	Evolutionary Centers Algorithm
EQPSO	Elite Quantum Behaved Particle Swarm Optimization
H-BLEMO	Hybrid BLEMO
MOBO	Multi-objective Bilevel Optimization
MOEA	Multi-objective Evolutionary Algorithm
mf-BLEAQ	BLEAQ based on multiple quadratic fibers
NSGA-II	Non-domianted Sorting Genetic Algorithm
PMM	Problem with Pseudo feasible solutions
QBCA	Quasi-Newton Bilevel Centers Algorithm
SMD	Test problem proposed by Sinha, Malo and Deb.
SPEA2	Improved Strength Pareto Evolutionary Algorithm
SMS-EMOA	Multiobjective selection based on dominated hypervolume
SMS-MOBO	Nested SMS-EMOA algorithm for multi-objective optimization
UL/LL	Upper Level/Lower Level

Chapter 1

Introduction

Researchers from computer science have become interested in a novel category of hierarchical optimization issues in recent years. This topic was first stated by Von Stackelberg in 1934 and is now known as the Bilevel Optimization problem [132].

The upper and lower level problems in a Bilevel Optimization (BO) problem are nested optimization problems. As a result, BO has a hierarchical structure, with a leader (upper-level authority) and a follower (lower level authority) playing a sequential game, in which the leader makes an optimal decision based on the best decisions that the follower can make, and such decisions can cause conflicts between both levels. In this approach, there is an optimal solution at the lower level for every solution (decision vector) at the higher level for a BO issue. Bilevel problems can be unconstrained, constrained, single-objective, multi-objective, continuous or discrete, and so on, but they always have a nested optimization problem as a constraint [13].

In general, the computing cost of solving a BOP is NP-hard [55]. [38]. However, problem transformation is not always viable when the objective functions or constraints are not mathematically well-behaved. For those cases, metaheuristic approaches can be used to approximate suitable solutions.

Designing metaheuristic algorithms to solve bilevel problems has been a challenging issue, due to computational effort required should be as reduced as possible. Moreover, it should be noted that the formalization of evolutionary bilevel optimization is relatively new, making it a fertile area to do research where offering effective solutions for real problems is necessary.

This doctoral proposal focuses on the general case of bilevel optimization, where a leader and a follower can give one or multiple conflicting objectives. Firstly, proposing a metaheuristic to address single-objective bilevel optimization problems, including the application in automated parameter tuning. Finally, a novel representation of solutions is given in the context of multi-objective bilevel problems.

1.1 Problem Statement

Bilevel Optimization has become an important and challenging issue in Evolutionary Computation because different real-world problems define two nested optimization models that can be single- or multi-objective problems. Furthermore, finding optimal feasible solutions to bilevel problems require high computational resources.

Let us consider the principal-agent problem, a real-world problem used here to exemplify how a bilevel optimization problem works. A principal subcontracts a job to an agent and uses an incentive scheme that aligns the agent's interests with the principal. In contrast, the agent prefers to act in his interests rather than the principal. In this example, the upper-level authority (principal) is looking for the best decisions subject to the lower-level authority (agent) maximizing her/his advantage, taking into account the incentive scheme provided by the principal.

In recent years different metaheuristic algorithms (including evolutionary algorithms) have been extended to handle those hierarchical problems. However, some important scenarios need to be addressed to improve the quality of outcomes from bilevel metaheuristics. A study of bilevel metaheuristics in single and multi-objective bilevel problems is required to improve the current state-of-the-art.

1.2 Research Hypothesis

1.2.1 Hypothesis #1

A metaheuristic inspired by the center of mass concept as a variation operator (referred to as BCA) can solve bilevel optimization problems. The algorithm performance will be quantified in two ways, (1) considering the number of function evaluations at the upper level (a lower value with respect to *state-of-the-art* algorithms is expected), and the precision (either similar or even better value is expected) concerning both levels.

1.2.2 Hypothesis #2

It can be possible to adapt mechanisms to BCA, in order to handle the pseudo-feasible solutions from bilevel optimization problems that are affecting the performance of Evolutionary Algorithms.

1.2.3 Hypothesis #3

A Multi-Objective Bilevel Optimization (MOBO) algorithm with a new representation of solutions based on a family concept can approximate optimal feasible solutions with competitive hypervolume values regarding a *state-of-the-art* algorithm.

1.3 General Objective

The main goal of this research is to perform a study on Bilevel Optimization using the metaheuristics (BCA and MOBO algorithm) stated in the research hypotheses.

1.4 Specific Objectives

In order to develop this research proposal, the following specific objectives have been considered:

1. To study some properties of bilevel optimization problems to identify the most important.
2. To implement a suitable metaheuristic for single-objective bilevel optimization problems such that is, at least, as accurate as other algorithms but with less problem evaluations computed.
3. To test the efficiency of BCA in benchmark functions as well as in bilevel problem from the automated parameter tuning.
4. To use a family concept to propose a multi-objective bilevel optimization approach.
5. To perform the corresponding statistical test analyses to evaluate the above mentioned hypotheses.

1.5 Contributions

The contributions of this thesis are given below:

- An updated literature review considering important concepts and solution approaches for single- and multi-objective bilevel optimization.

- A nested approach based on the center of mass concept for solving single-objective bilevel optimization problems.
- Mechanisms to mitigate the impact of misleading solutions in bilevel problems that affect the performance of current evolutionary approaches.
- A bilevel model and a solution approach for the automated parameter tuning problem to configure metaheuristics with a reduced computational cost.
- A novel solution representation based on a family concept to solve multi-objective bilevel optimization instances.

1.6 Thesis Structure

This thesis is structured in several chapters including theoretical background, literature review, and contributions. Figure 1.1 summarizes the main parts of this work. Each chapter is described in the following list:

- Chapter 2 introduces the bilevel optimization problems. It also presents a warning sign for evolutionary algorithms in bilevel optimization problems related to misleading results that EAs report. Some examples are given to illustrate this issue. Finally, some theoretical results are detailed to figure out if recent EAs might report misleading solutions.
- Chapter 3 presents a literature review of BO solution approaches, from classical to metaheuristic algorithms.
- Chapter 4 presents the proposal of a physics-inspired algorithm based on the center of mass concept (referred to as BCA) to deal with bilevel optimization problems. A nested approach is considered in this chapter.
- Chapter 5 describes the improved version of the nested algorithm introduced in Chapter 4 by implementing mechanisms to handle pseudo-feasible solutions, which is useful to prevent reporting misleading results.

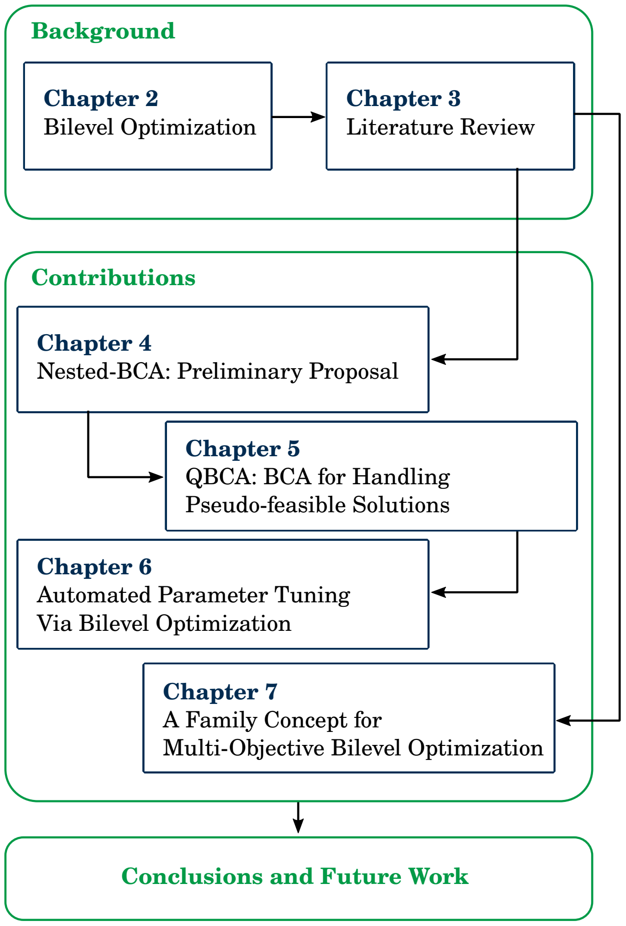

Background
Chapter 2
Bilevel Optimization
Chapter 3
Literature Review
Contributions
Chapter 4
Nested-BCA: Preliminary Proposal
Chapter 5
QBCA: BCA for Handling Pseudo-feasible Solutions
Chapter 6
Automated Parameter Tuning Via Bilevel Optimization
Chapter 7
A Family Concept for Multi-Objective Bilevel Optimization
Conclusions and Future Work

Figure 1.1: The rest of the thesis is structured into two main parts: Background and Contributions with their respective chapters. Finally, the conclusions and future work are given.

- Chapter 6 addresses the automated parameter tuning problem via bilevel optimization by using the BCA framework with surrogate models to handle the high computational cost related to the automated parameter tuning problem.
- Chapter 7 presents evolutionary mechanisms based on the family concept to deal with multi-objective bilevel optimization problems.
- Finally, the conclusions and future work are given.

Moreover, three Appendices A-C, are given at the end of the document as supplementary material. The first one includes theoretical background on optimization, the second one contains mathematical proofs of theorems stated in this work, and the last one contains some benchmark problems used in this thesis.

1.7 Publications

This section contains the research products derived from this work. The publications are divided in three parts, (1) journal papers, (2) conference papers, and (3) book chapters.

Journal Papers

The list of journal publications is given below. Note that the corresponding Impact Factor (IF) is also specified.

1. Jesus-Adolfo Mejía-de-Dios, Efrén Mezura-Montes and Marcela Quiroz-Castellanos, **Automated Parameter Tuning as a Bilevel Optimization Problem Solved by a Surrogate-Assisted Population-Based Approach**, Applied Intelligence, 2021. `https://doi.org/10.1007/s10489-020-02151-y`. **Q2, IF 5.019**. *Related to Chapter 6*.
2. Jesús-Adolfo Mejía-de-Dios, Efrén Mezura-Montes, and Porfirio Toledo-Hernández, **Pseudo-feasible Solutions in Evolutionary Bilevel Optimization: Test Problems and Performance Assessment**, Applied Mathematics and Computation, Volume 412, 2021. `https://doi.org/10.1016/j.amc.2021.126577`. **Q1, IF 4.397**. *Related to Chapter 5*.
3. Jesus-Adolfo Mejía-de-Dios, Alejandro Rodríguez-Molina, and Efrén Mezura-Montes, **Multi-objective Bilevel Optimization: A Survey of the State-of-the-Art**, IEEE Transactions on Systems, Man, and Cybernetics - Systems (second round of review). **Q1. IF 11.471**. *Related to Chapter 7*.
4. Jesus-Adolfo Mejía-de-Dios and Efrén Mezura-Montes, **Metaheuristics: A Julia Package for Single- and Multi-Objective Optimization**, The Journal of Open Source Software (submitted). ISSN 2475-9066.

5. Alejandro Rodríguez-Molina, Jesus-Adolfo Mejía-de-Dios, and Efrén Mezura-Montes, **Evolutionary Semi-Vectorial Bilevel Optimization in the Mechanical and Control Design of Systems**, IEEE Transactions on Cybernetics (submitted). **Q1. IF 19.118**.

Conference Papers

1. Jesús-Adolfo Mejía-de-Dios and Efrén Mezura-Montes. A physics-inspired algorithm for bilevel optimization. In **Power, Electronics and Computing (ROPEC), 2018 IEEE International Autumn Meeting on**, pages 1-6. IEEE, 2018. `https://doi.org/10.1109/ROPEC.2018.8661368`. *Related to Chapter 4*.

2. Jesús-Adolfo Mejía-de-Dios and Efrén Mezura-Montes, **A Metaheuristic for Bilevel Optimization Using Tykhonov Regularization and the Quasi-Newton Method**, in Proceedings of the IEEE Congress on Evolutionary Computation, IEEE Press, Weillington, New Zealand, 2019. `https://doi.org/10.1109/CEC.2019.8790097`. *Related to Chapter 4*.

3. Jesús-Adolfo Mejía-de-Dios and Efrén Mezura-Montes, **A Surrogate-Assisted Metaheuristic for Bilevel Optimization**, in Proceedings of the ACM Genetic and Evolutionary Computation Conference (GECCO), Cancun, Mexico, ACM Press, 2020. `https://doi.org/10.1145/3377930.3390236`. *Related to Chapter 6*.

4. Jesús-Adolfo Mejía-de-Dios and Efrén Mezura-Montes, **Generating multi-objective bilevel optimization problems with multiple non-cooperative followers**, in Companion Proceedings of the ACM Genetic and Evolutionary Computation Conference (GECCO), pages 187-188, Lille, France, ACM Press, 2021. `https://doi.org/10.1145/3449726.3459495`. *Related to Chapter 7*.

5. Jesús-Adolfo Mejía-de-Dios, Alejandro Rodríguez-Molina, and Efrén Mezura-Montes, **A Novel Evolutionary Framework Based on a Family Concept**

for Solving Multi-objective Bilevel Optimization Problems, in Companion Proceedings of the ACM Genetic and Evolutionary Computation Conference (GECCO), Boston, USA, ACM Press, 2022. `https://doi.org/10.1145/3520304.3529045`. *Related to Chapter 7*.

Book Chapters

1. Sebastián-José de-la-Cruz-Martínez, Jesús-Adolfo Mejía-de-Dios, and Efrén Mezura-Montes, **Efficient Archiving Method for Handling Preferences in Constrained Multiobjective Evolutionary Optimization**, Handbook on Decision Making-Volume 3: Trends and Challenges in Intelligent Decision Support Systems, Springer-Verlag, 2022. `https://doi.org/10.1007/978-3-031-08246-7_5`.

Chapter 2

Bilevel Optimization

In recent years, researchers have become interested in a new type of optimization problem. Von Stackelberg first proposed this problem in 1934, and it is now known as the Bilevel Optimization (BO) problem [132]. Bilevel problems can be unconstrained, constrained, single or multi-objective, continuous or discrete, and so on, but they always have a nested optimization problem as a constraint [13, 38]. This newly suggested hierarchical structure can be used to represent decision-making processes in which a leader (higher-level authority) maximizes its aims by limiting itself to the best decisions/solutions provided by a follower (lower-level authority) [23, 131, 133].

The following section defines and describes a general BO problem with single-objective functions at both levels.

2.1 Problem Definition

Bilevel optimization is an optimization task that contains an optimization problem in the constraints that should be solved to approximate a feasible solution. The main optimization task is the Upper-Level (UL) problem, while the nested optimization problem is the Lower-Level (LL) problem. The following gives the main definition of a BO problem [13, 38]:

Definition 2.1: Bilevel Optimization Problem

The 5-tuple $(F,\ f,\ X,\ Y,\ \mathbb{R})$ is called a bilevel optimization problem where F is the upper-level/leader's objective function and the lower-level/follower's objective function is given by f. Then, the hierarchical optimization process is formulated as follows:

Minimize

$$F(x,\ y),\ x \in X \tag{2.1}$$

subject to:

$$y \in \arg\min_{y \in Y}\{f(x,y)\ :\ g_j(x,y) \leq 0,\ j = 1,\dots,J\} \tag{2.2}$$

$$G_l(x,y) \leq 0,\ l = 1,\dots,L. \tag{2.3}$$

where G_l and g_j are the inequality constraints for the upper and lower levels, respectively.

Single-objective bilevel optimization problems work on two functions $F : X \times Y \to \mathbb{R}$ and $f : X \times Y \to \mathbb{R}$, known as the upper-level objective function (leader) and lower-level objective function (follower), respectively. In this work, $X \subseteq \mathbb{R}^n$ and $Y \subseteq \mathbb{R}^m$ are considered. Figure 2.1 shows a schematic diagram of a BO problem and describes what occurs when two decision vectors are taken at the upper level.

Hansen et al. investigated if BO problems are (strongly) NP-hard since as-

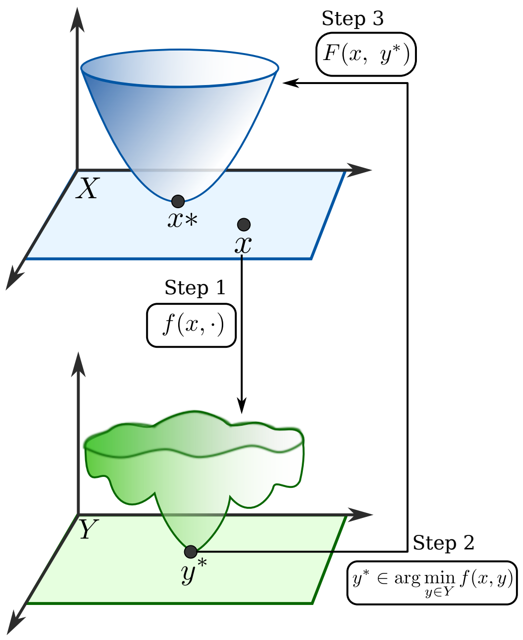


Figure 2.1: Diagram of a bilevel optimization problem and how a feasible solution is evaluated, i.e., the leader takes a decision in Step 1, then the follower takes her/his best decision based on her/his objectives but considering the leader policy, finally the leader evaluates her/his decision considering the response from follower. Here, y^* is defined as in Equation (2.2). Note that $(x,\ y^*)$ is a feasible solution.

sessing a solution in most basic BO problems (unimodal linear programming at both levels) is likewise NP-hard [55, 130]. Furthermore, many real-world issues may be easily represented as BO problems [116], for example: taxation, border security problems, transportation problems, machine learning algorithms tuning, among others [9, 13, 116].

It is worth mentioning that, for a UL decision vector $\boldsymbol{x}$, the LL can be rewritten as set-valued mapping:

$$\Psi(\boldsymbol{x}) = \underset{\boldsymbol{y}\in Y}{\text{argmin}} \left\{ f(\boldsymbol{x},\boldsymbol{y}) : g_j(\boldsymbol{x},\boldsymbol{y}) \le 0,\ j = 1, 2, \dots, J \right\}. \tag{2.4}$$

The set-valued mapping $\Psi(\boldsymbol{x})$ is commonly used to simplify notation and states important properties on certain BO problems. For instance, the bilevel problem given in Definition 2.1 can be rewritten as:

$$\min_{(\boldsymbol{x},\boldsymbol{y})\in X\times Y} \{F(\boldsymbol{x},\boldsymbol{y}) : \boldsymbol{y} \in \Psi(\boldsymbol{x}), G_l(\boldsymbol{x},\boldsymbol{y}) \le 0, l = 1, 2, \dots, L\}. \tag{2.5}$$

It is easy to determine that both problems are equivalent. The following section describes an important position that comes out due to the interaction between the leader and its follower.

Example 2.1

This example is used to illustrate the main properties of a bilevel problem.

Minimize:

$$F(x,y) = x^2 + y^2,\ x \in X$$

subject to:

$$y \in \operatorname*{argmin}_{y \in Y} \left\{ f(x,y) = (x-y)^2 \right\},$$

where $X = Y = \mathbb{R}$.

To solve this problem, assume that x is given by the leader, then the lower-level minimum for $f(x,y) = (x-y)^2$ is zero, which is obtained when $y = x$. Thus, $\Psi(x) = \{x\}$, implies that

$$\min_{(x,y) \in X \times Y} \{F(x,y) : y \in \Psi(x)\} = \min_{(x,y) \in X \times Y} \{F(x,y) = 2x^2\}.$$

Therefore, the optimal solution for this BO problem is obtained when $x = y = 0$.

2.2 Optimistic and Pessimistic Positions

The interaction between leader and follower is an important aspect to consider in bilevel models because the follower's decisions may affect the upper level objective values.

Here, the reformulation (2.4)-(2.5) is used to describe the optimistic and pessimistic positions. As mentioned above, these positions emerge since the leader can expect followers' decisions that benefit, or not, its objectives.

2.2.1 Optimistic Position

Here, the follower gives the leader those LL solutions that benefit the UL problem, i.e, if the lower level contains multiple optimal solutions, it brings the leader those solutions that minimize the UL objective function [104, 2].

Summarizing, if you assume that the LL problem contains multiple optimal solutions, i.e., $|\Psi(x)| \geq 2$, then an optimistic bilevel problem is on finding $\boldsymbol{y} \in \text{argmin}_{\boldsymbol{y} \in \Psi(\boldsymbol{x})}\{F(\boldsymbol{x}, \boldsymbol{y})\}$ for a given $\boldsymbol{x}$. It is worth noticing that most works assess the optimistic position because several situations reflect it, but also because such position does not introduce other requirements.

2.2.2 Pessimistic Position

Unlike the optimistic case, a pessimistic position occurs when the lower level has multiple optimal solutions, i.e., $|\Psi(x)| \geq 2$, and the leader assumes the worst case, i.e., if the leader is minimizing its objective function, then for a given upper level decision $\boldsymbol{x}$, it is expecting those LL optimal solutions that leads to the worst objective function values: $\boldsymbol{y} \in \text{argmax}_{\boldsymbol{y} \in \Psi(x)}\{F(\boldsymbol{x}, \boldsymbol{y})\}$.

It is important to note that the pessimistic position is, in general, less tractable than the optimistic one, since it is a problem that in several cases is not well-behaved because it introduces non-tractable functions during reformulations [83].

2.3 Feasibility

The feasibility and optimality notions in bilevel optimization are different concerning single-level problems, since the leader requires that the follower makes optimal decisions whilst the upper level is optimized. Part of the following results and definitions are detailed in [38].

Definition 2.2: Feasible Solution

A feasible solution for the bilevel optimization problem (Definition 2.1) is obtained when $\boldsymbol{x} \in X$, $\boldsymbol{y} \in \Psi(\boldsymbol{x})$, and $G_l(\boldsymbol{x}, \boldsymbol{y}) \leq 0$ for $l = 1, 2, \ldots, L$.

Note that a solution $(\boldsymbol{x}, \boldsymbol{y}) \in X \times Y$ satisfying only equality and inequality constraints at each level (g_j, G_l) is not necessarily a feasible solution because the lower-level solution $\boldsymbol{y}$ has to be an optimal solution.

Definition 2.3: Optimal Feasible Solution

The point $(\boldsymbol{x}^*, \boldsymbol{y}^*) \in X \times Y$ is an optimal solution if $\boldsymbol{y}^* \in \Psi(\boldsymbol{x}^*)$ and $F(\boldsymbol{x}, \boldsymbol{y}) > F(\boldsymbol{x}^*, \boldsymbol{y}^*)$ for all $(\boldsymbol{x}, \boldsymbol{y})$ satisfying $\boldsymbol{y} \in \Psi(\boldsymbol{x})$.

Note that when you assume that $\Psi(\boldsymbol{x})$ is a singular set (only contains one element) for all $\boldsymbol{x}$, then the unconstrained bilevel problem can be simplified as $\min_{\boldsymbol{x} \in X} F(\boldsymbol{x}, \boldsymbol{y}(\boldsymbol{x}))$ where $\boldsymbol{y}(\boldsymbol{x}) \in \Psi(\boldsymbol{x})$.

2.4 Optimality Conditions

This section gives two main results on optimality results, one to prove the existence of solutions for a class of bilevel problems and the second to state optimality conditions based on the Karush-Kuhn-Tucker conditions.

Note that Mangasarian-Fromovitz constraint qualification (MFCQ) states that the gradients of the equality constraints are linearly independent at y^* and there exists a vector $d \in \mathbb{R}^n$, such that $\nabla g_j(\boldsymbol{x}, \boldsymbol{y}^*)^\top d < 0$ for all active inequality constraints and $\nabla h_j(\boldsymbol{x}, \boldsymbol{y}^*)^\top d = 0$ for all equality constraints.

Theorem 2.1: Existence

Assume that the problem in Definition 2.1 has a convex lower-level problem *that satisfies MFCQ and has a unique optimal feasible solution for each $\boldsymbol{x} \in X$. Then the bilevel programming problem has a global optimal solution which is also feasible.*

The proof for this theorem is detailed in [38].

Theorem 2.1 proves the existence of optimal feasible solutions, particularly for some bilevel programming problems. This is helpful since once the existence is proved, then determining optimality conditions makes sense.

Theorem 2.2: Optimality Condition

Assume that the lower-level optimization corresponds to a convex optimization problem for all $\boldsymbol{x} \in X$ for which the Mangasarian-Fromovitz constraint qualification is satisfied at all points $(\boldsymbol{x}, \boldsymbol{y}) \in X \times Y$ with $G(\boldsymbol{x}, \boldsymbol{y}) \leq 0$, $\boldsymbol{y} \in \Psi(\boldsymbol{y})$, then the optimal solution from problem in Definition 2.1 is a feasible solution for the following optimization problem:

$$\begin{aligned} \min_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{\lambda},\boldsymbol{\mu}} \; & F(\boldsymbol{x}, \boldsymbol{y}) \\ G(\boldsymbol{x}, \boldsymbol{y}) &\leq 0 \\ \nabla_{\boldsymbol{y}} L(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\lambda}, \boldsymbol{\mu}) &= 0 \\ \min\{-g(\boldsymbol{x}, \boldsymbol{y}), \boldsymbol{\lambda}\} &= 0 \\ h(\boldsymbol{x}, \boldsymbol{y}) &= 0, \end{aligned}$$

where $L(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\lambda}) = f(\boldsymbol{x},\ \boldsymbol{y}) + \sum_{j=1}^{J} \lambda_j g_j(\boldsymbol{x},\ \boldsymbol{y}) + \sum_{l=1}^{L} \mu h_j(\boldsymbol{x},\ \boldsymbol{y})$.

The proof of this theorem is detailed in [38].

Theorem 2.2 concludes that optimal feasible solutions from a bilevel optimization problem are equivalent to a single-level optimization problem. At the beginning, a single-level reduction can be suggested. However, the opposite direction for Theorem 2.2 can be satisfied when the lower level is regular convex, unimodal and strongly stable for all $\boldsymbol{x} \in X$ (see [38] for details).

2.5 Pseudo-Feasibility in Evolutionary Computing for BO

Based on the fact that EAs are very competitive to solve complex optimization problems, their comparison process plays an important role to assess their performance and behavior. Comparing individuals within an EA is one of the most significant components since it can bias the search towards promising regions [11]. Therefore, EAs must have an appropriate selection criteria for determining when one solution is better than another. We provide a solution analysis as a result of the foregoing in the context of bilevel optimization. After that, a discussion about how EAs for BO should deal with problems as well.

Most of the concepts in this section were extracted from [33]. The discussion is initialized by defining concepts on the objective space generated by objective function values.

Definition 2.4: Bilevel Value

Assume an unconstrained bilevel optimization problem, then the point denoted as $(f(\boldsymbol{x}, \boldsymbol{y}),\ F(\boldsymbol{x}, \boldsymbol{y})) \in \mathbb{R}^2$ will be called a **bilevel value** for some $(\boldsymbol{x},\ \boldsymbol{y}) \in X \times Y$.

The collection of all bilevel values of a given bilevel optimization problem is referred to as the bilevel space. A feasible solution can be used to obtain a feasible bilevel value, which is described as follows.

Definition 2.5: Feasible Value

A **feasible value** is a bilevel value generated via a feasible solution and is given by the following expression $(f(\boldsymbol{x}, \boldsymbol{y}^*),\ F(\boldsymbol{x}, \boldsymbol{y}^*)) \in \mathbb{R}^2$ for each $\boldsymbol{y}^* \in \Psi(\boldsymbol{x})$.

As part of our research, in [33] the following bilevel optimization problem is used to illustrate the above concepts.

Example 2.2

Minimize

$$F(x,y) = \sin^2\left(x - y^2\right), \tag{2.6}$$

subject to

$$y \in \Psi(x) = \arg\min_{y\in Y}\left\{f(x,\ y) = y^2 - x\right\}, \tag{2.7}$$

where $X = Y = [-a,\ a]$ with $a = 1$.

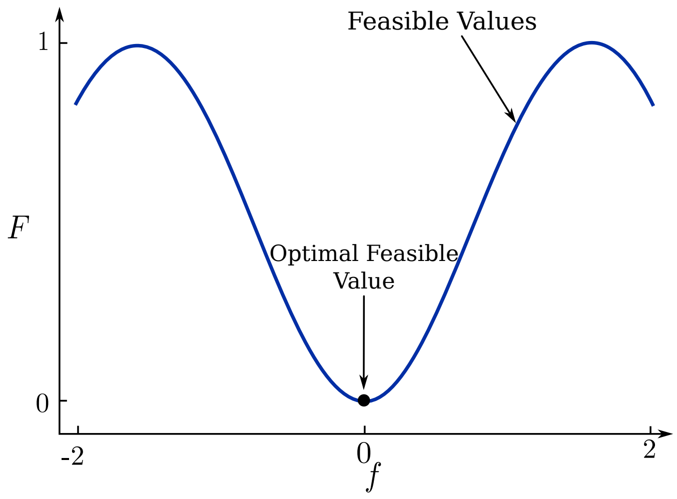


Figure 2.2: Representing the feasible values in a bilevel space from the problem stated in Equations (2.6)-(2.7) with $X = Y = [-1, 1]$.

Figure 2.2 shows the bilevel values distribution when feasible solutions are evaluated for the bilevel problem in Example 2.2. This example is used to illustrate that there exists infeasible solutions in objective space that are equal to optimal feasible solutions in some bilevel problems. That is, in Example 2.2, the set-mapping is $\Psi(x) = \{0\}$ for all $x \in X$. Also, if $y^* \in \Psi(x)$, a feasible solution is obtained. Thus, for $y^* \in \Psi(x)$ with $x \in X$ fixed, then $F(x,\ y^*) = \sin^2(x)$ and $f(x,\ y^*) = -x$. Therefore, an optimal feasible solution is $(x^*, y^*) = (0, 0)$, then $f(x^*, y^*) = 0$ and $F(x^*, y^*) = 0$.

Moreover, in our same work in [33] we studied the case when $f(x, y) = 0$ and $F(x, y) = 0$ even on infeasible solutions (x, y). This infeasible solution is a warning sign that bilevel values are not necessarily related to feasibility. The following theorem and corollary state mathematical results on “pseudo-feasibility”.

Theorem 2.3

Let $(F,\ f,\ X,\ Y,\ \mathbb{R})$ be a BO problem such that $f(\boldsymbol{x},\cdot)$ is non-injective for the unique LL global solution $\boldsymbol{y}^* \in \Psi(\boldsymbol{x})$, with $\boldsymbol{x} \in X$ fixed and $F(\boldsymbol{x},\boldsymbol{y}) = Q(f(\boldsymbol{x},\boldsymbol{y}))$ where $Q : Img(f) \to \mathbb{R}$ and $Img(f)$ is the image of f. Then, there exists $(\boldsymbol{w},\boldsymbol{z}) \in X \times Y$ such that $\boldsymbol{z} \notin \Psi(\boldsymbol{w})$, $f(\boldsymbol{x},\boldsymbol{y}^*) = f(\boldsymbol{w},\boldsymbol{z})$ and $F(\boldsymbol{x},\boldsymbol{y}^*) = F(\boldsymbol{w},\boldsymbol{z})$.

The proof for this result is detailed in Proof B.1 in Appendix B.

Note that Theorem 2.3 establishes the properties of a bilevel optimization problem that can contain a pseudo-feasible solution, i.e., proof the existence of pseudo-feasibility in some problems. Besides, Theorem 2.3 sets the theoretical groundwork for more sophisticated and interesting BO challenges, i.e., an upper-level function F can be defined by adding different compositions of non-injective lower-level functions. This is based on the notion that an injective function's translation is also injective.

Corollary 2.1

There exists a BO problem $(F, f, X, Y, \mathbb{R})$ such that $f(\boldsymbol{x}^*,\ \boldsymbol{y}^*) = f(\boldsymbol{x},\ \boldsymbol{y})$ and $F(\boldsymbol{x}^*,\ \boldsymbol{y}^*) = F(\boldsymbol{x},\ \boldsymbol{y})$ where $(\boldsymbol{x}^*,\ \boldsymbol{y}^*)$ is an optimal feasible solution while $(\boldsymbol{x},\ \boldsymbol{y})$ is infeasible.

The proof for this result is detailed in Proof B.2 in Appendix B.

The result given by Corollary 2.1 states that there are bilevel optimization problems such that the objective function values from each level (UL and LL) on a feasible solution can be related to an infeasible solution but with the same objective function values at both levels.

Based on those previous formal definitions and the examples shown, a pseudo-feasible solution is defined as follows.

Definition 2.6: Pseudo-feasible Solution

A solution $(\boldsymbol{w}, \boldsymbol{z})$ satisfying $(f(\boldsymbol{w}, \boldsymbol{z}),\ F(\boldsymbol{w}, \boldsymbol{z}))$ is a bilevel value and is called **pseudo-feasible**, i.e., a pseudo-feasible solution takes the same UL and LL objective function values that a feasible one. Also, we will say that pseudo-feasible solution $(\boldsymbol{w}, \boldsymbol{z})$ is related to the feasible one $(\boldsymbol{x}, \boldsymbol{y}^*)$.

Returning to Example 2.2, we can note in Equations (2.6)-(2.7) that the leader's objective function can be expressed as $F(x, y) = Q(f(x, y))$ where $Q(c) = \sin^2(c)$. Also, $f(x, z) = f(x, y^*)$ and $F(x, z) = F(x, y^*)$ when $z = \pm\sqrt{x}$ for $x > 0$. Therefore, an infeasible solution (x, z) is computed such that $f(x, z) = 0$ and $F(x, z) = 0$. That is, (x, z) is a pseudo-feasible solution related to (x, y^*).

2.5.1 Implication of Pseudo-feasibility in Evolutionary Computing for BO

The main issues on how pseudo-feasibility can affect evolutionary algorithms (and metaheuristics in general) when solving BOPs are the following.

Another Source of Difficulty

Because the presence of pseudo-feasible solutions in a BO problem might lead to inaccurate performance comparisons, two key issues must be addressed: (1) procedures to detect pseudo-feasible solutions, and (2) generate test problems where pseudo-feasible solutions are included.

Regarding the first point, depending on the characteristics of the BO problem, the detection of pseudo-feasible solutions can be quite specific. For example, a smooth and convex objective function at the lower level allows the application of some optimality conditions to determine whether a solution is indeed feasible or not [38]. However, designing a universal detection technique takes more time and effort. The second point is crucial because even when there are already approaches for constructing BO problems that have been published [114], pseudo-feasible solutions are not yet considered. Furthermore, in those test problems where the optimal solutions are known and there are pseudo-feasible solutions, the performance measures currently utilized in the BO area could lead to misleading comparisons [125].

Unstable Comparison Among Algorithms

Firstly, consider two metaheuristic algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$ which are used to solve the bilevel problem given by Equations (2.6)-(2.7). Suppose that $\mathcal{A}_1$ and $\mathcal{A}_2$ are always converging to $(0, 0)$ and $(1, -1)$, respectively. Note that, $f(1, -1) = 0$ and $F(1, -1) = 0$. Moreover, if the procedure and criteria to compare algorithms in [56] are adopted, we could say that $\mathcal{A}_1$ and $\mathcal{A}_2$ are not significantly different, and both algorithms are capable of solving the aforementioned task. However, it is clear that $\mathcal{A}_2$ converged to an infeasible solution, which is in fact far from the true optimum. Hence, different comparison mechanisms should be considered to avoid reporting misleading results.

2.5.2 Pseudo-feasible Solution Detection Mechanism

In [33], we characterized pseudo-feasible solutions for some bilevel optimization problems where the lower-level objective function $f(\boldsymbol{x}, \cdot)$ has a unique global optimal solution $\boldsymbol{y}^* \in \Psi(\boldsymbol{x})$ for each $\boldsymbol{x} \in X$. Besides, we proposed the following two conditions for detecting pseudo-feasible solutions.

Condition 1:

$$\begin{cases} \|\boldsymbol{x} - \boldsymbol{w}\| < \delta_1 \\ \|\boldsymbol{z} - \boldsymbol{y}^*\| \geq \delta_2(\boldsymbol{x}) > 0 \\ |F(\boldsymbol{x}, \boldsymbol{y}^*) - F(\boldsymbol{w}, \boldsymbol{z})| < \varepsilon_1 \\ |f(\boldsymbol{x}, \boldsymbol{y}^*) - f(\boldsymbol{w}, \boldsymbol{z})| < \varepsilon_2. \end{cases} \tag{2.8}$$

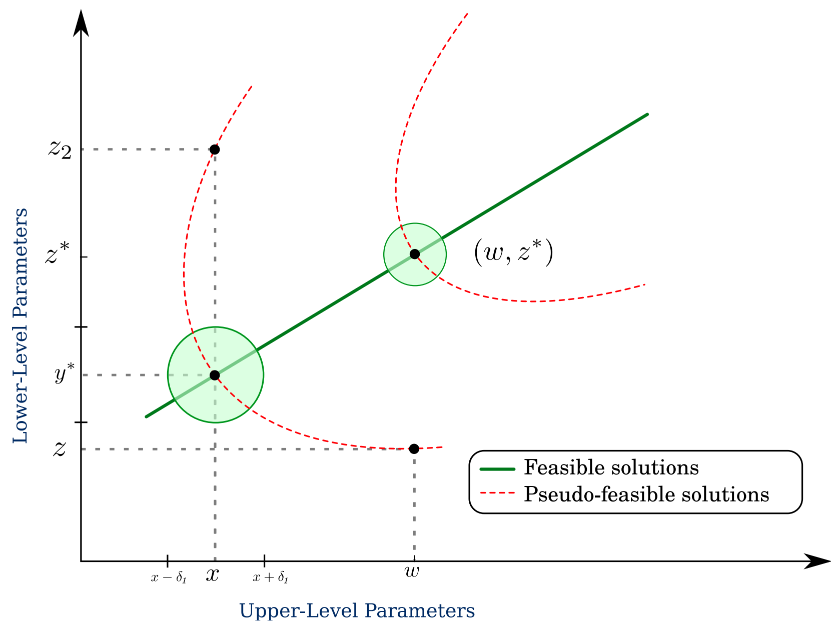


Figure 2.3: Representing pseudo-feasible solutions. Here $(\boldsymbol{w}, \boldsymbol{z})$ and $(\boldsymbol{x}, \boldsymbol{z_2})$ are two pseudo-feasible solutions related to a feasible solution $(\boldsymbol{x}, \boldsymbol{y}^*)$. Note that $\|\boldsymbol{x} - \boldsymbol{w}\| \geq \delta_1$ and $\|\boldsymbol{z_2} - \boldsymbol{y}^*\| > 0$ are illustrated.

Condition 1 can be interpreted as follows: Given $\boldsymbol{x}$ and $\boldsymbol{w}$ close enough and $\boldsymbol{z}$ far from $\boldsymbol{y}^* \in \Psi(\boldsymbol{x})$, and $F(\boldsymbol{x}, \boldsymbol{y}^*) = F(\boldsymbol{w}, \boldsymbol{z})$, and $f(\boldsymbol{x}, \boldsymbol{y}^*) = f(\boldsymbol{w}, \boldsymbol{z})$, imply that the objective function values associated to $(\boldsymbol{w}, \boldsymbol{z})$ are satisfying Definition 2.6, i.e., $(\boldsymbol{w}, \boldsymbol{z})$ is a pseudo-feasible solution.

Condition 2:

$$\begin{cases} \|\boldsymbol{x} - \boldsymbol{w}\| \geq \delta_1 \\ \|\boldsymbol{z} - \boldsymbol{z}^*\| \geq \delta_2(\boldsymbol{w}) > 0 \\ |F(\boldsymbol{x}, \boldsymbol{y}^*) - F(\boldsymbol{w}, \boldsymbol{z})| < \varepsilon_1 \\ |f(\boldsymbol{x}, \boldsymbol{y}^*) - f(\boldsymbol{w}, \boldsymbol{z})| < \varepsilon_2. \end{cases} \tag{2.9}$$

Note that Condition 2 is quite similar to Condition 1. However, the case where $\boldsymbol{x}$ is different with respect to $\boldsymbol{w}$ is considered. Moreover, Figure 2.3 gives a geometric interpretation of Conditions 1-2 by representing the variables space.

In order to compute Condition 1 or Condition 2, four parameters $\delta_1,\ \delta_2,\ \varepsilon_1$ and ε_2 have to be configured before numerically computing the conditions. It is worth mentioning that ε_1 and ε_2 are related to the target accuracy. The configuration of δ_1 and δ_2 can be carried out by using theoretical results such as the Mean Value Theorem [30, 88, 123], which suggest that $\delta_2 = \varepsilon_2/M$ with $|\partial f(\boldsymbol{x}, \boldsymbol{y})/\partial y_i| \leq M$, $i = 1, \ldots, D_{ll}$, for all $\boldsymbol{y}$ around $\boldsymbol{y}^*$. As this last result cannot be applied in the general case, then δ_1 and δ_2 are user-defined values at hand. Note that parameters δ_1 and δ_2 are non-negative numbers and should be as small as possible to avoid pseudo-feasible solutions, i.e., if a pair of solutions found $(\boldsymbol{x}, \boldsymbol{y}^*)$, $(\boldsymbol{w}, \boldsymbol{z})$ satisfy (2.8) with $\delta_1 > 0$ and $\delta_2 > 0$, then $(\boldsymbol{w}, \boldsymbol{z})$ is a pseudo-feasible solution. The concept of pseudo-feasible solutions can be weakened (for applicability purposes) by fixing values for δ_1 and δ_2.

There are other important properties related to Conditions 1 and 2 on improving the algorithms' comparison when solving BO problems. The first condition, in particular, can be useful for avoiding paired pseudo-feasible solutions offered by the lower-level optimizer. The second condition can be used to determine if a pseudo-feasible solution is related to two known feasible solutions, i.e., when an algorithm is solving a test problem where the optimum feasible solution is known, to determine whether the algorithm's obtained solution is indeed feasible and close to the optimum.

2.6 Multi-Objective Bilevel Optimization

Multi-objective Bilevel Optimization (MOBO) has become an important and challenging issue in evolutionary computation. The above is because different real-world problems require a mathematical model that considers multiple conflicting objective functions to be optimized simultaneously subject to another multi-objective problem [116]. This nested structure provides an interesting model to handle various engineering and scientific applications from optimization processes, game-playing, optimal control, strategy development, transportation problems, among others [115, 116, 133].

As mentioned before, a hierarchical structure is present in bilevel problems, where the UL optimization problem moves its decision variables and objectives, depending on the optimal solutions from a LL optimization problem. The bilevel model is even more complicated to solve when multiple objectives are present because an online decision-making process is required to handle the trade-offs from the LL problem. MOBO problems have been addressed using different approaches, including classical methodologies (e.g., mathematical programming) [43] and evolutionary approaches [116]. Particularly, the Evolutionary Computation community has proposed a variety of methods to tackle MOBO problems by using both evolutionary and swarm intelligence algorithms with successful results in test benchmarks and also in real-world problems [36, 77, 143].

2.6.1 Problem Definition

This section briefly describes Multi-Objective Bilevel Optimization (MOBO) and its variant, Semi-Vectorial Bilevel Optimization (SVBO). A MOBO problem is a bilevel optimization problem with one or two multi-objective optimization tasks in a hierarchical framework [43, 39].

Definition 2.7

Consider the following functions $F : X \times Y \to \mathbb{R}^M$, $f : X \times Y \to \mathbb{R}^m$ with M, m positive integers such that $M \geq 2$ or $m \geq 2$. Then, the following problem defines a MOBO problem: Minimize (UL)

$$F(\boldsymbol{x}, \boldsymbol{y}) = (F_1(\boldsymbol{x}, \boldsymbol{y}), F_2(\boldsymbol{x}, \boldsymbol{y}), \ldots, F_M(\boldsymbol{x}, \boldsymbol{y})),\ \boldsymbol{x} \in X \tag{2.10}$$

subject to (LL):

$$\boldsymbol{y} \in \underset{\boldsymbol{y} \in Y}{\text{argmin}} \left\{ \begin{matrix} f(\boldsymbol{x}, \boldsymbol{y}) = (f_1(\boldsymbol{x}, \boldsymbol{y}), \ldots, f_m(\boldsymbol{x}, \boldsymbol{y})) : \\ g_j(\boldsymbol{x}, \boldsymbol{y}) \leq 0,\ j = 1, 2, \ldots, J \end{matrix} \right\} \tag{2.11}$$

$$G_k(\boldsymbol{x}, \boldsymbol{y}) \leq 0,\ k = 1, 2, \ldots, K; \tag{2.12}$$

where $G_k, g_j : X \times Y \to \mathbb{R}$, represent the upper and lower-level constraints ($1 \leq k \leq K$, $1 \leq j \leq J$), respectively.

The problem in Definition 2.7 can be rewritten as follows:

Minimize $F(\boldsymbol{x}, \boldsymbol{y}),\ \boldsymbol{x} \in X$, subject to $\boldsymbol{y}* \in \Psi(\boldsymbol{x})$ where

$$\Psi(\boldsymbol{x}) = \arg\min_{\boldsymbol{y} \in Y} \left\{ \begin{aligned} f(\boldsymbol{x}, \boldsymbol{y}) \quad &= (f_1(\boldsymbol{x}, \boldsymbol{y}), \ldots, f_m(\boldsymbol{x}, \boldsymbol{y})) : \\ &g_j(\boldsymbol{x}, \boldsymbol{y}) \leq 0, j = 1, 2, \ldots, J \end{aligned} \right\} \tag{2.13}$$

Note that the LL is represented here as the set-valued mapping Ψ for an UL decision vector $\boldsymbol{x}$, which can help to describe both optimistic and pessimistic MOBO instances.

It is worth mentioning that both levels have a multi-objective optimization problem. Because the leader expects the follower to make judgments based on a set of Pareto solutions, the feasibility concept varies from single-objective bilevel optimization. As a result, MOBO problems are difficult to solve, and both optimistic and pessimistic formulations are difficult to solve [43, 120].

When $m > 1$, the feasibility concept in MOBO differs from that in Single-Objective Bilevel Optimization (SOBO) since the leader might demand that the follower make decisions based on a set of Pareto-optimal options.

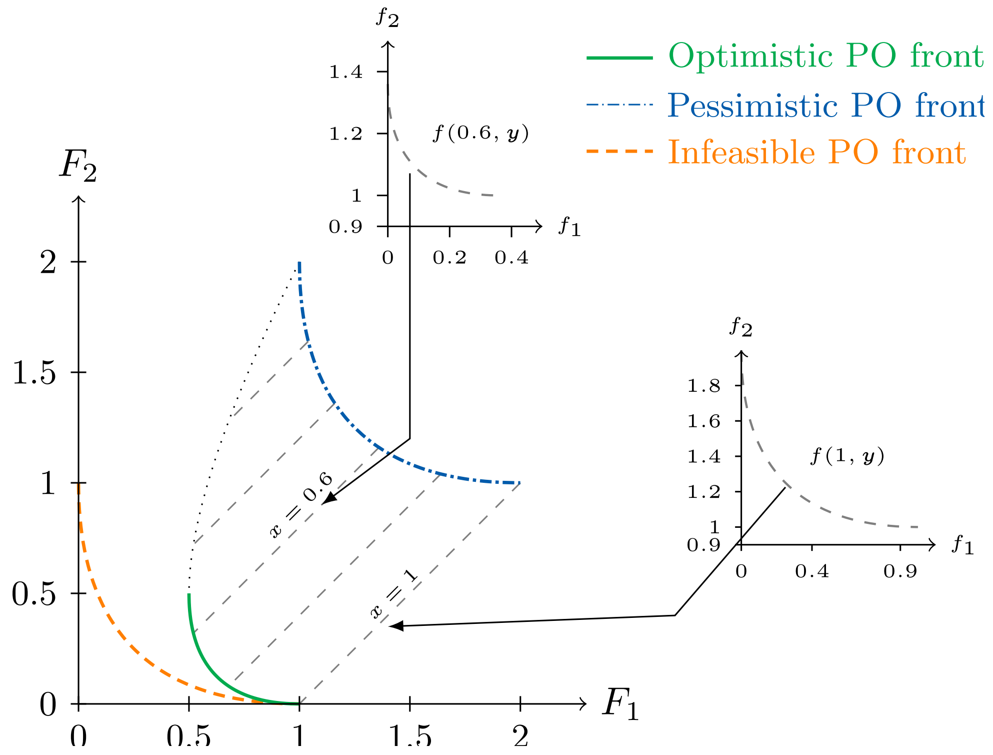


Figure 2.4: Pareto-Optimal (PO) fronts from Example 2.3. The leader F-space represents both the upper and lower PO fronts, as well as the relationship between the leader and the follower at various UL decision vectors.

Example 2.3

Firstly, let us assume that $x \in X = [-1, 2]$, $\boldsymbol{y} = (y_1, y_2) \in Y = [-1, 2]^2$ which are decision vectors for the following MOBO problem:

$$\min_{(x,\boldsymbol{y}) \in X \times Y} F(x, \boldsymbol{y}) = \begin{pmatrix} (y_1 - 1)^2 + y_2^2 + x^2, \\ (y_1 - 1)^2 + y_2^2 + (x - 1)^2 \end{pmatrix} \tag{2.14}$$

subject to:

$$\boldsymbol{y} \in \underset{\boldsymbol{y} \in Y}{\text{argmin}} \left\{ f(x, \boldsymbol{y}) = \begin{pmatrix} y_1^2 + y_2^2, \\ (y_1 - x)^2 + y_2^2 \end{pmatrix} \right\}. \tag{2.15}$$

For a given value of x, this problem has numerous possible solutions, the Pareto-optimal (PO) solutions from the LL problem $\boldsymbol{y} \in \Psi(x) = \{(y_1, y_2) \in Y : y_1 \in \; [0, x], y_2 = 0\}$. Note that when the leader's problem is optimized with the LL decision vector $\boldsymbol{y}$ as part of the UL decision vector, an infeasible UL PO front can be obtained. That is, putting $\boldsymbol{z} = (x, \boldsymbol{y})$ and optimizing $F(\boldsymbol{z})$ without considering the LL problem (see Fig. 2.4).

2.6.2 Optimistic and Pessimistic Scenarios

Because the leader can expect followers to make decisions that assist or hurt its aims, optimistic and pessimistic viewpoints arise. When the LL problem contains many optimal solutions, these places are likewise present in SOBO [116].

Moreover, the optimality conditions when the UL defines a MOP, generate different definitions for the optimistic and pessimistic scenarios. Those MOBO locations in [2] are defined as finding optimal viable solutions in the Inducible Area (IR) or feasible region $(\boldsymbol{x}, \boldsymbol{y}) \in IR = \{(\boldsymbol{x}, \boldsymbol{y}) \in X \times Y : \boldsymbol{y} \in \Psi(\boldsymbol{x}), G_k(\boldsymbol{x}) \leq$

$0, k = 1, \ldots, K\}$ such that it is not another $(\boldsymbol{x}', \boldsymbol{y}') \in IR$ such that $F(\boldsymbol{x}', \boldsymbol{y}') \preceq F(\boldsymbol{x}, \boldsymbol{y})$ for the optimistic position and $F(\boldsymbol{x}, \boldsymbol{y}) \preceq F(\boldsymbol{x}', \boldsymbol{y}')$ for the pessimistic MOBO.

2.7 Bilevel Optimization Test Problems

This section is used to describe different BO problems reported in the specialized literature that are used to assess the algorithm performance. Here, two suite sets are considered because they contain interesting properties related to real-world problems.

2.7.1 SMD Test Suite

The SMD test problems were proposed in [112, 113, 114]. There are twelve test functions in this test suite for box-constrained and constrained test problems. The first eight SMD test issues were supplied to provide controlled difficulty scalability at both levels. In addition, for each problem, the best solutions are provided. The main properties of SMD test problems are identified, and some of the most important are enumerated.

1. Multi-modality at upper level (SMD7 and SMD8).
2. Multi-modality at lower level (SMD3-SMD5 and SMD8).
3. Multiple global solutions (SMD6).
4. Inequality constraints and different properties associated to the objective functions (SMD9-SMD12).

This test suite has been regularly used to test preliminary algorithm versions by providing various sources of difficulty. Parameters in the SMD test problems regulate issues like convergence and interaction between the higher and lower-level problems.

	Upper Level		**Lower Level**	
	Multimodal	Differentiable	Multimodal	Differentiable
PMM1	No	No	No	Yes
PMM2	No	No	No	Yes
PMM3	No	No	Yes	Yes
PMM4	No	No	Yes	Yes
PMM5	No	No	Yes	No
PMM6	Yes	No	Yes	Yes

Table 2.1: Properties related to each instance problem in the PMM test suite.

2.7.2 PMM Test Suite

This subsection presents the PMM test suite, proposed in this thesis work (details in Appendix C), that contains six test problems for bilevel optimization. The main source of difficulty in those problems is that related to pseudo-feasible solutions, particularly at the optimal feasible solution. This test suite contains six problems; Table 2.1 summarizes the main property related to each test problem in the PMM test suite. Those test problems (named PMM1 to PMM6) are constructed so that the feasible optimum solution lies in $x = (0, \dots, 0) \in [-10, 10]^{D_{ul}}$ and $y = (0, \dots, 0) \in [-10, 10]^{D_{ll}}$ (see Appendix C for more details). Also, each UL problem is only differentiable on those feasible solutions.

2.8 Conclusions of the Chapter

This chapter introduced the bilevel optimization problem over continuous spaces and how such problem is currently solved. Moreover, a warning sign for evolutionary algorithms in bilevel optimization problems was presented. Such issue is related to misleading results that metaheuristic algorithms can report. Some examples were given to illustrate the defined pseudo-feasibility. Also, some theoretical results on bilevel optimization were discussed. Finally, two test problems sets for bilevel optimization, one designed as part of this thesis, were also included in this chapter.

Chapter 3

Literature Review

In this part, those BO solution approaches are reviewed by dividing bilevel problems regarding the number of objective functions at each level.

3.1 Single-Objective Bilevel Optimization

Regarding solution methods, many researchers have offered strategies for solving bilevel problems due to the important properties that bilevel models can provide [38, 109]. Such strategies include mathematical approaches from mathematical programming and Karush-Kuhn-Tucker conditions for single-level reduction. Other strategies include metaheuristic approaches, such as genetic algorithms, evolution strategies, and swarm intelligence [7, 40, 82], among others [43]. This part of the thesis document is on analyzing the context of metaheuristic approaches. The reader is referred to [38] for the analysis of classical approaches.

As previously stated, the majority of research efforts for population-based metaheuristics for solving BO problems are centered on evolutionary algorithms and swarm intelligence. This inclination stems from the fact that those algorithms have been successful in solving single-level optimization problems. However, depending on the number of evaluations, their computing cost in a bilevel problem might be rather substantial, especially at the lower level of the BO problem. In contrast, combining population-based metaheuristics with mathematical programming methods may provide efficient and accurate procedures with lower computational costs [116, 125]. The following approaches have been identified on the resolution of bilevel problems by using metaheuristic approaches.

1. **Single-level reduction:** This approach has two main stages, (1) reformulating the bilevel problem into a single-level optimization task, (2) employing a metaheuristic to solve the transformed problem. Usually, in the first stage, the Karush-Kuhn-Tucker conditions are used for the transformation by assuming that the problem is mathematically well-behaved. As a consequence, this strategy is restricted to differentiable functions [29, 38].

2. **Nested approaches:** Here, two metaheuristics (one for each level) are used to solve each level sequentially. Besides, nested-based strategies can be effective but with a high computational cost when objective functions are expensive to calculate or when high-dimensionality is present.

3. **Surrogate-assisted approaches:** These approaches use the nested scheme to solve problems. However, the lower-level problem is not solved by another metaheuristic but using instead an approximate strategy to infer feasible solutions during the process. Such approximate strategy is particularly useful to save objective function evaluations.

4. **Non-nested approaches:** In this approach, the problem is solved using single-level optimization metaheuristics and employing *a priori* information of the lower-level problem to avoid evaluating it as in the nested scheme. That is, the upper and lower-level problems are evaluated in a single step without optimizing the lower-level as usual.

3.1.1 Single-Level Reduction

Single-level reduction is an indirect method to solve BO problems because before solving the problem, a transformation is required. Assuming that the lower-level contains smooth constraints and objective function, then the BO problem can be transformed into a single-level optimization problem. In most cases, the Karush-Kuhn-Tucker conditions are used for such a purpose (when they apply), i.e., the lower-level conditions convert the lower-level optimization problem into equality and inequality constraints, providing a more tractable problem instance.

The following formulation is used to exemplify the result of transforming a bilevel problem. Firstly, assume the BO problem in Definition 2.1, then the following reformulation is obtained when the KKT conditions can be used:

$$\min\{F(\boldsymbol{x},\ \boldsymbol{y})\ :\ \boldsymbol{x} \in X,\ \boldsymbol{y} \in Y\}$$

subject to

$$\begin{aligned} G_k(\boldsymbol{x}, \boldsymbol{y}) &\leq 0, \\ \nabla_{\boldsymbol{y}} L(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\lambda}) &= 0, \\ g_j(\boldsymbol{x}, \boldsymbol{y}) &\leq 0, \\ \lambda_j g_j(\boldsymbol{x}, \boldsymbol{y}) &\leq 0, \\ \lambda_j &\geq 0, \end{aligned}$$

where $k = 1, \ldots, K$, $j = 1, \ldots, J$ and

$$L(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\lambda}) = f(\boldsymbol{x},\ \boldsymbol{y}) + \sum_{j=1}^{J} \lambda_j g_j(\boldsymbol{x},\ \boldsymbol{y}).$$

This transformed problem is equivalent to a single-level constrained optimization problem. Moreover, under certain assumptions, optimal feasible solutions for this reformulation are solutions for the corresponding bilevel problem [109, 121].

The specialized literature reports that single-level reduction is performed into two main stages. The first one includes the single-level reduction strategy to transform the problem into an equivalent single-level problem [38]. The second one aims to optimize the transformed problem using an exact approach such as branch-and-bound algorithms, interior region approaches, among others [38]. It is common that the resulting single-level optimization problem can not be solved by an exact approach, but metaheuristic approaches can be considered to approximate optimal solutions [116]. Metaheuristic algorithms that solved transformed problems have been proposed. For instance, in [135], the authors constructed a specific two-objective optimization problem and proposed a new constraint-handling technique to solve it. The proposed constraint-handling method can handle linear and nonlinear constraints. Regarding particle swarm optimization, in [69] the authors used the KKT condition for single level reduction, but the resulting problem is smoothed by a Chen-Harker-Kanzow-Smale (CHKS) smoothing function. Finally, the resulting problem is solved by a PSO algorithm.

3.1.2 Nested approach

Regarding the nested scheme, different metaheuristic algorithms have been adapted to solve BOPs, e.g., Nested-DE [6], Nested-PSO [82], and BL-CMEA-ES [56] use one EA to compute lower-level solutions for a decision vector generated by another EA. Given the utilization of two nested EAs, a substantial computational cost might be required in such approaches. In [125], two hierarchical optimization algorithm approaches are described, the first one is based on repairing mechanisms (one population with a follower as a constraint) and the second is a constructive approach (two algorithms with separate populations). In both circumstances, an algorithm must be chosen for each level, which is frequently a combination of an EA and an exact approach, as previously discussed [57, 75].

3.1.3 Surrogate Assisted Algorithms

Different population-based bilevel optimization algorithms that use metamodels to lower the computing cost of objective function evaluations at both levels have been reported in the specialized literature. Different EAs have been adapted to address BOPs in the nested scheme in addition with surrogate-models. Due to the high computational cost of such a strategy, it is customary to discover approximate approaches to save function evaluations [64, 65]. Here, some representative algorithms are described as follows.

Genetic algorithms that use some assistance are BLEAQ [118], BLEAQ-II [111], and ε-KKT [121], which use two nested genetic algorithms with quadratic approximation to the lower-level optimal solution to reduce the cost related to evaluating the LL problem. They also use the well-known Sequential Quadratic Programming (SQP) method to estimate lower-level optimum solutions in a local search.

Regarding different evolutionary algorithms to save evaluations, SABLA [64], BLMA [66], and BLEGO [65] approaches employ two nested DE algorithms and several surrogate models for the lower-level function. SABLA implements an Interior Point (IP) SQP method; BLMA uses Response Surface Models and Kriging; and BLEGO uses a combination of DE, EGO, and Gaussian Process Models. Finally, Surrogate-assisted BIDE [7] is s different approach because it implements two DE algorithms, one for each level, and updates the lower-level problem by introducing a metamodel based on the k-nearest neighbors method.

Algorithms from nested and surrogate approaches are summarized in Table 3.1.

To sum up, multiple works on solving bilevel optimization problems by using metaheuristic have been reported. However, to the best of the author's knowledge, the presence of pseudo-feasible solutions is taken into account. Finally, no physics-based approaches have been considered in the solution of BOPs.

Algorithm	UL	LL	Description
Nested-DE [6]	DE	DE	Differential Evolution search engine is used at both level.
Nested-PSO [82]	PSO	PSO	PSO search is performed at both levels updating the particles regarding the corresponding objective function.
BL-CMEA-ES [56]	CMEA-ES	CMEA-ES	Use two nested Covariance Matrix Adaptation Evolutionary Strategies with memory sharing to reduce computational effort when computing lower-level solutions.
BLDES [8]	DE	DE	Use Radial Basis Functions, k-Nearest Neighbors and Local Linear Regression to approximate objective function values at both levels.
ε-KKT [121]	GA	GA/SQP	The objective functions and constraints are approximated via quadratic and linear mappings to reduce the number of function evaluations.
BLEGO [65]	DE/EGO	EGO	Gaussian Process Model to approximate objective function values at both levels.
BLMA [66]	DE	DE	Response Surface Models and Kriging are used to approximate feasible solutions.
SABLA [64]	DE-IP	DE-IP	Sequential Quadratic Programming is used to handle inaccurate solutions.
BLEAQ(-II) [111, 118]	GA	GA/SQP	Approximations of the lower-level optimization problem using SQP.
Surrogate-assisted BIDE [7]	DE	DE	Uses k-Nearest Neighbors as meta-model to approximate lower-level solutions.

Table 3.1: Related work on nested approaches for single-objective bilevel optimization problems. Note that nested and surrogate approaches are included.

3.2 Multi-Objective Bilevel Optimization

This section reviews the most relevant approaches for solving MOBO problems. This part includes both classical and metaheuristic approaches.

3.2.1 Classical Approaches

Different exact techniques have been developed to solve linear, convex, or quasiconcave MOBO problems by converting them to single-level optimization tasks as a general framework. Four approaches have been identified: Fuzzy Methods, Penalty Function Methods, Methods based on Karush-Kuhn-Tucker Conditions, and Pareto Frontier Generators.

Fuzzy Methods

The proposal that uses fuzzy logic concepts to discover efficient solutions to MOBO problems is referred here to as a fuzzy technique. For instance, in [110], one of the first fuzzy-based approaches is described, in which the authors used a nested task to solve a non-linear multi-objective bilevel decision-making problem. It features an interactive algorithm based on the epsilon-constraint method as well as fuzzy logic ideas like satisfactoriness for dealing with UL and LL preferences.

After that, in [142], a single-level reduction approach is described, as well as the use of fuzzy logic ideas for single-level reduction to apply an interactive optimization method. In this sense, in [144], another interactive solution is proposed, this time using the satisfactoriness idea at the UL and a measurement function to tackle the LL problem.

On the other hand, in [139], an alternative fuzzy-based solution called FTOPSIS is described, which employs preference ordering by similarity to the ideal point and combines Taylor series and Karush-Kuhn-Tucker conditions to turn the problem into a fractional programming problem.

Penalty Function Methods

The exact methods using a penalty function approach are, in general, devoted to perform a level reduction by considering the LL functions at the UL as a penalization term.

Firstly, in [87, 138], the authors present a methodology for performing a single-level reduction using an exact penalization function and then propose a weighted-sum-based algorithm that handles the multiple objectives using the linear scalarization transformation.

Besides single-level reduction, penalty function methods are also used as a constraint-handling technique. Some optimality conditions to transform the MOBO problem into a single-level problem were provided in [137].

Methods based on Karush-Kuhn-Tucker Conditions

As previously stated, KKT conditions are commonly used for single-level reduction when the LL problem contains convex and smooth objective functions and constraints. This allows using such conditions to propose exact methods to solve MOBO problems.

The authors of [67, 68] proposed a hybrid approach to solve MOBO problems with stochastic UL variables in the objective functions, which was the first application of KKT conditions in MOBO problems.

When the KKT conditions are used for reformulation, the transformed problem is not equivalent to the original bilevel programming problem, even for LL convex problems [39]. However, there are alternatives to the KKT conditions, such as using an optimal value function reformulation as detailed in [86], which is particularly useful for semivectorial linear bilevel optimization problems that are reformulated to single-level non-smooth optimization problems.

Classical Pareto Frontier Generators

Most traditional approaches recommend *a priori* preferences or interactive methods to approximate a solution in the Pareto-optimal set. However, some methods attempt to approximate solutions (not just one) along the Pareto optimal front, and these are referred to as *Pareto Frontier Generators*.

This approach was discussed in [103], where a strategy to solve bilevel linear multi-objective programming problems in two steps was defined.

The use of branch-and-bound strategies has been also used to handle bilevel linear problems [3]. The UL in this situation is a MOP, while the LL is a single-objective optimization problem.

3.2.2 Nested Multi-objective Metaheuristics

Evolutionary algorithms for MOBO have been proposed by extending Evolutionary Multi-objective Algorithms incorporating novel mechanisms to handle this hierarchical structure.

For example, a popular algorithm, known as Bilevel Evolutionary Multi-objective Optimization (BLEMO), uses two nested NSGA-II to approximate solutions for MOBO problems [36]. Moreover, different variants of BLEMO were also developed. For instance, the Hybrid and Self Adaptive Bilevel Evolutionary Multi-objective Optimization Algorithm (H-BLEMO) uses the SQP algorithm as a local search mechanism to improve the BLEMO results [37]. Another example is the Multi-objective Bilevel Evolutionary Algorithm based on multiple quadratic fibers (mf-BLEAQ), which uses quadratic approximations to feasible solutions to save function evaluations [119]. On the other hand, swarm-intelligence-based algorithms have been proposed with competitive results [143].

BLEMO [36] utilizes a traditional representation of solutions, i.e., solutions $(\boldsymbol{x}, \boldsymbol{y}) \in X \times Y$ are used as the base to carry out the variation operators. This representation can be inefficient since solutions with different objective values are represented by the same upper-level decision vector, promoting redundancy and high memory usage. BLEMO also uses lower-level subpopulations (besides the population) to save LL evaluations. Regarding H-BLEMO [37], its representation is similar to BLEMO but incorporates different mechanisms to reduce the population size based on the problem dimension and inherits the same issues observed in BLEMO. Moreover, mf-BLEAQ [119] mitigates the issues in BLEMO by incorporating sophisticated mechanisms when the subpopulations are generated. However, all solutions are always compared during the non-dominated sort making it unable to reduce the computational effort. Regarding the PSO-based algorithms, EQPSO [143] adopts the same strategy as BLEMO, and in turn, the same related problems with the solution representation.

3.2.3 Non-nested Multi-objective Metaheuristics

The LL problem can be incorporated within the upper one under certain situations. As a result, the multi-objective bilevel model, turned into a multi-objective single-level problem, can be solved using a single multi-objective metaheuristic algorithm, referred to as non-nested multi-objective method.

As seen in [70], [79] and [80], the LL problem can be integrated into the UL as constraints. The KKT conditions are used to achieve embedding in [79, 80], while the primal and dual theory is used in [70]. All of these works use a multi-objective metaheuristic optimizer with a constraint-handling technique to discover a Pareto approximation once the MOBO problem is reduced to a simple MOP. Moreover, in [70], [79] and [80], an alternative form of a Genetic Algorithm (GA), Decomposition-based Constrained Multiobjective Differential Evolution (CMODE/D), and Non-dominated Sorting Genetic Algorithm (NSGA-II), and a variation of the Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D-DE) are respectively used.

Finally, in [20, 134], the LL problem is included in a new method. The objectives of both levels become a fuzzy one in [20]. The updated fuzzy form of the problem is then solved using a GA. Finally, the LL is not expressly integrated in [134]. Even so, MOEA/D includes a new genetic operator to ensure that a solution meets the problem constraints, including those imposed by the LL problem.

3.3 Conclusions of the Chapter

This chapter reviewed the most important works on single- and multi-objective bilevel optimization. Regarding the single-objective case, three main approaches have been identified: (1) single-level reduction strategies, (2) nested algorithms, and (3) surrogate-assisted algorithms. The first approach uses the available mathematical properties to transform the bilevel model into a single-level problem, whilst approaches (2) and (3) use a nested scheme to adapt an optimizer for each level. Surrogate or meta-models in approach (3) are adopted to reduce the computational cost of evaluating the objective functions.

It is worth mentioning that no studies were found on how pseudo-feasibility can affect the two-level evolutionary approach in the current state of the art. Besides, most of the works are adapting evolutionary algorithms and swarm intelligence methods to solve bilevel problems. Therefore, studies on the effect of pseudo-feasibility can be carried out to analyze the performance in the presence of pseudo-feasible solutions. Also, different algorithms such as physics-inspired algorithms can be designed and studied in this context.

Regarding the studies on multi-objective bilevel optimization, three main solution approaches are identified from this literature review: (1) Classical Approaches, (2) Nested Multi-objective Metaheuristics, and (3) Non-nested Multi-objective Metaheuristics. Regarding classical approaches, it was observed that most of the works use fuzzy-based methods to solve linear problems in comparison to penalty function methods and those using the KKT conditions for problem transformation. Nested metaheuristics, on the other hand, are the most commonly used because the non-nested methods require a theoretical study to transform the bilevel problems that can not be applied in most cases.

Moreover, the adapted multi-objective algorithms to handle multi-objective bilevel problems use a traditional representation of solutions even when the hierarchical structure can be better represented with a more sophisticated representation.

Chapter 4

Nested-BCA: Preliminary Proposal

Information

Part of this chapter is based on the paper: Jesús-Adolfo Mejía-de-Dios and Efrén Mezura-Montes, *A Physics-Inspired Algorithm for Bilevel Optimization*, in Proceedings of the 2018 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC), pages 1-6, IEEE Press, 2018, [89].
https://doi.org/10.1109/ROPEC.2018.8661368

4.1 Introduction

From the literature review mentioned before, regarding population-based meta-heuristics for BO problems, most research efforts are focused on evolutionary computation and swarm intelligence algorithms because they have been successfully applied to solve single-level optimization problems [19]. In recent years, physics-inspired algorithms have become popular to solve complex single-level optimization problems [58, 25]. It is worth mentioning that the algorithm performance is quantified via numerical indicators such as accuracy (distance to the true optimum), robustness, and *rationality*, among others [125]. Therefore, an algorithm is said to be competitive if it reports results with the desired values of the performance indicators.

One of the main motivations of this research work is derived from a competitive physics-inspired algorithm named Evolutionary Centers Algorithm (ECA), which was originally designed to solve global optimization (single-level) problems [90], but it is now extended and adapted to deal with BO problems. The unconstrained nested scheme is considered at the beginning. This ECA extension for solving BOPs is called Bilevel Centers Algorithm (BCA), and uses the search elements provided by ECA, which is based on the center of mass concept for generating biased solutions from promising regions in the search space [90].

The rest of the chapter is organized as follows: Section 4.2 presents a variation operator using the center of mass concept. The proposed BCA is detailed in Sections 4.3-4.4. After that, Section 4.5 shows the experimental design and the corresponding results, and discusses BCA's performance. A state-of-the-art algorithm for BO is used for comparison purposes. The conclusions of this research are described in Section 4.6.

4.2 The Center of Mass Concept as a Variation Operator

The center of mass has been used as a concept to propose a variation operator for generating good candidate solutions for an optimization problem. The main idea is to use the objective function values as the mass unit, and assume that desired values correspond to a larger mass, i.e., a solution has a larger mass than another one if the objective function is better in the first solution [93].

4.2.1 Center of Mass Concept

Firstly, the center of mass concept is described. After that, the variation operator based on the center of mass concept will be analyzed.

The center of mass is the unique point $\boldsymbol{c}$ such that the mass distribution $U = \{\boldsymbol{u}_1,\ \boldsymbol{u}_2, \dots, \boldsymbol{u}_K\}$ in a space has the following property: The weighted sum of K position vectors relative to this point is zero [90], i.e.

$$\sum_{i=1}^{K} m(\boldsymbol{u}_i)(\boldsymbol{u}_i - \boldsymbol{c}) = 0, \text{ implies } \boldsymbol{c} = \frac{1}{M}\sum_{i=1}^{K} m(\boldsymbol{u}_i)\boldsymbol{u}_i, \tag{4.1}$$

where $m(\boldsymbol{u}_i)$ is the mass located in $\boldsymbol{u}_i$ and M is the sum of the masses at each point in U. It is worth noticing that m is a non-negative function. If the objective function f is assumed non-negative and needs to be maximized, the mass function can be directly defined as $m(\boldsymbol{u}) := f(\boldsymbol{u})$. The following example is used to illustrate the application of the center of mass concept in maximization problems.

Example 4.1

Assume you want to maximize

$$f(\boldsymbol{x}) = 2\exp(-(x_1+1)^2 - (x_1+1)^2) + \exp(-(x_1-1)^2 - (x_1-1)^2)$$

such that $-5 \le x_1, x_2 \le 5$. It can be noted that f is non-negative.

Now, consider the following mass distribution $U = \{(-1,0), (1,2), (0,-1), (0.5,0.5), (-1.6,-1), (1.5,1)\}$. The corresponding masses are computed as follows. Since f is non-negative, then the mass for each point is $m(\boldsymbol{u}) = f(\boldsymbol{u})$, then the center of mass for U is given by the following expression.

$$\boldsymbol{c} = \frac{1}{M}\sum_{\boldsymbol{u}\in U} f(\boldsymbol{u}) \cdot \boldsymbol{u}, \text{ with } M = \sum_{\boldsymbol{u}\in U} f(\boldsymbol{u}).$$

Figure 4.1 represents the mass associated with the Example 4.1.

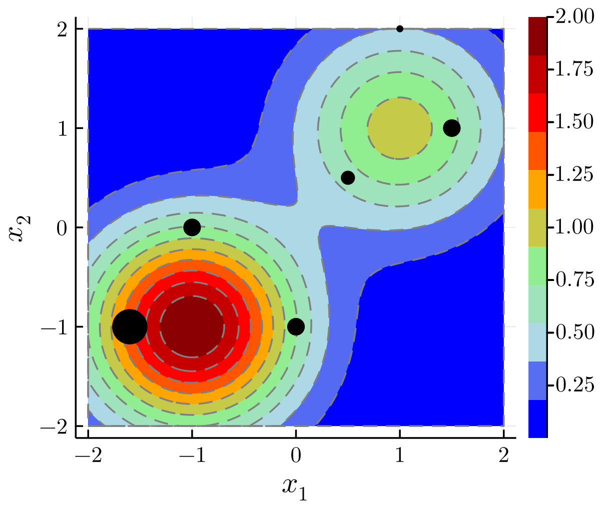


Figure 4.1: Center of mass representation. The radius represents the mass value which is associated with better values for the objective function.

In a general optimization problem, the objective function is not necessarily satisfying the non-negativity property. Therefore, the following translating and escalating procedure has been proposed to use the center of mass concept in the general minimization task. The mass calculation regarding a set U is obtained as follows:

$$m(\boldsymbol{x}) = 2\left(\max_{\boldsymbol{u}\in U} |f(\boldsymbol{u})|\right) - f(\boldsymbol{x}) \tag{4.2}$$

From Equation (4.2) can be observed that $m(\boldsymbol{x})$ is a non-negative function satisfying $m(\boldsymbol{x})$ is larger when $f(\boldsymbol{x})$ is minimum, and vice versa. Those are the desired properties for a general minimization problem.

4.2.2 The Variation Operator: Unconstrained Case

Now, the variation operator can be described. As mentioned in Subsection 4.2.1, the center of mass provides a bias toward a solution that optimizes the objective function. However, other aspects are considered, e.g., a suitable strategy to generate a new solution without promoting premature convergence. To tackle this issue we propose the following formulation [93].

Let P be a population with N individuals, and $U \subset P$ with size K. Thus, a candidate solution is determined by

$$\boldsymbol{x}_{\text{new}} = \boldsymbol{x} + \eta(\boldsymbol{c} - \boldsymbol{u}_{\text{worst}}), \tag{4.3}$$

where $\boldsymbol{x} \in P$ is the solution used to compute a new candidate, and $\boldsymbol{u}_{\text{worst}}$ is computed as follows:

$$\boldsymbol{u}_{\text{worst}} \in \text{argmin}\{f(\boldsymbol{u}) \ : \ \boldsymbol{u} \in U\}.$$

A set of candidate solutions is generated as follows: For each $\boldsymbol{x} \in P$, randomly choose elements (without replacement) in P to compute $U \subset P$ with size K and a positive stepsize value η to compute $\boldsymbol{x}_{\text{new}}$. At the end of that step, you will have N new candidate solutions [93].

4.2.3 The Variation Operator: Constrained Case

The bias generated by the center of mass has been useful to solve unconstrained problems [93]. However, poor performance can be observed in constrained optimization problems, due to the mass being particularly defined on the objective function values. This section is used to propose a new definition for the mass to also consider constrained problems.

Let P be a population with size N. Assume you are solving a constrained optimization problem (Definition 1.1). The mass is computed as follows for all solutions in P.

$$m(\boldsymbol{x}) = 2\left(\max_{\boldsymbol{u} \in U} |f_c(\boldsymbol{u})|\right) - f_c(\boldsymbol{x}) \tag{4.4}$$

where

$$f_c(\boldsymbol{x}) = f(\boldsymbol{x}) + 2f_{\text{max}} \cdot \text{vio}(\boldsymbol{x}) \tag{4.5}$$

$$\text{vio}(\boldsymbol{x}) = \sum_{j=1}^{J} \max\{0, g(\boldsymbol{x})\} + \sum_{k=1}^{K} |h(\boldsymbol{x})| \tag{4.6}$$

Note that $\text{vio}(\boldsymbol{x})$ is the constraint violation sum, and $f_{\max} = \max_{\boldsymbol{x} \in P} f(\boldsymbol{x})$ is the largest value of the objective function in the current population. Also, $f_{\max}$ is used as penalty factor.

Moreover, the mass defined by Equations (4.4)-(4.6) satisfies the following properties:

- It is a non-negative function. This property is a necessary condition to construct a consistent solution.
- Equations (4.2) and (4.4) take the same value on feasible solutions. That is, Equation (4.2) takes the same value as Equation (4.4) if $\boldsymbol{x}$ is feasible.
- Feasible solutions have a larger mass than infeasible solutions in the same population. Also, with two solutions with the same constraint violation sum, the mass will be reflected by the objective function values.

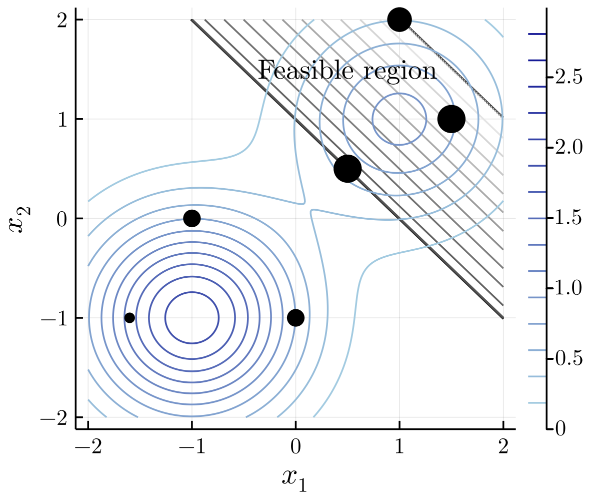


Figure 4.2: Center of mass representation in constrained spaces. It can be noted that feasible solutions have a larger mass even when there are other solutions with better objective function values.

Returning to Example 4.1, the following constraints are incorporated to the optimization problem: $g_1(\boldsymbol{x}) = 1 - x_1 - x_2, g_2(\boldsymbol{x}) = x_1 + x_2 - 3$. Figure 4.2 shows the mass distribution for the constrained optimization problem.

Summarizing, the center of mass concept is used to propose a suitable variation operator which can be used to generate promising solutions in constrained and unconstrained problems. The Equation (4.3) will be used as variation operator to generate a candidate solution for bilevel optimization, due to the mass function m has been adapted for each optimization level.

4.3 Generic Framework for Bilevel Optimization

This section presents the components of an Evolutionary Bilevel Optimization Algorithm (EBOA) [113, 116, 125]. Algorithm 1 summarizes each component.

Algorithm 1: Nested Algorithm.

```
1  Input: Upper and lower-level metaheuristics.
2  Output: Approximate optimal solution.
3  Initialize a population of UL vectors.
4  For each UL vector, solve the LL problem.
5  Evaluate UL objective function and constraints.
6  while stopping criteria is not met do
7      Generate new UL vectors from population.
8      For each UL vector, solve the LL problem.
9      Evaluate UL objective function and constraints.
10     Perform UL environmental selection to reduce population if necessary.
11 Report the best solution in population (regarding UL objective function).
```

4.3.1 Representation

Assuming the single-objective bilevel optimization case, a suitable encoding for an EBOA is given by the mathematical representation of a solution, i.e., the pair $(\boldsymbol{x}, \boldsymbol{y}) \in X \times Y$ can be useful at the beginning. Therefore, the population for a BO problem can be defined as in Equation (4.7):

$$P = \{(\boldsymbol{x}_1,\ \boldsymbol{y}_1),\ (\boldsymbol{x}_2,\ \boldsymbol{y}_2), \dots, (\boldsymbol{x}_N,\ \boldsymbol{y}_N)\} \subset X \times Y. \tag{4.7}$$

Note that P is a finite subset of solutions contained in the search space $X \times Y$. Those solutions or individuals in the population should be feasible, but guaranteeing feasibility is hard to meet. Here, feasibility means satisfying equality and inequality constraints but also the lower-level optimality, i.e., $\boldsymbol{y}_i \in \operatorname{argmin}\{f(\boldsymbol{x}_i,\ \boldsymbol{y}) : \boldsymbol{y} \in Y\}$ for $i = 1, \dots, N$.

4.3.2 Initialization

For single-level optimization, the population is, in general, initialized at random with uniform distribution of solutions in the search space [93]. The same occurs for the bilevel case. That is, the decision variables at the upper-level are initialized at random with uniform distribution within the corresponding search space (X). After that, the lower-level decision variables are obtained from the lower-level problem parameterized by the upper-level decision vector [116, 125, 90].

The following steps can be used to initialize a population with feasible solutions (approximated).

- **Step 1.** Generate $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N$ at random (uniform distribution) at the upper-level search space X.
- **Step 2.** For $i = 1, 2, \dots, N$, use $\boldsymbol{x}_i$ to optimize lower-level problem parametrized by $\boldsymbol{x}_i$, i.e., solve $\boldsymbol{y}_i^* \in \operatorname{argmin}\{f(\boldsymbol{x}_i,\ \boldsymbol{y}) : \boldsymbol{y} \in Y\}$ using the Lower Level Optimizer (Section 4.3.4).
- **Step 3.** Set the initial population as follows

$$P = \{(\boldsymbol{x}_1,\ \boldsymbol{y}_1^*),\ (\boldsymbol{x}_2,\ \boldsymbol{y}_2^*), \dots, (\boldsymbol{x}_N,\ \boldsymbol{y}_N^*)\}.$$

Note that **Step 1** is not complicated to perform, whilst **Step 2** can be the most computationally demanding because it depends on those features of the LL optimization problems. This computational effort is hard to avoid because the initial population needs to reflect a suitable starting solution to avoid converging into local optimal or infeasible solutions [116].

4.3.3 Upper-Level Optimizer

Once the initial population has been generated, new upper-level candidate solutions are computed to find promising solutions at the upper-level search space. Moreover, the upper-level optimizer (leader) needs to handle preferences that fit the problem application. For example, if the problem is on finding an optimistic or pessimistic solution. The proposed upper-level optimizer is detailed in the following steps:

- **Step 1.** *Generate UL candidate solutions:* Regarding the generation of UL candidate solutions, it is common to find that existent variation operators are used for that purpose. Genetic operators such as the SBX crossover and Polynomial Mutation [113], Differential Evolution operators, [7], flight equations in PSO [82], among others, have been used to generate candidate solutions at the upper level.

- **Step 2.** *Handle lower-level optimal solutions:* Here, the leader waits until the lower-level optimizer reports its results to the leader. The leader must be prepared if the follower reports a set of optimal solutions.

- **Step 3.** *Evaluate upper-level objective function:* Once the lower-level optimizer reports its optimal solution, the leader can evaluate the upper-level objective function.

- **Step 4.** *Environmental selection:* Once the approximated feasible solutions have been generated and the upper-level function values are available, this final step is used to save those solutions in the population that will be considered for the next generation.

It is worth mentioning that **Step 2** requires a high computation cost due to the resolution of the lower-level problems. A significant amount of research in evolutionary bilevel optimization is devoted to reduce the computational cost associated with **Step 2** by using sophisticated mechanisms for saving computational effort (see Section 3.1). In contrast, the remaining steps are not usually computationally demanding.

4.3.4 Lower-Level Optimizer

The lower-level algorithm, which must approximate lower-level optimal solutions for a given upper-level decision vector (say $\boldsymbol{x}$), is one of the most significant components. The parameterized problem to be solved at the lower level is defined as follows:

$$\boldsymbol{y}^* \in \operatorname{argmin}\{f(\boldsymbol{x},\, \boldsymbol{y}) \;:\; \boldsymbol{y} \in Y\}$$

Such an optimization problem is a single-level optimization problem that can be solved by classical or evolutionary approaches. The election of the optimizer is important due to the high precision required in some applications [116]. Some of the most popular algorithms for solving the lower-level instance are Differential Evolution [7], Genetic Algorithms [113], Particle Swarm Optimization [82] and Sequential Quadratic Programming [116].

4.4 Bilevel Centers Algorithm

Here, the Bilevel Centers Algorithm (BCA) is presented as an instance of the framework above mentioned. BCA will use the variation operator based on the center of mass concept (Section 4.2.1), and the lower-level optimizer will implement the Evolutionary Centers Algorithm [93].

BCA is based on the nested scheme described in Sections 3.1 and 4.3, i.e., an upper-level optimizer generates UL solutions but lets a lower-level optimizer find the corresponding lower-level optimal solution to create a feasible solution.

Algorithm 2 summarizes each component of the proposal and the next subsections are used to detail each one of them. As it can be noted, the search at both levels is based on the center of mass operator described in Section 4.2. BCA

Algorithm 2: BCA pseudocode

```
1  $N \leftarrow K * D$
2  Initialize population $P$ with $N$ elements (Section 4.3.2)
3  while the end criterion is not achieved do
4      for each $(\boldsymbol{x},\ \boldsymbol{y})$ in $P$ do
5          Generate a subset $U \subset P$ such that $|U| = K$
6          Calculate $\boldsymbol{c}$ using $U$ with Equation (4.9)
7          $\eta \leftarrow \text{rand}(0,\ \eta_{\max})$
8          Calculate $\boldsymbol{p}$ using Equation (4.8)
9          Find $\boldsymbol{q} \in \operatorname{argmin}_{\boldsymbol{z} \in Y} f(\boldsymbol{p},\ \boldsymbol{z})$ by using Algorithm 3
10         if $F(\boldsymbol{x},\ \boldsymbol{y}) < F(\boldsymbol{p},\ \boldsymbol{q})$ then
11             Replace the worst element in $P$ with $(\boldsymbol{p},\ \boldsymbol{q})$.
12     Resize $P$ with Equation (4.11).
13 Report the best solution in $P$
```

also incorporates a population size reduction to exploit regions of interest in the search space.

4.4.1 Upper-Level Variation Operator

New candidate solutions are generated for each solution $(\boldsymbol{x}_i,\ \boldsymbol{y}_i) \in P$, a new upper-level parameter is generated using Equation (4.8):

$$\boldsymbol{p}_i = \boldsymbol{x}_i + \eta_i(\boldsymbol{c}_i - \boldsymbol{u}_{\text{worst}}), \tag{4.8}$$

where $\boldsymbol{c}_i$ is the center of mass computed by Equation (4.9):

$$\boldsymbol{c}_i = \frac{1}{M} \sum_{(x,y) \in U} m(\boldsymbol{x},\ \boldsymbol{y}) \cdot \boldsymbol{x}, \qquad M = \sum_{(x,y) \in U} m(\boldsymbol{x},\ \boldsymbol{y}), \tag{4.9}$$

where $m(\boldsymbol{x}, \boldsymbol{y})$ is defined as in Equation (4.4) but replacing f in such expression by $Q(\boldsymbol{x},\ \boldsymbol{y}) = F(\boldsymbol{x},\ \boldsymbol{y}) + f(\boldsymbol{x},\ \boldsymbol{y})$, $U \subset P$ such that $|U| = K$ and $\boldsymbol{u}_{\text{worst}}$ is the first coordinate of the worst element in U, see Equation (4.10).

$$\boldsymbol{u}_{\text{worst}} \in \text{argmin}\{Q(\boldsymbol{u},\ \boldsymbol{y}) \ : \ \boldsymbol{u} \in U_i\} \tag{4.10}$$

Note that function $Q(\boldsymbol{x},\ \boldsymbol{y})$ is utilized to direct the upper-level population to areas where the upper-level objective function $F(\boldsymbol{x}, \boldsymbol{y})$ is maximized while the lower-level function $f(\boldsymbol{x}, \boldsymbol{y})$ will control the bias to feasible solutions.

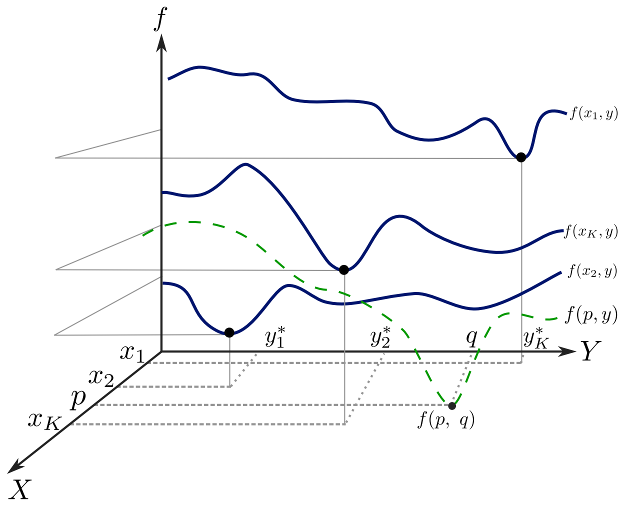


Figure 4.3: BCA diagram. Here, $x_1, \ldots, x_K$ are used to compute better upper-level parameters $\boldsymbol{p}$. Note that $(x_i,\ y_i^*)$ and $(\boldsymbol{p},\ \boldsymbol{q})$ represent feasible solutions.

Therefore, the new generated solution is determined by $(\boldsymbol{p},\ \boldsymbol{q})$ which may replace the worst solution in P if it is better than $(\boldsymbol{x}_i,\ \boldsymbol{y}_i)$. Here, the lower-level solution $\boldsymbol{q} = \text{argmin}_{\boldsymbol{z} \in Y} f(\boldsymbol{p},\ \boldsymbol{z})$ is obtained by applying the lower-level optimizer outlined by Algorithm 3. Figure 4.3 shows a representation of BCA solution update.

Due to this stochastic strategy, this variation operator can help the exploration-exploitation process to avoid premature convergence because it combines the center of mass as an attractor to promising regions of the search space but uses the position of the worst solution as a reference to get far away from it [90]. Taking this into account, the replacement operator works as follows: if $(\boldsymbol{p},\ \boldsymbol{q})$ is better than $(\boldsymbol{x}, \boldsymbol{y})$, then the worst element in P is replaced by $(\boldsymbol{p},\ \boldsymbol{q})$.

4.4.2 Solving the Lower Level Problem

In this part, the ECA algorithm [90] is used to approximate feasible solutions with a limited number of evaluations at the lower-level. The procedure for the implementation of the lower-level optimizer (ECA) is summarized in Algorithm 3.

Algorithm 3: Pseudocode Lower Level Optimizer ECA

```
1  Input: upper-level parameter $\boldsymbol{p}$, $K = 7,\ \eta_{\max} = 2$
2  $N \leftarrow K * D$
3  Initialize a population $P \subset Y$ with $N$ elements
4  while the end criterion is not achieved do
5      for each $\boldsymbol{y}$ in $P$ do
6          Generate a subset $U \subset P$ with $K$ solutions
7          Calculate $\boldsymbol{c}$ using $U$ with (4.1)
8          $\eta \leftarrow \text{rand}(0,\ \eta_{\max})$
9          Compute $\boldsymbol{q}$ using Equation (4.3)
10         if $f(\boldsymbol{p},\ \boldsymbol{y}) < f(\boldsymbol{p},\ \boldsymbol{q})$ then
11             Replace the worst element in $P$ with $\boldsymbol{q}$
12     Resize $P$ with Equation 4.11
13 Report the best solution in $P$
```

4.4.3 Adaptive Population Size Reduction

The population size is reduced linearly (the worse members are removed). The population size at the start is $N(0) = K * D$ and the final population size $N(T) = 2 * K$ (for successfully generating the center of mass). Thus, the population size over time is as in given by Equation 4.11:

$$N(t) = KD - \frac{(KD - 2K)t}{T} = K\left(D - \frac{(D-2)t}{T}\right), \tag{4.11}$$

where $t = 0, 1, 2, \ldots, T$ and T is the maximum number of iterations.

4.4.4 Parameters

BCA requires three input parameter values: stepsize $\eta_{\max}$, K, and the population size N. Large K values offer fast convergence to local optima, which is useful when dealing with multi-modality (useful for unimodal functions). Small values (respect to the population size) of K are in general preferred on multi-modal problems. As the step size is controlled by this option, $\eta_{\max}$ mostly governs the exploratory process. Preliminary investigations suggest using $K = 7$ and $\eta_{\max} = 2$ as parameter settings [94].

To evaluate the performance of BO algorithms, this proposed algorithm (Algorithm 2) was utilized to solve a set of eight test problems. Those problem instances were discussed in Chapter 2.

4.5 Experiments and Discussion

The following was the configuration used in BCA for the experiments. The limited number of function evaluations has been used as a stopping criterion for this preliminary proposal. For the upper level, the Number of Function Evaluations (NFEs) was set as $500D_{UL} = 2500$, and the lower level to $500D_{UL} * 500D_{LL}$, i.e., $6,250,000$ NFEs for both levels. The remaining parameters were taken from the suggested values in [90], but adapted for the corresponding decision variable at the upper- and lower-level.

- Upper-level dimension $D_{UL} = 5$
- Lower-level dimension $D_{LL} = 5$
- Upper-level population size $K * D_{UL}$
- Lower-level population size $K * D_{LL}$
- $K = 7$
- $\eta_{\max} = 2$

	Best	**Median**	**Mean**	**Worst**	**Std.**
SMD1	1.51E–05	5.35E–05	5.28E–05	8.34E–05	1.90E–05
SMD2	1.50E–05	4.68E–05	5.88E–05	3.11E–04	5.13E–05
SMD3	1.57E–05	5.51E–05	2.40E–04	3.54E–03	6.60E–04
SMD4	4.09E–08	4.90E–05	6.20E–05	3.01E–04	6.70E–05
SMD5	2.30E–05	5.03E–05	4.78E–05	8.33E–05	1.74E–05
SMD6	1.57E–01	1.90E+01	2.59E+01	1.40E+02	2.95E+01
SMD7	9.34E–01	9.75E–01	1.19E+00	3.81E+00	6.00E–01
SMD8	1.58E–05	6.49E–05	1.83E–04	2.23E–03	4.11E–04

Table 4.1: Upper-level accuracy statistics by BCA were obtained from 31 independent runs.

- Stop condition: BCA stopped when the accuracy (1×10^{-4}) or the maximum NFEs allowed were reached.

BCA was compared against a state-of-the-art evolutionary algorithm (BLEAQ) for BO problems [116, 113]. The BLEAQ algorithm is based on quadratic approximations of ideal lower-level variables concerning upper-level ones. As a result of this method, BLEAQ can reduce NFEs while maintaining competitive outcomes. The authors' proposed parameters were used in BLEAQ [113, 116] and the stopping criteria were maintained, but accuracy was set at 1×10^{-4} as in BCA. Both algorithms were employed to solve each test problem 31 times.

The statistical results of the accuracy values produced by the proposed BCA in the two levels of each BO test problem are presented in Tables 4.1 and 4.2, blackincluding the best, median, mean, worst, and standard deviation values of the obtained accuracy. In addition, Table 4.3 and 4.4 show the best, median, mean, worst, and standard deviation values of the NFEs required to solve each BO test problem using BCA. The median accuracy and NFEs at the upper and lower levels needed by BCA are compared to the values produced by BLEAQ in Tables 4.5 and 4.6. Moreover, the best value discovered is indicated by a boldfaced result.

	Best	Median	Mean	Worst	Std.
SMD1	2.67E–06	2.06E–05	2.14E–05	4.75E–05	9.77E–06
SMD2	3.56E–06	1.81E–05	2.07E–05	4.60E–05	1.15E–05
SMD3	1.87E–07	2.24E–05	3.12E–04	4.31E–03	9.28E–04
SMD4	6.23E–06	3.65E–05	1.53E–04	7.96E–04	2.33E–04
SMD5	2.50E–06	2.01E–05	2.19E–05	4.55E–05	1.28E–05
SMD6	3.91E–04	2.86E–02	4.23E–02	1.84E–01	4.35E–02
SMD7	3.71E+02	3.74E+02	3.74E+02	3.75E+02	1.01E+00
SMD8	1.18E–06	2.33E–05	6.53E–05	7.93E–04	1.44E–04

Table 4.2: Lower-level accuracy statistics by BCA were obtained from 31 independent runs.

	Best	Median	Mean	Worst	Std.
SMD1	1244	1526	1539.42	1879	152.119
SMD2	1244	1481	1482.84	2501	220.7
SMD3	1365	1526	1647.84	2501	287.517
SMD4	1169	1435	1437.26	1800	131.393
SMD5	1317	1526	1554.42	1898	137.766
SMD6	2501	2501	2501	2501	0
SMD7	2501	2501	2501	2501	0
SMD8	1780	2219	2234.65	2501	206.959

Table 4.3: Upper-level NFEs statistics by BCA were obtained from 31 independent runs.

BCA consistently achieves very competitive performance on all test instances at both levels, as can be seen in Tables 4.1 and 4.2. Only in problem SMD7 BCA showed difficulties to solve it. Therefore, BCA had a comparable strong performance in the upper level when it came to NFEs, as seen in Table 4.3. In contrast, more variation in the number of NFEs was reported in Table 4.4 for the lower level.

Regarding the comparison against the state-of-the-art algorithm, from Table 4.6, BCA outperformed BLEAQ in five BO test problems at the upper level and five BO test problems at the lower level.

Moreover, BCA successfully solved SMD1-SMD5 and SMD8 but could not reach the global optimum (at median value) for SMD6-SMD7 in which the upper and lower level have conflicting objectives, and both problems contain multiple local optimums in the upper-level search space. This indicates that BCA can be improved to handle multimodal problems, perhaps incorporating mechanisms to promote the exploitation of promising regions at the upper-level search space.

	Best	**Median**	**Mean**	**Worst**	**Std.**
SMD1	3110000	3815000	3848548.4	4697500	380298.6
SMD2	3110000	3702500	3707096.8	6252500	551750.7
SMD3	3412500	3815000	4119596.8	6252500	718793.3
SMD4	2922500	3587500	3593145.2	4500000	328481.3
SMD5	3292500	3815000	3886048.4	4745000	344415.4
SMD6	6252500	6252500	6252500	6252500	0
SMD7	6252500	6252500	6252500	6252500	0
SMD8	4450000	5547500	5586612.9	6252500	517397.6

Table 4.4: Lower-level NFEs statistics by BCA were obtained from 31 independent runs.

	Upper Level		Lower Level	
	BCA	BLEAQ	BCA	BLEAQ
SMD1	**1526**	1600	3815000	**116088**
SMD2	**1481**	1925	3702500	**113504**
SMD3	**1526**	1630	3815000	**122542**
SMD4	**1435**	1750	3587500	**70906**
SMD5	**1526**	3031	3815000	**147289**
SMD6	2501	**1016**	6252500	**7055**
SMD7	2501	**2104**	6252500	**130195**
SMD8	**2219**	5569	5547500	**289886**

Table 4.5: Median NFEs values by BCA and BLEAQ obtained from 31 independent runs.

	Upper Level		Lower Level	
	BCA	BLEAQ	BCA	BLEAQ
SMD1	**5.35E–05**	9.91E–05	**2.06E–05**	6.72E–05
SMD2	**4.68E–05**	2.82E–04	**1.81E–05**	3.84E–04
SMD3	5.51E–05	**4.96E–06**	2.24E–05	**6.26E–06**
SMD4	**4.90E–05**	1.54E–04	**3.65E–05**	6.12E–04
SMD5	**5.03E–05**	1.62E–04	**2.01E–05**	3.08E–04
SMD6	1.90E+01	**1.46E–13**	2.86E–02	**8.66E–16**
SMD7	9.75E–01	**9.76E–02**	3.74E+02	**1.25E+02**
SMD8	**6.49E–05**	7.46E–03	**2.33E–05**	5.63E–03

Table 4.6: Median accuracy values by BCA and BLEAQ obtained from 31 independent runs.

Furthermore, based on Table 4.5, BCA required fewer NFEs at the upper level of six BO test instances. However, BLEAQ outperformed BCA in the number of lower-level NFEs as indicated in the same Table 4.5.

4.6 Conclusions of the Chapter

The application of a physics-inspired method based on the center of mass (BCA) to handle bilevel optimization problems was reported in this chapter. To locate promising regions of the search space, both levels used a variation operator based on the center of mass and a greedy replacement based on fitness. BCA is a basic algorithm that just requires the user to fine-tune three parameters (the population size, the size of the subset to compute the center of mass, and the stepsize for the variation operator). Eight test problems were solved to compare the suggested approach to a state-of-the-art evolutionary algorithm for BO in terms of upper/lower level accuracy and function evaluations. Overall, the results suggest that BCA was able to produce comparable accuracy results to those produced by BLEAQ, despite requiring fewer evaluations at the upper level. In terms of the number of evaluations computed at the lower level, BLEAQ outperformed BCA.

Chapter 5

QBCA: BCA for Handling Pseudo-feasible Solutions

Information

Part of this chapter is based on the paper: Jesús-Adolfo Mejía-de-Dios, Efrén Mezura-Montes, and Porfirio Toledo-Hernández (2022). *Pseudo-feasible solutions in evolutionary bilevel optimization: Test problems and performance assessment*. Applied Mathematics and Computation, 412, 126577. `https://doi.org/10.1016/j.amc.2021.126577`

To deal with bilevel optimization problems, this chapter proposes a population-based metaheuristic technique named Quasi-Newton Bilevel Centers Algorithm (QBCA). At the lower level, a quasi-Newton approach is used to deal with infeasible solutions. Representative test functions for bilevel optimization are used to evaluate the performance of this approach (see [92]). The main mechanisms in QBCA are focused on the approximation of feasible solutions and the avoidance of pseudo-feasible solutions. QBCA has the following considerations:

- It adopts the first condition given in Section 2.5.2 to discard pseudo-feasible solutions. Firstly, it is essential to introduce as many feasible solutions as possible into the population (if exact feasible solutions cannot be generated, the usage of metaheuristic approaches is suggested to approximate them).
- Once the population has been generated, if the first condition is not satisfied, then $(\boldsymbol{x}, \boldsymbol{y})$ is not replaced by the new generated solution $(\boldsymbol{w}, \boldsymbol{z})$, because $(\boldsymbol{w}, \boldsymbol{z})$ can be a pseudo-feasible solution and must be discarded.
- Individuals in population $P \subset X \times Y$ should be reevaluated by performing improvement procedures in a solution $\boldsymbol{x}$ to approximate the corresponding $\boldsymbol{y} \in \Psi(x)$ by a global search.

Taking into account the above considerations, the following section is used to detail the baseline methodology to adapt BCA for handling pseudo-feasible solutions.

5.1 Baseline Solution Methodology

As a first step, the unconstrained BO problem $(F,\ f,\ X,\ Y,\ \mathbb{R})$ is considered, and a population with N feasible solutions is given by: $P = \{(\boldsymbol{x}_i,\ \boldsymbol{y}_i) \in X \times Y\ :\ \boldsymbol{x}_i \in X,\ \boldsymbol{y}_i \in Y,\ i = 1, \dots, N\}$, where each $\boldsymbol{x}_i \in X$ has a corresponding $\boldsymbol{y}_i \approx \boldsymbol{y}(\boldsymbol{x_i}) \in \Psi(\boldsymbol{x}_i)$, which is an approximation of a feasible solution. Note that $(\boldsymbol{x}_i,\ \boldsymbol{y}_j)$ for $i \neq j$ could not be a feasible solution.

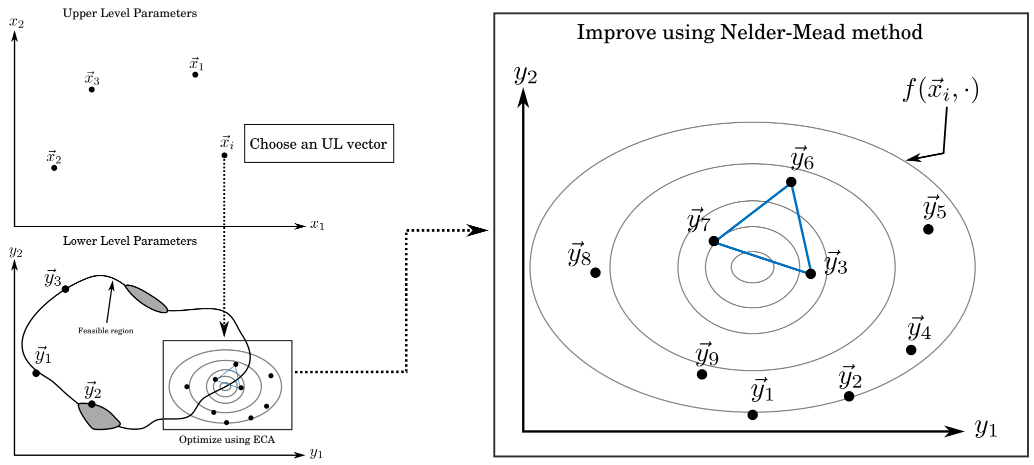


Figure 5.1: Initialization of the QBCA approach. First, an upper-level vector is randomly generated, then the lower-level optimizer (ECA) is used to approximate the corresponding LL solution which is later improved using Nelder-Mead method.

5.1.1 Initial Population

The QBCA method should start with K feasible solutions, i.e., a completely feasible population. For such a task, an EA called ECA [90] is used to approximate feasible solutions with a limited number of function evaluations at the lower level. After that, the result is refined by using a heuristic method.

The QBCA population initialization is represented in Figure 5.1 and summarized in the following steps.

- **Step 1.** Set the initial population $P = \emptyset$.
- **Step 2.** Generate N vectors $\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\}$ uniformly at random in the UL search space X.
- **Step 3.** Randomly choose K ($K < N$) UL vectors to generate the set $U = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_K\} \subset P$,
- **Step 4.** For each element $\boldsymbol{x} \in U$, use ECA (limited to $1000D_{ll}$ LL function evaluations) to solve problem $\min\{f(\boldsymbol{x}, \boldsymbol{y}) : \boldsymbol{y} \in Y\}$ and choose the best

$D_{ll} + 1$ solutions.

- **Step 5.** Improve each solution by using the above $D_{ll} + 1$ results and the Nelder-Mead method with adaptive parameters [52] (with $\alpha = 1$ for the reflection, $\beta = 1 + 2/D_{ll}$ for the expansion, $\gamma = 0.75 - 1/(2D_{ll})$ for the contraction, and $\delta = 1 - 1/D_{ll}$ for the shrink step) to generate the approximate K solutions $(\boldsymbol{x}, \boldsymbol{y})$. Details of the Nelder-Mead method are found in Algorithm 4. The main idea is to improve each $\boldsymbol{x_i}$ by calling the Nelder-Mead method to minimize $f_p(\cdot) = f(\boldsymbol{x}_i, \cdot)$ for $1 \leq i \leq K$. Nelder-Mead method stopped when the mean distance between the current solution and the centroid was less than 1×10^{-8}.

- **Step 6.** For each $N - K$ remaining UL solutions, apply the lower-level optimizer defined in Section 5.1.3.

- **Step 7.** Evaluate the solutions generated in steps 4-6 and insert them in P.

Algorithm 4: Nelder-Mead algorithm with adaptive parameters $\alpha = 1$, $\beta = 1 + 2/Dll$, $\gamma = 0.75 - 1/(2D_{ll})$ and $\delta = 1 - 1/D_{ll}$.

```
1  Input. An objective function $f_p$ and a simplex $\Delta = \{\boldsymbol{x_1},\ \boldsymbol{y}_2, \dots, \boldsymbol{y}_{D_{ll}}\}$.
2  while the end criterion is not achieved do
3      Put $\boldsymbol{y}_c \leftarrow \frac{1}{D_{ll}} \sum_{i=1}^{D_{ll}} \boldsymbol{y}_i$ be the centroid of the best $D_{ll}$ solutions.
4      Sort. Evaluate $f_p$ at solutions of $\Delta$ and sort them such that
         $f_p(\boldsymbol{y}_1) \leq f_p(\boldsymbol{y}_2) \leq \cdots \leq f_p(x_{D_{ll}+1})$.
5      Reflection. Calculate the reflection point $\boldsymbol{y}_r \leftarrow \boldsymbol{y}_c + \alpha(\boldsymbol{y}_c - \boldsymbol{y}_{D_{ll}+1})$.
6      Expansion. If $f_p(\boldsymbol{y}_r) < f_p(\boldsymbol{y}_1)$ then compute the expansion point
         $\boldsymbol{y}_e = \boldsymbol{y}_c + \beta(\boldsymbol{y}_r - \boldsymbol{y}_c)$. If $f_p(\boldsymbol{y}_e) < f_p(\boldsymbol{y}_r)$, replace $\boldsymbol{y}_{D_{ll}+1}$ with $\boldsymbol{y}_e$ ;
         otherwise replace $\boldsymbol{y}_{D_{ll}+1}$ with $\boldsymbol{y}_r$.
7      Outside Contraction. If $f_p(\boldsymbol{y}_n) \leq f_p(\boldsymbol{y}_r) < f_p(\boldsymbol{y}_{D_{ll}+1})$, compute the
         outside contraction point $\boldsymbol{y}_{oc} \leftarrow \boldsymbol{y}_c + \gamma(\boldsymbol{y}_r - \boldsymbol{y}_c)$. If $f_p(\boldsymbol{y}_{oc}) \leq f_p(\boldsymbol{y}_r)$,
         replace $\boldsymbol{y}_{D_{ll}+1}$ with $\boldsymbol{y}_{oc}$ ; otherwise go to step 9.
8      Inside contraction. If $f_p(\boldsymbol{y}_r) \geq f_p(\boldsymbol{y}_{D_{ll}+1})$, compute the inside
         contraction point $\boldsymbol{y}_{ic} \leftarrow \boldsymbol{y}_c - \gamma(\boldsymbol{y}_r - \boldsymbol{y}_c)$. If $f_p(\boldsymbol{y}_{ic}) < f_p(\boldsymbol{y}_{D_{ll}+1})$,
         replace $\boldsymbol{y}_{D_{ll}+1}$ with $\boldsymbol{y}_{ic}$ ; otherwise, go to step 9.
9      Shrink. For $2 \leq i \leq D_{ll} + 1$, translate $\boldsymbol{y}_i \leftarrow \boldsymbol{y}_1 + \delta(\boldsymbol{y}_i - \boldsymbol{y}_1)$.
10 Return $\boldsymbol{y}_1$
```

5.1.2 Upper-Level Optimizer

Once the initial population is available, new UL decision vectors in X are generated by a variation operator inspired in the center of mass and presented in [90, 92]. To compute the center of mass in a minimization problem, the following strategy is adopted: $m_F(\boldsymbol{x}, \boldsymbol{y}) = 2\left[\max_{(\boldsymbol{u},\boldsymbol{v})\in U} |F(\boldsymbol{u}, \boldsymbol{v})|\right] - F(\boldsymbol{x}, \boldsymbol{y})$. where $U \subset P$ is generated by choosing its elements uniformly at random. Here, m_F defines a non-negative function on a finite set U such that small values of $F(\boldsymbol{x}, \boldsymbol{y})$ represent larger values of $m_F(\boldsymbol{x}, \boldsymbol{y})$.

If it is the case that in both levels there are maximization problems with non-negative objective functions (assumptions of ECA [90]), then $m_F(\boldsymbol{x}, \boldsymbol{y}) = F(\boldsymbol{x}, \boldsymbol{y})$.

$$\boldsymbol{c}_X = \frac{1}{W} \sum_{(\boldsymbol{x},\boldsymbol{y})\in U} m_F(\boldsymbol{x},\ \boldsymbol{y}) \cdot \boldsymbol{x}, \qquad W = \sum_{(\boldsymbol{x},\boldsymbol{y})\in U} m_F(\boldsymbol{x},\ \boldsymbol{y}), \tag{5.1}$$

Note that, c_X is biased toward smaller F values with respect to U. According to that behavior, a new UL vector is generated as in (5.2):

$$\boldsymbol{p} = \boldsymbol{x} + \eta_X(\boldsymbol{c}_X - \boldsymbol{u}_{\text{worst}}), \tag{5.2}$$

where $\eta_X \in (0, \eta_{\max}]$ and $\eta_{\max}$ is the user-defined stepsize, and $(\boldsymbol{u}_{\text{worst}}, \boldsymbol{y}) \in \arg\max\{F(\boldsymbol{x}, \boldsymbol{y}) \ : \ (\boldsymbol{x}, \boldsymbol{y}) \in U\}$. This new vector $\boldsymbol{p}$ is indeed generated by a *current-to-center-like* direction (refer to [90, 92]). It can be noted that $\boldsymbol{p}$ represents a new candidate solution at the UL. However, it is necessary to approximate the corresponding $\boldsymbol{q}^* \in \Psi(\boldsymbol{p})$, then the next section is devoted to the approximation of this feasible solution.

5.1.3 Lower Level Optimizer

The LL optimizer uses the population $P = \{(\boldsymbol{x}_i, \boldsymbol{y}_i) : i = 1, 2, \ldots, N\}$. Considering the information provided by the current distribution of feasible solutions in P, new vectors $\boldsymbol{q}_c$ are produced using the center of the mass variation operator. The modified vector is then used as the initial solution of a local optimizer, such as the Quasi-Newton method in our context.

The center of mass for the LL is defined on $V \subset P$ in (5.3):

$$\boldsymbol{y}_c = \frac{1}{W} \sum_{(\boldsymbol{x},\boldsymbol{y})\in V} m_f(\boldsymbol{x},\ \boldsymbol{y}) \cdot \boldsymbol{y}, \text{ with } W = \sum_{(\boldsymbol{x},\boldsymbol{y})\in V} m_f(\boldsymbol{x},\ \boldsymbol{y}), \tag{5.3}$$

where each mass value is computed as usual:

$$m_f(\boldsymbol{x},\boldsymbol{y}) = 2\left[\max_{(\boldsymbol{u},\boldsymbol{v})\in V} |f(\boldsymbol{u},\boldsymbol{v})|\right] - f(\boldsymbol{x},\boldsymbol{y}).$$

As can be seen, (5.3) may be an infeasible solution. A Quasi-Newton approach is then utilized to move such solution near the feasible region.

When the objective function is smooth enough, this mathematical program-

Algorithm 5: BFGS-LL: Quasi-Newton method for the lower-level problem. This method finds a local optima of the lower-level function $f_p(\cdot) = f(\boldsymbol{p}, \cdot)$ given the upper-level solution $\boldsymbol{p} \in X$.

```
1  Input. Initial solution $\boldsymbol{y}_c \in Y$, upper-level solution $\boldsymbol{p} \in X$, objective
     function $f_p(\boldsymbol{y}) = f(\boldsymbol{p}, \boldsymbol{y})$.
2  Set $H_0$ to a positive definite matrix (identity matrix $I$)
3  Put $\boldsymbol{q}_0 \leftarrow \boldsymbol{y}_c$ and $k \leftarrow 0$
4  while the end criterion is not met do
5      $\boldsymbol{h}_k \leftarrow -H_k \nabla f_p(\boldsymbol{q}_k)$
6      Compute $\gamma_k \leftarrow \arg\min f_p(\ \boldsymbol{q}_k + \gamma_k \boldsymbol{h}_k)$.
7      $\boldsymbol{q}_{k+1} \leftarrow \boldsymbol{q}_k + \gamma_k \boldsymbol{h}_k$
8      if $f$ is not differentiable at $(\boldsymbol{p},\ \boldsymbol{q}_{k+1})$ then
9          Stop
10     $z_k \leftarrow \nabla f_p(\ \boldsymbol{q}_{k+1}) - \nabla f_p(\ \boldsymbol{q}_k)$
11     $A_k \leftarrow I - (\boldsymbol{h}_k^T \boldsymbol{z}_k)^{-1} \boldsymbol{h}_k \boldsymbol{z}_k^T$
12     $H_{k+1} \leftarrow A_k H_k A_k^T + \gamma_k (\boldsymbol{h}_k^T \boldsymbol{z}_k)^{-1} \boldsymbol{h}_k \boldsymbol{h}_k^T$ which is a positive definite
         matrix and satisfies the secant condition $H_{k+1} z_k = \gamma_k \boldsymbol{h}_k$
13     $k \leftarrow k + 1$
14
15 Return $\boldsymbol{q}_k$
```

ming methodology (Quasi-Newton method) is extensively employed for numerical optimization, and empirical results imply that it can be useful in practice [49, 78, 84]. Here, the proposal utilizes a modified Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm, denoted as BFGS-LL, detailed in [92] and included in Algorithm 5. The update consists in using $\boldsymbol{y}_c$ as an initial point to the classical BFGS to minimize function $f(\boldsymbol{p}, \cdot)$ by approximating local gradients. The resulting optimal solution is therefore denoted as $\boldsymbol{q}$. BFGS-LL stopped when the maximum number of lower-level functions evaluations (1000) was reached or $\|\nabla f_p(\boldsymbol{q_k})\| \leq 10^{-8}$ or $\|f_p(\boldsymbol{q}_k) - f_p(\boldsymbol{q}_{k-1})\| = 0$ or $\|\boldsymbol{q}_k - \boldsymbol{q}_{k-1}\| = 0$.

5.1.4 Mechanism to Detect Pseudo-feasible Solutions

Here, the pseudo-feasible solution detection mechanism is implemented as follows: a solution $(\boldsymbol{p}, \boldsymbol{q})$ is better than $(\boldsymbol{x}, \boldsymbol{y})$ if the first condition in Section 2.5.2 is not satisfied and $F(\boldsymbol{p}, \boldsymbol{q}) < F(\boldsymbol{x}, \boldsymbol{y})$ (see line 13 in Algorithm 6). That is, $(\boldsymbol{p}, \boldsymbol{q})$ is NOT better than $(\boldsymbol{x}, \boldsymbol{y})$ if $F(\boldsymbol{p}, \boldsymbol{q}) \geq F(\boldsymbol{x}, \boldsymbol{y})$ or the following (Condition 1) is satisfied:

$$\begin{cases} \|\boldsymbol{x} - \boldsymbol{p}\| < \delta_1 \text{ and} \\ \|\boldsymbol{y} - \boldsymbol{q}\| \geq \delta_2 \text{ and} \\ |F(\boldsymbol{x}, \boldsymbol{y}) - F(\boldsymbol{p}, \boldsymbol{q})| < \varepsilon_1 \text{ and} \\ |f(\boldsymbol{x}, \boldsymbol{y}) - f(\boldsymbol{p}, \boldsymbol{q})| < \varepsilon_2. \end{cases} \tag{5.4}$$

where $\delta_1,\ \delta_2,\ \varepsilon_1$ and ε_2 are parameters given by the user. For simplicity (and applicability purposes) such values can be fixed within $(0, 1]$.

Finally, the complete algorithm to solve BO problems is summarized in Algorithm 6. To avoid convergence to local optima, QBCA executes ECA (limited to $1000D_{ll}$ LL function evaluations) on all solutions in the population every 10 iterations to update them and improve the search by applying the Nelder-Mead method as in Section 5.1.1. The stopping criteria implemented for Algorithm 6 is related to the maximum-number of upper-level function evaluations (see Section 5.3).

Algorithm 6: QBCA pseudocode. The suggested parameter values are $K = 3$, $\eta_{\max} = 2$, $\delta_1 = \delta_2 = 10^{-2}$.

```
1  Choose $K$, $\eta_{\max}$, $\delta_1$ and $\delta_2$.
2  Set $N \leftarrow K * D$ and $t \leftarrow 0$
3  Initialize population $P \subset X \times Y$ with $N$ elements
4  while the end criterion is not achieved do
5      $t \leftarrow t + 1$ for each $(\boldsymbol{x},\ \boldsymbol{y})$ in $P$ do
6          Generate $U \subset P$ and $V \subset P$ with $|U| = |V| = K$
7          Choose $\eta_X$ and $\eta_Y$
8          Compute $\boldsymbol{c_X}$ using $U$
9          $\boldsymbol{p} \leftarrow \boldsymbol{x} + \eta_X(\boldsymbol{c}_X - \boldsymbol{u}_{\text{worst}})$
10         Calculate $\boldsymbol{y}_c$ using $V$ with (5.3)
11         $\boldsymbol{q}$ is calculated using $\boldsymbol{y}_c$ and the BFGS-LL method
12         if $(\boldsymbol{p},\ \boldsymbol{q})$ is better than $(\boldsymbol{x},\ \boldsymbol{y})$ then
13             Replace the worst element in $P$ with $(\boldsymbol{p},\ \boldsymbol{q})$.
14     if $t \mod 10 = 0$ then
15         Reevaluate the best solution $(\boldsymbol{w}_{\text{best}}, \boldsymbol{z}_{\text{best}})$ in $P$ to obtain
           $(\boldsymbol{x}_{\text{best}}, \boldsymbol{y}_{\text{best}})$.
16         if $(\boldsymbol{w}_{best}, \boldsymbol{z}_{best})$ and $(\boldsymbol{x}_{best}, \boldsymbol{y}_{best})$ satisfies the rule 1 in Section 2.5.2
           then
17             Reevaluate the entire population $P$
18 Report the best solution in $P$
```

5.2 Computational Complexity

Regarding the computational complexity of QBCA (Algorithm 6), it is difficult to determine the complexity due to the different mechanisms implemented in QBCA, which have not yet been studied regarding the cost. However, it can be approximated this calculation in the worst case. The upper-level variation operator (lines 6-10 in Algorithm 6) requires $O(ND)$, the BFGS-LL (line 11 in Algorithm 6) requires $O(D_{ll}^2)$ computations [78], where D_{ll} is the number of lower-level variables. Assuming that the lower-level optimizer (lines 14-17 in Algorithm 6) requires $O(N_{ll}D_{ll}, T_{ll})$ computations, where N_{ll} is the lower-level population size, and T_{ll} is the number of lower-level iterations. Therefore, the overall complexity of one iteration of Algorithm 6 is $O(N_{ll}D_{ll}, T_{ll} + D_{ll}^2 + ND)$.

5.3 Experiments and Discussion

This section contains two studies for the QBCA performance assessment. Two *state-of-the-art* algorithms, BLCMAES and BLEAQ-2, are considered in this chapter since their empirical results demonstrate that they are effective to solve different test problems [56, 111]. The main goal of these experiments is to answer the following question: are both *state-of-the-art* algorithms reporting misleading solutions? The QBCA ability to detect pseudo-feasible solutions is also analyzed.

BLCMAES implements a CMA-ES algorithm at each level to transfer knowledge from an upper-level CMA-ES to a lower-level CMA-ES in a hierarchical scheme, and it approximates a distribution sharing mechanism. BLEAQ-2, on the other hand, is an EA for solving BO problems and can use quadratic functions to approximate the mapping Ψ and save lower-level function evaluations [114, 113].

Both the scalable SMD test suite and the PMM test functions were used to test QBCA, BLEAQ-2, and BLCMAES. The SMD test functions were recently presented to provide controlled scaling of challenges (such as multimodality and nonconvexity) at both levels. Because the lower-level objective functions are low-bounded and such a minimum is attained if and only if the solution is feasible, the SMD test problems do not offer pseudo-feasible solutions associated with the optimal feasible solution.

After that, a comparison with QBCA is conducted to assess the success of the rules for avoiding pseudo-feasible solutions, i.e., QBCA should not report misleading solutions. The first experiment evaluates the performance of the three algorithms mentioned above on SMD test issues, while the second exposes the performance of the two *state-of-the-art* evolutionary algorithms on the PMM test problems.

Algorithm 6 has been implemented in the Julia Programming Language [16], and the codes of BLCMAES and BLEAQ-2 were downloaded from the authors' websites. Also, The maximum number of function evaluations and the number of iterations are limited to 10^6. BFGS-LL terminates if $\|\nabla f_p(\boldsymbol{q_k})\| \leq 10^{-8}$ or $\|f_p(\boldsymbol{q}_k) - f_p(\boldsymbol{q}_{k-1})\| = 0$ or $\|\boldsymbol{q}_k - \boldsymbol{q}_{k-1}\| = 0$.

5.3.1 Experiment 1

Here, low and high-dimensional versions of the SMD test suite are considered. The low-dimensional problems consist of 5 variables where $D_{ul} = 2$ and $D_{ll} = 3$. The high-dimensional SMD instances consider 10-dimensional problems where $D_{ul} = 5$ and $D_{ll} = 5$. The description for the SMD test suite can be found in Appendix C.

The parameters adopted for QBCA were $K = 3$, the stepsize parameters $\eta_X \in (0, \eta_{\max}]$ with $\eta_{\max} = 2$ and $\delta_1 = \delta_2 = 10^{-2}$. It is worth mentioning that K and $\eta_{\max}$ values were found by the automated parameter tuning strategy developed in [94] and detailed in Chapter 6, whilst δ_1, δ_2 are tolerance values related to the desired accuracy. QBCA stopped if the desired accuracy was obtained or the maximum number of functions evaluations (NFEs) for the upper level was reached. BLCMAES and BLEAQ-2 adopted the settings provided by their authors

Instance	QBCA	BLCMAES	BLEAQ-2	QBCA	BLCMAES	BLEAQ-2
SMD1	100%	100%	100%	100%	100%	100%
SMD2	100%	100%	100%	100%	100%	100%
SMD3	100%	100%	100%	100%	100%	100%
SMD4	100%	100%	100%	100%	100%	100%
SMD5	100%	100%	100%	100%	100%	100%
SMD6	71%	**100%**	**100%**	3%	**100%**	**100%**
SMD7	97%	58%	**100%**	81%	6%	**100%**
SMD8	100%	100%	100%	**100%**	87%	**100%**

Table 5.1: QBCA, BLCMAES, and BLEAQ-2 Success rates when executed on 10-variable SMD test problems.

[111, 56]. Those settings were fixed for all experiments.

The desired accuracy was fixed to $\varepsilon_1 = 1\times10^{-2}$ and $\varepsilon_2 = 1\times10^{-3}$ for the upper and lower level, respectively. The NFEs were 2500 for low-dimensional functions and 3500 for high-dimensional instances. Each algorithm limited the lower-level function evaluations according to its mechanisms.

The success rates of QBCA, BLCMAES, and BLEAQ-2 in terms of intended accuracy at the upper and lower levels are shown in Table 5.1. It should be noted that QBCA performs similarly to BLEAQ-2 in most circumstances, while BLEAQ-2 outperforms QBCA in SMD6 and SMD7 cases, and QBCA outperforms BLCMAES in most cases. In this context, a successful run is one in which at least one feasible solution is found in NFEs.

5.3.2 Experiment 2

This experiment is carried out to show how unfeasible solutions might be mistakenly considered as feasible optimal solutions supplied by *state-of-the-art* EAs. Low and high-dimensional PMM test problems are considered in this study, i.e., the low-dimensional problems consider $D_{ul} = 2,\ D_{ll} = 3$ and $D_{ul} = 5,\ D_{ll} = 5$ for the high-dimensional problems. The following statement is used to test if a solution is near to a feasible optima: the solution (x, y) is close to the feasible optimal solution if $\|x - x^*\| < \delta_1$ and $\|y - y^*\| < \delta_2$ with $y^* \in \Psi(x)$. For simplicity $\delta_1 = \delta_2 = 2 \times 10^{-1}$ was considered and fixed for all experiments.

Here, a misleading run occurs when the algorithm, at the end, reports a solution that satisfies: $\|x - x^*\| < \delta_1$, $\|y - y^*\| \geq \delta_2$, $|F(x, y) - F(x^*, y^*)| < \varepsilon_1$ and $|f(x, y) - f(x^*, y^*)| < \varepsilon_2$. A Non-Convergent (NC) run is when the algorithm was unable to find a solution with the desired accuracy in NFEs, i.e., $|F(x, y) - F(x^*, y^*)| \geq \varepsilon_1$ or $|f(x, y) - f(x^*, y^*)| \geq \varepsilon_2$. A successful run, on the other hand, is obtained when the algorithm reports a correct and feasible solution. Similarly to the other measures, the successful rate is the number of successful runs divided by the total number of runs.

Algorithm	Instance	Misleading	NC	Success
QBCA	PMM1	0	0	100
	PMM2	0	0	100
	PMM3	0	0	100
	PMM4	0	3	97
	PMM5	3	0	97
	PMM6	6	0	94
BLCMAES	PMM1	0	0	100
	PMM2	0	0	100
	PMM3	0	0	100
	PMM4	16	0	84
	PMM5	0	0	100
	PMM6	29	61	10
BLEAQ-2	PMM1	23	74	3
	PMM2	0	0	100
	PMM3	13	6	81
	PMM4	10	26	64
	PMM5	6	3	91
	PMM6	19	61	20

Table 5.2: Results of 31 independent runs of QBCA and BLEAQ-2 on low-dimensional PMM test functions.

Tables 5.2 and 5.3 present the measured results (misleading runs, non-convergent runs, and successful rate) of 31 independent runs by each compared algorithm for the low and high-dimensional PMM test problems, respectively. Moreover, Tables 5.4 and 5.5 detail the error statistics and the number of functions evaluations for low and high-dimensional PMM test problems.

As can be seen in Tables 5.4 and 5.5, accuracy should be evaluated in terms of bilevel values and their corresponding vectors of parameters at both levels to avoid unfair comparisons and produce reliable findings.

It is also worth mentioning that BLCMAES produced misleading outcomes in both low and high instances; for example, Table 5.3 shows that BLCMAES converged to pseudo-feasible solutions in PMM4 test problem in several runs. According to the empirical findings, BLCMAES is unsuitable for multimodal LL scenarios. Furthermore, based on Table 5.2, BLEAQ-2 reports an important amount of pseudo-feasible solutions in low-dimensional test problems and is unable to converge in high-dimensional PMM problems (Table 5.3). QBCA, on the other hand, avoided pseudo-feasible solutions in most low-dimensional PMM instances and was able to solve high-dimensional PMM1-PMM5 test problems.

QBCA is able to successfully resolve instances even in the presence of misleading solutions, as the results suggest. The benefits and strengths of our approach are described below according to the characteristics of each function:

- **UL-Unimodal/LL-Unimodal:** These problems are not that difficult to solve (as suggested by the results) because there is only one optimal solution at both levels. For this reason, QBCA resolved the PMM1-PMM2 instances.
- **UL-Unimodal/LL-Multimodal:** Bilevel evolutionary algorithms can converge to local optima when one of the two levels presents some multimodality. When the lower level contains multiple local optima, then the lower level can produce infeasible solutions regarding the bilevel problem. For this reason, the evolutionary algorithms with which QBCA was compared reported misleading solutions or converged to infeasible solutions. Even some QBCA runs did not report convergence to the global optimum.
- **UL-Multimodal/LL-Unimodal:** If the lower level is unimodal, but the upper level contains multiple local optima, then the algorithms can easily converge to a feasible local optimum. For this reason, QBCA (and the rest of the bilevel algorithms) did not report optimal feasible solutions in the high dimensional PMM6 instance due to the strong multimodality. Moreover,

BLEAQ-II and BL-CMEAES also failed to solve PMM3-PMM6 instances. A mutation with a higher degree of disruption can be provided for the upper level to improve the performance of the algorithms.

- **UL-Multimodal:/UL-Multimodal:** One of the most difficult bilevel problems to solve is those that present multi-modality at both levels, and it can be even more difficult if both levels conflict. For this type of instance (PMM6), the algorithms must handle the upper level's optimality and the feasibility obtained when the lower level is solved. Note that QBCA can solve this type of instance for low-dimensional problems; however, for high-dimensional problems (PMM6), QBCA loses the ability to report optimal solutions, although it does not report misleading solutions, which can be useful for those seeking feasibility.

5.4 Conclusions of the Chapter

In this chapter, to prevent misleading results caused by pseudo-feasible solutions, a baseline approach with the rules presented in Chapter 2 was provided. The results obtained in the experiments revealed that two *state-of-the-art* algorithms are subject to report pseudo-feasible solutions. On the other hand, QBCA coupled with the pseudo-fesible solution detection mechanism was able to report highly competitive results while avoiding pseudo-feasible solutions.

	Instance	Misleading	NC	Success
QBCA	PMM1	0	0	100
	PMM2	0	0	100
	PMM3	0	0	100
	PMM4	0	0	100
	PMM5	0	0	100
	PMM6	0	100	0
BLCMAES	PMM1	0	13	87
	PMM2	0	100	0
	PMM3	0	0	100
	PMM4	58	6	36
	PMM5	32	0	68
	PMM6	23	77	0
BLEAQ-2	PMM1	3	94	3
	PMM2	0	100	0
	PMM3	6	87	7
	PMM4	0	100	0
	PMM5	0	97	3
	PMM6	0	100	0

Table 5.3: Results of 31 independent runs of QBCA and BLEAQ-2 on high-dimensional PMM test functions.

	QBCA				BLCMAES				BLEAQ-2			
	$\|x-x^*\|$		$\|y-y^*\|$		$\|x-x^*\|$		$\|y-y^*\|$		$\|x-x^*\|$		$\|y-y^*\|$	
	Median	Std.	Median	Std.	Median	Std.	Median	Std.	Median	Std.	Median	Std.
PMM1	2.07E-02	7.08E-03	1.25E-06	1.38E-04	1.04E-03	7.05E-04	3.42E-04	3.83E-04	5.71E-01	6.95E-01	2.76E-01	2.06E+00
PMM2	1.95E-02	7.88E-03	1.88E-09	4.14E-09	8.41E-04	3.04E-04	5.52E-04	3.53E-04	5.53E-04	8.83E-03	2.36E-06	8.93E-05
PMM3	2.22E-02	7.13E-03	1.30E-08	4.63E-08	1.06E-03	7.68E-04	1.24E-03	1.10E-03	2.60E-02	7.42E-02	4.97E-02	1.78E-01
PMM4	2.30E-02	1.85E-02	6.77E-07	9.54E-04	1.07E-03	4.09E-01	4.94E-04	4.08E-01	3.43E-02	6.22E-01	8.60E-03	6.27E-01
PMM5	2.36E-02	1.75E-01	1.22E-06	1.78E-01	1.11E-03	7.69E-04	5.36E-03	4.67E-03	3.34E-02	1.76E-01	3.76E-02	1.83E-01
PMM6	1.54E-02	1.20E-01	6.60E-10	2.47E-01	9.95E-01	3.91E-01	6.25E-01	4.32E-01	3.25E-01	4.37E-01	6.30E-01	7.20E-01

	QBCA				BLCMAES				BLEAQ-2			
	UL		LL		UL		LL		UL		LL	
	Median	Std.	Median	Std.	Median	Std.	Median	Std.	Median	Std.	Median	Std.
PMM1	2.04E+02	3.71E+01	8.15E+04	1.37E+04	1.18E+03	9.24E+01	6.80E+04	6.93E+03	1.06E+03	3.90E+02	8.36E+03	2.10E+03
PMM2	2.15E+02	4.21E+01	2.52E+04	2.68E+03	3.11E+02	3.37E+01	2.18E+04	1.76E+03	4.12E+02	2.50E+02	4.47E+04	4.77E+02
PMM3	2.13E+02	4.79E+01	2.64E+04	3.06E+03	4.23E+02	2.88E+01	2.35E+04	1.48E+03	7.45E+02	4.17E+02	9.33E+04	1.17E+04
PMM4	2.05E+02	5.97E+01	3.15E+04	1.85E+04	5.64E+02	2.88E+02	3.88E+04	1.19E+04	8.78E+02	3.24E+02	8.51E+04	1.21E+04
PMM5	3.12E+02	8.49E+01	4.75E+04	2.27E+04	4.36E+02	4.45E+01	2.43E+04	2.32E+03	5.63E+02	1.83E+02	7.01E+04	7.52E+03
PMM6	6.00E+02	1.99E+02	1.82E+05	7.00E+04	1.04E+03	2.69E+02	5.21E+04	8.78E+03	1.08E+03	4.56E+02	8.33E+04	1.17E+04

Table 5.4: UL-LL error statistics were obtained from 31 independent runs of QBCA, BLCMAES, and BLEAQ-2 in low-dimensional PMM test problems. UL-LL function evaluations in low-dimensional PMM test problems.

	QBCA				BLCMAES				BLEAQ-2			
	$\|x-x^*\|$		$\|y-y^*\|$		$\|x-x^*\|$		$\|y-y^*\|$		$\|x-x^*\|$		$\|y-y^*\|$	
	Median	Std.	Median	Std.	Median	Std.	Median	Std.	Median	Std.	Median	Std.
PMM1	2.73E-02	4.70E-03	4.71E-07	1.42E-05	6.74E-03	1.93E+00	3.81E-03	2.96E+00	8.56E-01	6.71E-01	7.32E-01	2.12E+00
PMM2	2.29E-02	5.49E-03	8.34E-06	1.55E-04	8.82E+00	3.32E+00	6.18E+00	3.03E+00	1.39E+01	3.38E+00	1.53E+01	4.61E+00
PMM3	2.75E-02	4.35E-03	1.97E-06	3.35E-06	1.47E-03	7.21E-04	1.41E-03	8.36E-04	3.61E-01	5.96E-01	5.83E-01	1.41E+00
PMM4	2.68E-02	6.13E-03	1.24E-07	1.57E-04	9.97E-01	6.31E-01	9.95E-01	5.92E-01	1.79E+00	8.63E-01	1.70E+00	8.51E-01
PMM5	2.82E-02	3.50E-03	1.35E-03	1.47E-03	3.14E-03	4.97E-01	4.41E-02	4.81E-01	1.41E+00	6.52E-01	1.21E+00	6.43E-01
PMM6	1.19E+00	4.61E-01	1.59E+00	6.98E-01	1.43E+00	4.14E-01	1.07E+00	4.98E-01	2.22E+00	7.70E-01	1.61E+00	1.05E+00

	QBCA				BLCMAES				BLEAQ-2			
	UL		LL		UL		LL		UL		LL	
	Median	Std.	Median	Std.	Median	Std.	Median	Std.	Median	Std.	Median	Std.
PMM1	1.24E+03	9.74E+01	6.29E+05	8.12E+04	3.50E+03	6.81E+02	3.30E+05	6.58E+04	1.62E+03	4.45E+02	1.51E+04	2.46E+03
PMM2	2.04E+03	1.48E+02	1.42E+06	1.48E+05	3.50E+03	9.04E+02	2.65E+05	6.83E+04	1.51E+03	4.27E+02	7.19E+04	5.59E+03
PMM3	1.27E+03	1.24E+02	2.42E+05	2.03E+04	1.16E+03	5.61E+01	1.01E+05	4.79E+03	1.76E+03	5.17E+02	1.52E+05	1.37E+04
PMM4	1.46E+03	1.79E+02	7.42E+05	2.50E+05	2.55E+03	9.14E+02	2.35E+05	6.29E+04	2.02E+03	6.12E+02	1.13E+05	1.65E+04
PMM5	1.82E+03	2.47E+02	9.91E+05	1.64E+05	1.83E+03	6.36E+02	1.54E+05	3.42E+04	1.94E+03	7.16E+02	1.23E+05	1.69E+04
PMM6	3.50E+03	0.00E+00	2.91E+06	5.70E+04	2.91E+03	5.14E+02	2.22E+05	3.11E+04	2.24E+03	6.26E+02	1.06E+05	1.59E+04

Table 5.5: UL-LL error statistics were obtained from 31 independent runs of QBCA, BLCMAES, and BLEAQ-2 in high-dimensional PMM test problems. UL-LL function evaluations in high-dimensional PMM test problems.

Chapter 6

Automated Parameter Tuning Via Bilevel Optimization

Information

Part of this chapter is based on the paper: Jesus-Adolfo Mejía-de-Dios, Efrén Mezura-Montes and Marcela Quiroz-Castellanos (2021). *Automated parameter tuning as a bilevel optimization problem solved by a surrogate-assisted population-based approach*. Applied Intelligence, 51(8), 5978-6000.
`https://doi.org/10.1007/s10489-020-02151-y`

The automated parameter tuning problem (APTP) is modeled as a bilevel optimization problem in this chapter. To begin, the main definitions and theoretical results are presented to formalize the APTP in the framework of bilevel optimization. To spot potential regions in the parameter search space, the obtained bilevel optimization problem is solved using an adapted BCA combined with surrogate models. The suggested parameter tuner is tested on four typical numerical optimization metaheuristics on a collection of well-known and challenging test problems. Furthermore, a competitive algorithm for a well-known combinatorial optimization problem is also set up (considering a large benchmark suite). The experimental results are compared to those of Iterated Racing for Automatic Algorithm Configuration (also known as IRACE), a *state-of-the-art* parameter tuning approach. The results were validated by the Bayesian signed-rank statistical test.

6.1 Introduction

Identifying the conditions that promote the performance of search algorithms is one of the most important challenges in their development. As such algorithms are parameterized, the challenge is to discover a parameter setting that results in a suitable algorithmic behavior. Almost all metaheuristic approaches fall within this category. Because they offer a fair trade-off between efficacy and runtime, these approaches have proven to be particularly able to handle NP-hard problems. However, it is generally recognized that, in contrast to traditional exact methods, those approaches typically have parameters to fine-tune, and those values have a significant impact on the algorithm's final performance. The goal of parameter configuration can be accomplished in one of two ways: (a) offline, where the objective is to achieve an initial parameter setting before the optimization process begins (also known as parameter fine-tuning); or (b) online, where the parameter values are adjusted whilst the algorithm is running (also known as parameter control).

Many challenging search problems are frequently solved by algorithms configured by values, in which the user must identify configurations that allow the algorithm to perform properly [42]. In other words, the effectiveness of an algorithm is greatly influenced by its parameter values, which is a difficult challenge to solve in a variety of fields (e.g., evolutionary computation, machine learning, etc.) [59].

Traditional methodologies such as the brute force approach [42, 140], race-based methods [18, 85], population-based strategies [122, 141], and developing specific frameworks have been used to tackle the parameter tuning problem [60, 61, 44]. Traditional processes, in general, are limited to a few parameters of the algorithm to be set (target algorithm), because they may involve exhaustive searches for parameters of the target algorithm over discretized domains. Furthermore, those strategies are incompatible with expensive target algorithms. On the other hand, evolutionary parameter tweaking techniques have been presented with promising results [42].

One of the most popular parameter tuners was presented in [85], which includes the iterative race-based technique (IRACE). Such approach can identify good enough configurations while simultaneously minimizing the computational cost of invoking the algorithm to be configured by performing statistical checks. That is why it was chosen as a reference algorithm in this study.

The parameter tuning problem has been represented as a bilevel optimization task in recent years [116], with a hierarchical structure assisting in the construction of successful techniques to configure algorithms. Sinha et al. [117], for example, represented the parameter tuning problem as a bilevel optimization problem, which they addressed via an evolutionary bilevel algorithm. Furthermore, Andersson et al. [4, 5] used a bilevel optimization strategy to improve the efficiency of evolutionary algorithms by modifying and adjusting their parameters, despite the high computational cost of this tuning work.

In light of the foregoing, this chapter focuses on proposing a competitive alternative to address the automated parameter tuning problem by utilizing a novel bilevel optimization model, as well as considering an algorithm that incorporates a mechanism to reduce the computational load associated with this process by utilizing surrogate models to identify suitable parameter settings for the target algorithm. For specific algorithms that require fine-tuning, the model for the parameter tuning problem is a bilevel optimization work that takes into account scalability in terms of parameters and the number of problem instances. Here, the most convenient aspects of bilevel optimization are taken into account to decrease the parameter tuning problem's computing cost. The suggested method's anticipated benefit is that the solved bilevel optimization problem has an objective function that is defined directly in terms of the hardest instances that the algorithm to be tuned can solve, resulting in a reduction in the tuning process' computing cost.

The rest of the chapter is organized as follows: The Automated Parameter Tuning Problem (APTP) is detailed and modeled as a bilevel optimization task in Section 6.2, Section 6.3 explains the construction of the surrogate models to deal with the computational cost associated with the search process. Section 6.4 presents the extension of BCA to solve APTP. The experiments and discussion on the findings are presented in Section 6.5. Finally, Section 6.6 provides the conclusions.

6.2 The Automated Parameter Tuning Problem

Assume you need to calibrate an algorithm that has multiple types of parameters (also known as hyperparameters). Those parameters can be [59, 94]:

1. *Boolean parameters*: commonly used to activate or deactivate an algorithm feature, e.g., `{true, false}`.
2. *Categorical parameters*: finite unordered options, usually used to choose from a set of alternative features or mechanisms, e.g., `{sort, shuffle}`.
3. *Real-valued parameters*: related to the behavior of the algorithm's mechanisms, e.g., `{3.14, 2.7, 0.1}`.
4. *Integer-valued parameters*: a special case of real-valued parameters, or a numerical representation of categorical parameters, e.g., `{2, 3, 5, 7}`.

Because boolean and categorical parameters can be represented by integer numbers using a one-to-one function without losing generality, those different types of parameters can be reduced to either integer- or real-valued parameters (see Section 6.4). Furthermore, an algorithm is typically created to solve a certain type of optimization problem before being tested on a group of similar problems known as instances. A set of instances is referred to as a benchmark in this document and is denoted by $\mathcal{I}$.

The parameter tuning problem can be formally defined as follows [41, 59, 94]:

Definition 6.1: Parameter Configuration

Given an algorithm $\mathcal{A}(\Phi,\ I)$ with parameters $\Phi = (\phi_1, \ldots, \phi_n) \in X$ that affect its behavior when solving a set of instances $I \in \mathcal{I}$, the aim is then to find a configuration $\Phi^* \in X$ that provides an optimal performance of $\mathcal{A}$ on $\mathcal{I}$ according to some performance indicators. Here, X denotes the set of all valid configurations for $\mathcal{A}$.

The preceding definition applies to any algorithm that requires parameter values to be fine-tuned, as long as $\mathcal{A}$ produces a numerical number that represents the method's performance. As a result, to (numerically) assess how good an algorithm is at solving a collection of instances, measurements must be established. If the mean and variance of an algorithm's error are equal to zero (assuming the examples' solutions are known), the algorithm has successfully solved the instance.

As an example for Definition 6.1, consider three parameters: two real-valued parameters (step size and crossover probability), one integer-valued parameter (population size), and one categorical parameter related to the variation operator used to compute a solution (e.g., `DE/rand/1/bin` or `DE/rand/1/exp`), as proposed by Storn and Price in the Differential Evolution Algorithm [124]. DE was developed for global optimization, and each instance represents an optimization problem in this scenario.

The following section is focused on defining some indicators to quantify the performance of a target algorithm over either an instance or a set of instances.

6.2.1 Performance Indicators

This section focuses on evaluating the performance of algorithms such as metaheuristics and machine learning methods, taking into account three key factors: (1) accuracy, (2) stability, and (3) computational cost based on calculation time. The accuracy (also known as effectiveness) of a solution is determined by how near it is to the ideal answer (an error close to or equal to zero). Assume that ξ_I is a random variable that yields the probability distribution given by the error of an algorithm $\mathcal{A}(\Phi, I)$, when solving the instance I with a configuration Φ. Thus, the expected value of the error $\mu_1(\Phi) = E[\xi_I \mid \Phi]$ has to be minimum for a given configuration Φ. For simplicity, $E[\xi_I \mid \Phi]$ is denoted as $\mathcal{A}(\Phi, I)$.

The algorithm's performance variance, specifically the error value obtained by the metaheuristic under examination, is related to the algorithm's stability or robustness. Furthermore, a metaheuristic should be able to produce erroneous values without outliers, in which case the following indicators are taken into account. Here, $\mu_2(\Phi) = \text{var}[\xi_I \mid \Phi]$ and $\mu_3(\Phi) = |\xi_{\max} - \xi_{\min}|$.

The computing budget spent on each test problem (instance) is another important performance indicator to consider when configuring an algorithm, and this number can be controlled. $\mu_4 = R(\Phi)$ is a performance indicator that can be a regularization term or a computational cost quantifier required by the algorithm.

6.2.2 The Automated Parameter Tuning Task as a BO Problem

The automated parameter tuning problem is modeled and described in this section as a bilevel optimization problem (the original development of this mathematical problem has been detailed in [94]). Here, the upper-level problem is used to improve the performance of a target algorithm $\mathcal{A}$ (which solves a set of test problems referred to as instances) by taking a vector of parameters Φ whilst the lower-level problem finds the minimum number of instances in a benchmark $\mathcal{I}$ (with D_{ll} instances) that $\mathcal{A}$ is unable to solve for the given vector of parameters Φ provided.

Firstly, suppose that $\mathcal{A}(\Phi, I)$ is the target algorithm with parameters defined in Φ that solves a set of instances $\mathcal{I} = \{I_1, I_2, \ldots, I_{D_{ll}}\}$ in independent executions. The task of finding the best vector of parameters for $\mathcal{A}$ to solve several instances $\mathcal{I}$ is modeled as a bilevel optimization problem that works as follows:

1. A vector of parameters Φ is selected from the upper-level space of parameters.
2. The lower-level utilizes Φ to find the instances that are successfully solved by $\mathcal{A}(\Phi, I_i)$. A binary vector $\Theta = (\theta_1, \theta_2, \ldots, \theta_{D_{ll}})$ is used to set $\theta_i = 1$ when $\mathcal{A}(\Phi, I_i)$ successfully solves the instance I_i; otherwise $\theta_i = 0$ for $i = 1, 2, \ldots, D_{ll}$.
3. After that, the upper level evaluates each instance I_i satisfying $\theta_i = 0$. This allows the upper level to only evaluate the performance of unsolved instances in $\mathcal{I}$, i.e., $\mathcal{A}$ only evaluates instances I_i where the corresponding $\theta_i = 0$.

To represent the above-mentioned bilevel optimization technique, certain factors must be taken into account to generate objective functions with practical behavior. Firstly, assume that, $\Phi_1, \Phi_2 \in X$ are two different vectors of parameters or configurations, also $\Theta_1 \in \Psi(\Phi_1),\ \Theta_2 \in \Psi(\Phi_2)$ are two lower-level optimal solutions (from the lower-level), and $\mathcal{I}$ is a set of problem instances. Then:

1. $f(\Phi_1, \Theta_1) \geq 0$.
2. $f(\Phi_1, \Theta_1) = 0$ if and only if $\mathcal{A}$ successfully solves the entire benchmark.
3. $F(\Phi, \Theta) \geq 0$ for all $(\Phi, \Theta) \in X \times Y$.
4. $F(\Phi_1, \Theta_1) < F(\Phi_2, \Theta_2)$ if the performance of $\mathcal{A}(\Phi_1)$ is better than $\mathcal{A}(\Phi_2)$, i.e., $\mathcal{A}(\Phi_1)$ solves more instances and the mean error in each instance given by $\mathcal{A}(\Phi_1)$ is less than the mean error obtained by $\mathcal{A}(\Phi_2)$.

The first three conditions are stated to ensure low-bounded functions, while the fourth requirement is mentioned to numerically evaluate two different configurations for the target method. To achieve the aforesaid conditions, the next bilevel optimization problem is proposed.

Minimize:

$$F(\Phi, \boldsymbol{y}) = \left[\sum_{i=1}^{D_{ll}} (1 - y_i) E[\mathcal{A}(\Phi, I_i)]\right] + \lambda_1 \sum_{i=1}^{D_{ll}} (1 - y_i) + \lambda_2 R(\Phi) \tag{6.1}$$

s.t.

$$y \in \Psi(\Phi) = \text{argmin}\,\{f(\Phi, \boldsymbol{y}) \ : \ \boldsymbol{y} \in Y\} \tag{6.2}$$

where,

$$f(\Phi, \boldsymbol{y}) = \sum_{i=1}^{D_{ll}} |S(E[\mathcal{A}(\Phi, I_i)]) - y_i| \,. \tag{6.3}$$

Here, $\boldsymbol{y} \in Y$ is a binary vector, i.e., $Y = \{0,1\}^{D_{ll}}$ where y_i is associated to the instance I_i. E is the expected value of the error given by $\mathcal{A}(\Phi, I_i)$ and S is a mapping defined as follows:

$$S(a) = \begin{cases} 1 & \text{if } a = 0 \\ 0 & \text{otherwise} \end{cases} . \tag{6.4}$$

Moreover, $\lambda_1 > 0$ and $\lambda_2 > 0$ where

$$\lambda_1 > \sum_{i=1}^{D_{ll}} E[\mathcal{A}(\Phi, I_i)] \text{ for all } \Phi \in X \tag{6.5}$$

and $\lambda_2 > 0$ penalizes the regularization term defined by $R(\Phi)$. Here, R is used to control the complexity of $\mathcal{A}$. An illustrative example for the BO problem (6.1)-(6.3) is given in Figure 6.1 which offers an interesting step function that favors promising configurations for target algorithm $\mathcal{A}$.

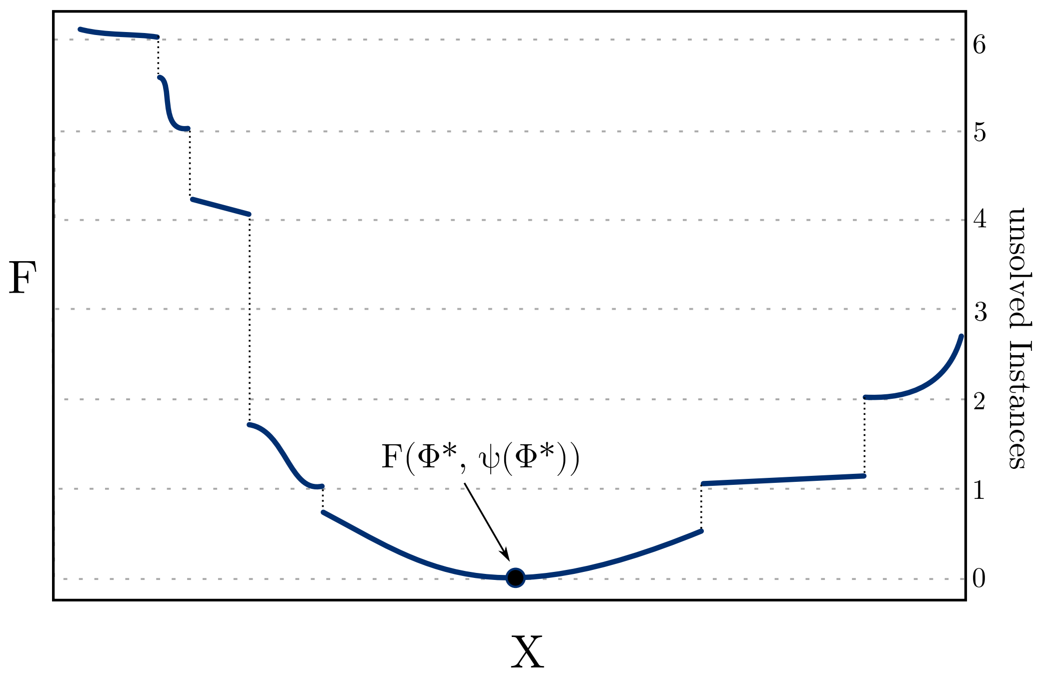


Figure 6.1: Illustrating the objective function values for the APTP modeled as a bilevel optimization task.

The following properties emerge from the the above bilevel optimization problem.

Proposition 6.2.1. *Assume that equations (6.1)-(6.3) define a bilevel optimization problem and algorithm $\mathcal{A}$ is solving a set of instances $\mathcal{I}$. Then, $|\Psi(\Phi)| = 1$, i.e., $\Psi(\Phi)$ is a unit set for all $\Phi \in X$*

Proof. Suppose that there exists $\Phi \in X$ such that $\boldsymbol{y}, \boldsymbol{z} \in \Psi(\Phi)$ are different, then for some $i, j \in \{1, 2, \ldots, D_{ll}\}$ such that $y_i \neq z_i$. Thus, $E[\mathcal{A}(\Phi, I)] = 0$ and $E[\mathcal{A}(\Phi, I)] > 0$ for an instance $I \in \mathcal{I}$ but that is a contradiction. Therefore, $\boldsymbol{y} = \boldsymbol{z}$. $\square$

The above proposition states an important result to be used for approximating the mapping $\Psi(\Phi)$ because only the optimistic position can exist [38]. That is convenient because different algorithms have been proposed in the specialized literature for such optimistic position. Hence, the notation $\psi(\Phi) \in \Psi(\Phi)$ for all $\Phi \in X$ is introduced, where $(\Phi, \psi(\Phi))$ will denote a feasible solution.

The following results are related to the way the UL interacts with infeasible solutions [94].

Theorem 6.1

For each feasible solution $(\Phi, \psi(\Phi)) \in X \times Y$ for the bilevel optimization problem given by equations (6.1)-(6.3), there exists $\boldsymbol{y} \in Y$ such that $F(\Phi, \psi(\Phi)) \geq F(\Phi, \boldsymbol{y})$.

Proof. Assume that $(\Phi, \psi(\Phi)) \in X \times Y$ such that $\boldsymbol{y} = \psi(\Phi)$ the result becomes trivial. However, if $\boldsymbol{y} \in Y$ such that $\sum_{i=1}^{D_{ll}} y_i < \sum_{i=1}^{D_{ll}} \psi(\Phi)_i$, then $F(\Phi, \psi(\Phi)) \geq F(\Phi, \boldsymbol{y})$ based on the inequality in Equation (6.5). $\square$

Corollary 6.1

If there exists a feasible solution $(\Phi^*, \psi(\Phi^*)) \in X \times Y$ such that $E[\mathcal{A}(\Phi, I)] = 0$, $\forall\ I \in \mathcal{I}$, then $F(\Phi^*, \psi(\Phi^*)) = \lambda_2 R(\Phi^*)$ and $f(\Phi^*, \psi(\Phi^*)) = 0$.

This section was used to describe the APTP as a BO problem using several varieties of quantifiable-performance algorithms. The structure described above, on the other hand, can be modified for parameter tuning operations in which an algorithm solves a collection of instances. However, it is common to discover that the algorithm to be configured is computationally expensive, in which case the above definition may also be computationally expensive. The following part describes how to handle the aforementioned computational cost.

6.3 Surrogate Model

The purpose is to train a surrogate model by offering feasible solutions that have been evaluated in the exact model (Equation 6.6), i.e., a feasible solution for a problem defined by Equations (6.1)-(6.3) is obtained when $\mathcal{A}(\Phi, I)$ solves every instance in $\mathcal{I}$.

A feasible solution can also be represented as $(\Phi, \psi(\Phi))$. As a result, the following BO problem is transformed:

$$\min_{\Phi \in X} \{F(\Phi) = F(\Phi,\ \psi(\Phi))\}. \tag{6.6}$$

To tackle the high computing cost of evaluating solutions in the exact model, this transformed problem is utilized to approximate the leader objective function using a surrogate model. The goal is to create an adaptive statistical model that can approximate outcomes at a minimal cost of computing. Here, a training dataset is defined as follows:

$$P_{\text{train}} = \{(\Phi_1,\ \psi(\Phi_1)),\ (\Phi_2,\ \psi(\Phi_2)), \dots, (\Phi_m,\ \psi(\Phi_m))\} \tag{6.7}$$

As can be seen, solely feasible solutions are taken into account while constructing a prediction model. Approximate solutions are generated by radial basis function-based models utilizing feasible data points in the search space, following guidelines given in [102]. It is worth mentioning that different authors [102, 108] suggest that polynomial term Q will help to improve the approximation $F(x) = \hat{F}(x) + \varepsilon$:

$$\hat{F}(x) = \sum_{i=1}^{n} \alpha_i k(x, x_i) + Q(x).$$

In practice, Q is considered to be constant because it offers adequate approximations, which implies that $z = A\beta$ where:

$$\beta = \begin{pmatrix} \alpha \\ \theta \end{pmatrix}, \quad A = \begin{pmatrix} G & 1 \\ 1 & 0 \end{pmatrix} \in \mathbb{R}^{(N+1)\times(N+1)}.$$

Note that, θ represents the coefficient of a constant polynomial Q.

Additionally, various adjustments should be made to direct the search to promising regions in the parameter search space while avoiding over-fitting. This is because stochastic algorithms' parameter tuning problem can result in random outcomes influenced by outliers. As a result, the polynomial kernel function model of noisy data is adopted:

$$\beta = (A - \lambda I)^{-1} z,$$

where λ is a noise variance-related regularization parameter. If λ is greater than 0, the approximate model $\hat{F}$ will not meet the condition $F(\Phi) = \hat{F}(\Phi)$. It will, however, distinguish between representative data and noise while maintaining the environmental bias.

6.4 Proposed Approach

This section describes the BCAP (Bilevel Centers Algorithm for Parameter Tuning) algorithm, which is used to solve the BO problem specified in Section 6.2.2, (6.1)-(6.3). Here, BCAP is developed under the BCA framework with a surrogate-model strategy suggested in [91], which is useful to save evaluations mainly at the upper level of the BO problem.

An algorithm $\mathcal{A}$ can have either discrete or real value parameters as described in Section 6.2. That is, the upper-level search space (parameter space) denoted by X is defined, without loss of generality, on $X \subset \mathbb{Z}^m \times \mathbb{R}^n$. It can be noted that X defines a mixed discrete-continuous search space. Therefore, X is transformed by using an injective function. Firstly, consider $X = X_1 \times X_2$ with $X_1 = \mathbb{Z}^m$,

$X_2 = \mathbb{R}^n$, and $\tilde{X}_1$ as the shortest close interval satisfying $X_1 \subset \tilde{X}_1$. Then X_1 is transformed into $\tilde{X}_1$ by using the identity mapping, and $\tilde{X}_1$ into X_1 via the transformation $T : \tilde{X} \to X_1$ where $T(\tilde{\Phi}) = (\lfloor\tilde{\phi}_1\rfloor, \lfloor\tilde{\phi}_2\rfloor, \dots, \lfloor\tilde{\phi}_m\rfloor)$ with $\lfloor\cdot\rfloor$ defined as the floor function.

Example 6.1

The Differential Evolution algorithm (details in Section 6.2) has three main parameters:

- *Population size*: a non-negative integer number $\phi_1 \in [3, 1000]$
- *Crossover probability*: a real number $\phi_2 \in [0, 1]$.
- *Stepsize*: a real number $\phi_3 \in [0, 2]$.

Here, parameters space can be transformed into $\tilde{X} = [3, 1000] \times [0, 1] \times [0, 2]$.

The main objective here is the configuration of metaheuristics for solving numerical optimization problems stated as:

Minimize

$$f_i(w),\ w \in \mathbb{R}^D$$

subject to:

$$w \in W$$

Assume that $\mathcal{A}$ is an algorithm with parameters $\Phi = (\phi_1, \phi_2, \dots, \phi_n)$ which should minimize, independently, a set of test instances $f_i(w)$ subject to $w \in W$ for $i = 1, 2, \dots, D_{ll}$. The next subsections detail BCAP and its usage to solve the APTP.

6.4.1 BCAP Components

BCAP works with a population given by $\tilde{X} \times Y$, and its elements are utilized to train a surrogate model to help with the optimization process. After that, an iterative procedure using the variation operator based on the center of mass approach is utilized to generate alternatives (parameter values) to be assessed in the approximated or exact objective function (see Section 6.3). To increase the quality of the surrogate model, the size of the training dataset N_{train} is also changed. Each of the steps is described in-depth in the following sections.

- *Initial population of parameters:* Here, the population for the BO problem for APTP (6.1)-(6.3) is defined as usual: $P = \{(\Phi_1, y_1), (\Phi_2, y_2), \dots . (\Phi_N, y_N)\}$.
- *Generating candidate parameters:* Let $(\Phi, \boldsymbol{y}) \in P$ a solution, then a new upper-level vector of parameters is generated by using the center of mass concept with the upper-level function in Equation (6.1) as to compute the mass as in Section 5.1.2:

$$\Phi_{\text{new}} = \Phi + \eta(\boldsymbol{c} - \boldsymbol{u}_{\text{worst}}), \tag{6.8}$$

- *Fixing solutions:* When an element in Φ_{new} exceeds its established limits, then such entry is replaced by the corresponding entry in the center of mass used to compute Φ_{new}. Finally, the new solution Φ_{new} replaces the worst solution in P if it is better than current Φ.
- *Finding feasible solutions:* When Φ_{new} is computed, approximation for $\boldsymbol{y} \in \Psi(\Phi_{\text{new}})$ is performed. Here, BCAP uses feasible solutions in P as detailed in Section 6.4.2.
- *Training surrogate model:* Regarding the surrogate modeling outlined in Section 6.3, to choose the train data set (P_{train} with N_{train} elements) used to approximate the exact model $\hat{F}$, it is suggested to use all those solutions evaluated in $\hat{F}$ to construct the surrogate model.

Figure 6.2 shows a simplified diagram to illustrate the main processes during each BCAP call. Note that when the surrogate model $\hat{F}$ is calculated, a local search is carried out at the upper-level search space (parameters space) by considering the surrogate mapping $\hat{F}$ trained with P_{train}. The best solution $(\Phi_{\text{best}}, y_{\text{best}}) \in$ argmin$\{F(\Phi, \boldsymbol{y}) : (\Phi, \boldsymbol{y}) \in P\}$ is utilized for approximating a local optimum of $\hat{F}$ which is a continuous and smooth function. This local search is performed via the quasi-Newton method BFGS [78, 91]. Assume that $\hat{F}$ was successfully trained, then the properties of feasible solutions can be discovered by BCAP, but with a reduced number of calls to the target algorithm.

In order to avoid premature convergence, an exploration stage is performed as follows: For each vector of parameters Φ in a population P, Φ_{new} is computed by using Equation (6.8). After that, Φ_{new} is evaluated in the upper-level function F, and the worst solution in P is updated by Φ_{new} when this new one takes better F values than the current solution $F(\Phi_{\text{new}}, y) < F(\Phi, \boldsymbol{y})$ defined in Equations (6.1)-(6.3).

6.4.2 Lower-Level Procedure

Let Φ_{new} be a new UL parameter vector, its corresponding feasible solution $\psi(\Phi_{\text{new}})$ is computed by adopting the strategy given in [91]. This strategy works as follows: find the nearest UL in P to Φ, i.e., $(\Phi_{\text{nearest}}, \boldsymbol{y}_{\text{nearest}}) \in$ argmin$\{\|\Phi_{\text{new}} - \Phi\| : (\Phi, \boldsymbol{y}) \in P\}$, then a simplistic approximation for the lower level is $\psi(\Phi) \approx \boldsymbol{y}_{\text{nearest}}$.

Figure 6.3 illustrates how the approximations to LL optimal solutions are obtained. Considering that these approximations are used, the upper level only calls the algorithm $\mathcal{A}(\Phi_{\text{new}}, I_i)$ on those instances where the corresponding entries in $\boldsymbol{y}_{\text{nearest}}$ are equal to zero. This step helps to save computational effort associated with the target algorithm $\mathcal{A}$.

Since this naive strategy can lead to an infeasible solution, a model reevaluation is performed to ensure lower-level optimality. Therefore, each solution in P is re-evaluated at the lower level every certain number of iterations, i.e., for each vector of parameters $(\Phi, \boldsymbol{y}) \in P$, solve every problem instance with the target algorithm $\mathcal{A}$ to form the corresponding optimal solution $\psi(\Phi)$ as described in Section 6.2.

6.4.3 Complete Algorithm

This part summarizes the design of BCAP to solve the bilevel optimization problem defined by Equation (6.6) for the automated parameter tuning:

- **Step 1:** Create a population $P \subset X \times Y$ with N vector of parameters randomly generated (uniform distribution) at the upper-level search space.
- **Step 2:** Build the surrogate model $\hat{F}$ as described in Section 6.3 to detect biased parameters.
- **Step 3:** Find an optimal solution from $\hat{F}$ by using the Quasi-Newton method BFGS using the best configuration found so far as an initial guess.
- **Step 4:** Approximate the lower-level problem for the newly generated candidate solution.
- **Step 5:** Evaluate the upper level and replace the worst solution in the population if the newly generated solution is better than the current configuration.
- **Step 6:** After a certain number of generations, reevaluate the entire population to avoid infeasible solutions.
- **Step 7:** Generate new upper-level solutions and go to Step 4.

A detailed pseudocode is given in Algorithm 7.

6.5 Experiments and Discussion

This section details the experimentation to evaluate the performance of BCAP, which is used to determine the parameter configuration of two evolutionary al-

Algorithm 7: BCAP

```
1  Choose $\ell$ and $\lambda > 0$
2  $N \leftarrow K * D$, $t_r \leftarrow 5$
3  Initialize a population $P \subset \tilde{X} \times Y$ with $N$ elements.
4  while the end criterion is not achieved do
5      Compute a surrogate model $\hat{F}$ by using the current population $P$ for
         training (Section 6.3).
6      Since $\hat{F}$ is smooth, use a gradient-based method to find
         $\hat{\Phi} \in$ argmin $\hat{F}(\Phi)$.
7      Choose the best solution in $P$ to initialize that search procedure.
8      Solve lower-level problem $\hat{y} \in \text{argmin}_{z \in Y} f(\hat{\Phi}, z)$ (Section 6.2).
9      if $(\hat{\Phi}, \hat{y})$ satisfies the Corollary 6.1 then
10         Stop and report $(\hat{\Phi}, \hat{y})$
11     else
12         Update the best solution $(\Phi_{\text{best}}, y_{\text{best}})$ in $P$ by $(\hat{\Phi}, \hat{y})$ if
             $F(\hat{\Phi}, \hat{y}) < F(\Phi_{\text{best}}, y_{\text{best}})$.
13     for each $(\Phi,\ y)$ in $P$ do
14         Compute $\Phi_{\text{new}} = \Phi + \eta_X(c_X - u_{\text{worst}})$ (Section 6.4.1).
15         Use the procedure described in Section 6.4.2 to find
             $y_{\text{new}} \in \text{argmin}_{z \in Y} f(\Phi, z)$.
16         if $F(\Phi_{new},\ y_{new}) < F(\Phi,\ y)$ then
17             Replace the worst element in $P$ with $(\Phi_{\text{new}},\ y_{\text{new}})$.
18     Re-evaluate all solutions in $P$ every $t_r$ generations to avoid infeasible
         solutions.
19 Report the best solution in $P$.
```

gorithms, two swarm intelligence algorithms, and one physics-based algorithm. Differential Evolution (DE), Particle Swarm Optimization (PSO), Artificial Bee Colony (ABC), and Evolutionary Centers Algorithm (ECA) are considered to solve a set of test problems taken from a recent benchmark for numerical optimization, while GGA-CGT, a grouping-based genetic algorithm, is also analyzed when solving combinatorial optimization problems (Bin-Packing problems). The BCAP results are compared to those of IRACE, a state-of-the-art and widely used parameter tuning approach described later in this section.

Before presenting the experiments related to the configuration of representative metaheuristics for optimization, they are described (as well as their parameters) in the following list:

- ABC [73]: The Artificial Bee Colony (ABC) is a swarm intelligence algorithm based on the honey bee foraging behavior. ABC works like this: given N bees, a percentage Ne of them are hired to identify food sources (solutions), and these solutions are visited up to "limit" times by $No = N - Ne$ observer bees seeking to enhance them.
- ECA [90]: The center of mass concept is used in the physics-based Evolutionary Centers Algorithm to create biased vectors for adjusting the worst elements into better regions of the search space. ECA works by computing a center of mass (biased to promising regions) for each solution in a population with N elements, then using $K < N$ elements in the population to generate a new direction to the center of mass (biased by the worst solution to avoid premature convergence) with a stepsize $\eta_{\max}$.
- DE [124]: Differential Evolution is an evolutionary algorithm for solving global optimization problems in continuous spaces. Since its most popular variation, DE/rand/1/bin, can provide competitive results in a variety of optimization problems, DE is an efficient population-based technique. DE starts with a population of N solutions, and then computes directions with stepsize F and applies a binomial crossover with probability CR to construct a new solution.
- PSO [74]: The behavior of a birds flock inspired Particle Swarm Optimization, a swarm intelligence algorithm. PSO works with N particles and their velocities (search direction), which are iteratively computed using the C_1 and C_2 parameters for social and cognitive knowledge, respectively, as well as the inertia weight ω, which governs the influence of the preceding velocity.
- GGA-CGT [105]: For the Bin Packing Problem, the Grouping Genetic Algorithm with Controlled Gene Transmission (GGA-CGT) uses a population of

N individuals, with p_c and p_m determining the percentage of individuals to be recombined and mutated, respectively. The parameters k_{ncs} (to remove bins in a non-cloned solution), k_{cs} (to remove bins in a cloned solution), B_{size} (number of individuals in an elite set), and L_s (generations where at most an individual can be cloned) regulate the methods used by this algorithm to develop and improve solutions to update the population.

BCAP was implemented in the Julia Programming Language [16]. The BCAP parameters were obtained by applying a grid search with $\ell \in \{3, 4, \dots, 10\}$, $\eta_{\max} \in \{0.8, 1, 1.2, 1.4, \dots, 4\}$, and $\lambda \in \{0, 0.1, 0.2, \dots, 1\}$. The suggested parameter values were the following: $N = \ell \cdot n$, $\ell = 6$, $\eta_{\max} = 1.2$ and $\lambda = 10^{-1}$. This empirical study suggested that, for large ℓ values (> 7), BCAP could converge faster into a local minimum, and small values of $\eta_{\max}$ (< 1) are related to the exploitation of promising regions of the search space. On the other hand, large values of λ are not recommended since the surrogate model can converge to constant functions.

The original versions of the five metaheuristics are used to see if a proper configuration lets them handle difficult optimization instances without needing extra mechanisms.

The first ten test functions (instances) from the CEC 2017 benchmark were used to evaluate the efficiency of each numerical optimization metaheuristic (ABC, ECA, DE, and PSO) [10]. These are 10-dimensional optimization problems with unimodal, multimodal, separable, and/or non-separable objective functions. The main qualities of such functions are reported in Table 6.1. Using representative Bin Packing Problem (BPP) examples, the performance of GGA-CGT was evaluated. It is worth noting that most algorithms proposed in the literature have been tested for the BPP study utilizing a well-studied test benchmark; it includes 1615 instances in which the number of items n_{items} varies within $[50, 1000]$, the bin capacity c is within [100, 100000] and the ranges of the weights are within $(0, c]$.

IRACE [85] is an iterated F-race (I/F-Race) extension with an iterated racing method that is part of the IRACE procedure (also known as the IRACE package).

Problem	Function	Properties	Minimum
f_1	Bent Cigar	Unimodal, non-separable and smooth but narrow ridge	100
f_2	Sum of Different Power	Unimodal, symmetric and non-separable	200
f_3	Zakharov	Unimodal and non-separable	300
f_4	Rosenbrock	Multimodal (huge number of local minimums) and non-separable	400
f_5	Rastrigin	Multimodal (huge number of local minimums), asymmetrical and non-separable	500
f_6	Expanded Scaffer F6	Multimodal (huge number of local minimums) and non-separable	600
f_7	Lunacek Bi-Rastrigin	Multimodal, differentiable nowhere, asymmetrical and non-separable	700
f_8	Non-Continuous	Multimodal (huge number of local minimums), asymmetrical and non-separable	800
f_9	Levy	Multimodal and non-separable	900
f_{10}	Schwefel	Multimodal, non-separable and second better local minimum is far from the global minimum	1000

Table 6.1: The first ten CEC 2017 test functions, where their search space is bounded within $[-100, 100]^D$ and $D = 10$. Each function has a shifted and rotated optimum to provide real-world problem properties.

IRACE was created to select the proper algorithm configuration for optimization and decision-making problems, as well as strategies to prevent premature convergence. IRACE also supports real, integer, category, and ordinal parameters. Figure 6.4 depicts a diagram that explains how IRACE works.

BCAP and IRACE independently ran 10 times per each considered metaheuristic (ECA, DE, ABC, and PSO). BCAP was limited to call $200D_{ll} = 2000$ times the target algorithm over the whole benchmark. On the other hand, IRACE called $400D_{ll} = 4000$ times the target metaheuristic. As can be seen, BCAP had a constrained budget (half of the IRACE target metaheuristic calls) since the aim is to explore if BCAP could produce comparable outcomes at a lower computational cost. Other alternatives, it should be noted, require a bigger budget (up to $1000D_{ll}$) to configure an algorithm [128].

The pairwise Bayesian signed-rank test, recently offered as a helpful option for comparing stochastic algorithms, was used to analyze the statistical results produced by BCAP and IRACE [24] as follows:

1. Each approach was executed ten times, as mentioned before, to configure a target algorithm $\mathcal{A}$ on a set of instances $\mathcal{I}$.
2. The ten results of each approach were stored as: $C_{\text{BCAP}} = \{\Phi^{(1)}_{\text{BCAP}}, \ldots, \Phi^{(10)}_{\text{BCAP}}\}$ and $C_{\text{IRACE}} = \{\Phi^{(1)}_{\text{IRACE}}, \ldots, \Phi^{(10)}_{\text{IRACE}}\}$,
3. For each one of the ten configurations in C_{BCAP} and C_{IRACE}, each test instance in $\mathcal{I}$ was solved thirty-one times.
4. The ten configurations in C_{BCAP} and C_{IRACE} were sorted by the number of test instances successfully solved (the error per instance is smaller than 10^{-4}) or the mean error value.
5. The pairwise Bayesian signed-rank test was applied to the set of thirty-one runs located in the median value of the ten configurations.

Furthermore, to configure GGA-CGT for BPP, we utilize an approach based on its authors' suggestions [105], which can be summarized as follows: (1) Select

50 typical instances from 1615 available; (2) solve the complete benchmark and report the percentage of successfully solved instances once GGA-CGT is configured. BCAP and IRACE each ran 10 times to configure GGA-CGT (on the same 50 instances), and the percentage of solved instances was reported after each run. To summarize, “train” instances are used to configure this technique for combinatorial optimization and “test” instances are used to evaluate the algorithm performance.

6.5.1 Configuring ABC

Regarding the experiments with ABC, the following parameters were considered: $\Phi = (N, \text{ limit}, \ Ne) \in \mathbb{R}^3$ where $N \in [10, 500]$ is the number of bees in the swarm (population size), limit $\in [0, 1000]$ and the number of employed bees is $\lfloor Ne \cdot N + 0.5 \rfloor$ with $Ne \in [0, 1]$ (based on the empirical study carried out in [1]). The best configurations found by BCAP and IRACE are presented in Table 6.2 and the Bayesian signed-rank test result (in terms of the instances solved) is in Figure 6.5.

Based on the statistical test results in Fig. 6.5, BCAP was able to provide better parameter configurations in almost 60% of the executions, compared with the almost 26% provided by IRACE. Taking a closer look at the results in Table 6.2, it was clear that none of the configurations was able to provide successful runs (premature convergence was observed in Figure 6.6). On the other hand, BCAP obtained parameter values within the following ranges: $N \in [254, 395]$, limit $\in [273, 949]$, $Ne \in [0.22345, 0.530335]$, while IRACE’s ranges were as follows: $N \in [253, 357]$, limit $\in [72, 783]$, and $Ne \in [0.179173, 0.555798]$. Those values suggest that BCAP can find better configurations even if the ranges are somehow similar. Furthermore, the computational cost required by BCAP is half of that required by IRACE.

	Run	N	Ne	limit	Instances solved	Mean Error
BCAP	1	254	0.288465	949	0	1.118e+06
	2	374	0.407581	924	0	1.385e+06
	3	341	0.501753	801	0	1.705e+06
	4	295	0.22345	690	0	2.198e+06
	5	353	0.410992	486	0	3.163e+06
	6	371	0.493897	538	0	3.385e+06
	7	292	0.378883	326	0	4.572e+06
	8	340	0.473353	324	0	5.545e+06
	9	395	0.352415	394	0	5.859e+06
	10	360	0.530335	273	0	6.672e+06
IRACE	1	352	0.318848	783	0	1.706e+06
	2	357	0.555798	722	0	2.387e+06
	3	253	0.312723	435	0	3.177e+06
	4	273	0.179173	216	0	6.521e+06
	5	273	0.179173	216	0	6.521e+06
	6	301	0.291851	187	0	9.079e+06
	7	270	0.374526	164	0	9.324e+06
	8	320	0.210675	170	0	9.632e+06
	9	317	0.249801	83	0	1.541e+07
	10	325	0.442315	72	0	1.695e+07

Table 6.2: Best configuration of each one of the ten independent runs by BCAP and IRACE when configuring ABC.

6.5.2 Configuring ECA

The physics-inspired metaheuristic called ECA was configured by using the following vector of parameters $\Phi = (N, K, \eta_{\max}) \in \mathbb{R}^3$, where $N \in [10, 500]$, $K \in [2, 10]$, and $\eta_{\max} \in [0, 4]$. The best configurations found by IRACE and BCAP are presented in Table 6.3 and the statistical test results are found in Fig. 6.7.

The Bayesian signed-rank test results suggest that BCAP provided better configurations in 47.5% of the executions, a similar performance concerning IRACE was shown in the remaining 52.5%, and IRACE was unable to outperform BCAP in any single case. Unlike the configurations for ABC, for ECA, both approaches (BCAP and IRACE) were able to tune them to get instances successfully solved (see that local optima were avoided in some convergence graphs in Figure 6.8).

BCAP found configurations for ECA in the following ranges: $N \in [138, 289]$, $K \in [4, 8]$, and $\eta_{\max} \in [1.037687, 1.379108]$. On the other hand, IRACE provided the following intervals: $N \in [125, 247]$, $K \in [3, 10]$, $\eta_{\max} \in [1.01588, 2.01618]$. It is interesting to note that for $\eta_{\max}$, BCAP suggested lower values with respect to IRACE, and that difference (smaller stepsize values for the variation operator) led to a better performance by BCAP with 50% of the computational cost.

6.5.3 Configuring DE

DE was configured by using the following parameter vector $\Phi = (N, F, CR)$, where the limits were $N \in [10, 500]$, $F \in [0, 2]$, and $CR \in [0, 1]$ (obtained from [51]). The best configurations found by IRACE and BCAP are presented in Table 6.4, whereas the statistical test results are depicted in Fig. 6.9.

Regarding DE, BCAP, based on Fig. 6.9, was competitive in 78.3% of the executions, while a similar performance with IRACE was observed in just 1.3%, and finally, IRACE was better in 21.7% of the executions.

The DE parameters were bounded by BCAP as follows: $N \in [62, 340]$, $F \in [0.239642, 0.36847]$, $CR \in [0.692803, 0.988431]$. IRACE found the next ranges: $N \in [13, 57]$, $F \in [0.462729, 0.987232]$, $CR \in [0.117163, 0.97588]$.

Unlike the ranges for ABC and ECA, those found by BCAP and IRACE for DE were particularly different, i.e., larger populations, smaller stepsizes, and larger crossover rates by BCAP and the opposite by IRACE.

It was expected that DE was able to obtain instances successfully solved because it is one of the most competitive metaheuristics for numerical optimization [32] (see Figure 6.10 where a suitable convergence behavior was observed). However, it is worth highlighting that BCAP was able to let ECA provide similar

	Run	N	K	$\eta_{\max}$	**Instances solved**	**Mean Error**
BCAP	1	223	5	1.132768	6	4.956e+01
	2	152	6	1.185632	6	5.710e+01
	3	262	7	1.379108	6	6.506e+01
	4	157	7	1.376814	6	6.596e+01
	5	186	8	1.49591	6	7.115e+01
	6	201	4	1.037687	5	4.404e+01
	7	289	5	1.051635	5	4.689e+01
	8	209	4	1.143711	5	4.810e+01
	9	183	6	1.047635	5	4.890e+01
	10	138	4	1.207521	5	6.350e+01
IRACE	1	189	3	1.368956	6	2.172e+01
	2	154	5	1.362471	6	2.590e+01
	3	181	4	1.597606	6	3.075e+01
	4	137	7	2.016182	6	3.195e+01
	5	137	7	2.016182	6	3.195e+01
	6	188	9	1.447275	6	3.329e+01
	7	125	8	1.921186	6	3.395e+01
	8	247	5	1.198838	6	4.048e+01
	9	179	10	1.785714	6	4.639e+01
	10	204	8	1.015882	5	3.069e+01

Table 6.3: The best configuration of each one of the ten independent runs by BCAP and IRACE when configuring ECA.

performance as that of DE (see Tables 6.3 and 6.4).

	Run	N	F	CR	**Instances solved**	**Mean Error**
BCAP	1	340	0.269756	0.945315	5	7.556e+01
	2	316	0.319333	0.907851	5	9.046e+01
	3	292	0.297554	0.866267	5	9.783e+01
	4	141	0.36847	0.988431	5	1.023e+02
	5	181	0.285236	0.919108	5	1.424e+02
	6	243	0.332965	0.857937	4	9.901e+01
	7	332	0.244348	0.811175	4	1.081e+02
	8	231	0.239642	0.793729	3	1.055e+02
	9	168	0.2867	0.799051	3	1.359e+02
	10	62	0.340536	0.692803	2	3.224e+02
IRACE	1	57	0.600528	0.97588	6	5.518e+01
	2	22	0.987232	0.78687	6	1.650e+07
	3	13	0.744198	0.236023	5	1.921e+01
	4	26	0.575771	0.817107	5	4.352e+01
	5	16	0.499888	0.173194	5	8.527e+01
	6	13	0.5337	0.200412	5	5.664e+02
	7	14	0.495335	0.317386	4	1.789e+02
	8	13	0.462729	0.117163	4	3.707e+03
	9	13	0.462729	0.117163	4	3.707e+03
	10	35	0.468767	0.865054	4	6.669e+03

Table 6.4: The best configuration of each one of the ten independent runs by BCAP and IRACE when configuring DE.

6.5.4 Configuring PSO

PSO was configured with the following parameter vector $\Phi = (N, C_1, C_2, \omega) \in \mathbb{R}^4$ with $N \in [10, 500]$, $C_1, C_2 \in [0, 4]$, and $\omega \in [0, 1]$ (based on the theoretical study performed in [71]).

The best configurations found by IRACE and BCAP are presented in Table 6.5 and the statistical test results are found in Fig. 6.11.

In this case, the performances by BCAP (48%) and IRACE (47.6%) are similar. However, BCAP just requires half of the calls to the target algorithm (PSO in this comparison).

BCAP reported parameters bounded in the following ranges: $N \in [151, 439]$, $C_1 \in [1.130716, 3.89815]$, $C_2 \in [0.857492, 3.516415]$, $\omega \in [0.489866, 0.890124]$. In contrast, the best IRACE configurations were limited by the ranges: $N \in [297, 488]$, $C_1 \in [0.324315, 3.4954]$, $C_2 \in [0.792152, 3.2211]$ and the inertial weight $\omega \in [0.36353, 0.753773]$. As it can be noted, the ranges vary between the compared approaches. However, a similar performance was observed in this particular metaheuristic for which the specialized literature reports that its fine-tuning process can be difficult [126, 127]. Finally, premature convergence was observed, see Figure 6.12, that coincides with the low number of instances solved in Table 6.5.

6.5.5 Configuring GGA-CGT

GGA-CGT was configured with the following parameter vector:

$$\Phi = (N, p_m, p_c, k_{ncs}\ , k_{cs}, B_{\text{size}}, L_s) \in \mathbb{R}^7$$

with $N \in [50, 400]$, $p_m \in [0, 0.9]$ and $p_c \in [0, 0.6]$, $k_{ncs}, k_{cs} \in [1, 6]$, $B_{\text{size}} \in [0, 0.5]$, $L_s \in [1, 50]$. It is worth mentioning that those bounds have been suggested by the authors of GGA-CGT [105]. Here, BCAP was limited to call 2000 times GGA-CGT, whilst IRACE was limited to call 5000 times GGA-CGT. Both, BCAP and IRACE

	Run	N	C_1	C_2	ω	**Instances solved**	**Mean Error**
BCAP	1	301	3.89815	3.38136	0.489866	1	3.631e+02
	2	219	3.725067	2.712283	0.690293	1	4.329e+02
	3	411	1.520623	3.516415	0.543125	1	4.475e+02
	4	151	3.022239	1.648836	0.810039	1	5.439e+02
	5	297	1.81692	1.957531	0.732172	1	5.723e+02
	6	439	1.130716	3.24939	0.579927	1	6.444e+02
	7	416	2.63786	3.162043	0.633675	1	6.477e+02
	8	194	2.106226	0.857492	0.890124	1	7.319e+02
	9	218	3.056313	2.463493	0.735224	1	7.340e+02
	10	166	2.607728	2.916444	0.690893	1	7.383e+02
IRACE	1	344	0.324315	3.221096	0.492841	1	5.446e+02
	2	371	2.246175	1.925117	0.678153	1	6.707e+02
	3	488	0.374345	1.186899	0.753773	1	7.076e+02
	4	351	2.193293	1.894675	0.625818	0	7.444e+02
	5	297	0.488361	2.891988	0.479847	0	7.810e+02
	6	433	0.786563	3.203202	0.382314	0	8.900e+02
	7	414	0.914002	2.903228	0.36353	0	9.543e+02
	8	414	0.914002	2.903228	0.36353	0	9.543e+02
	9	381	3.153645	0.923493	0.534409	0	4.116e+04
	10	324	3.495401	0.792152	0.550182	0	7.560e+04

Table 6.5: The best configuration of each one of the ten independent runs by BCAP and IRACE when configuring PSO.

utilized the same seed and instances to configure this target algorithm in order to promote a fair comparison.

The best configurations found by IRACE and BCAP are presented in Table 6.6 and the percentage of the optimal solutions is shown in Fig. 6.13.

In this case, the results by BCAP and IRACE suggest that both approaches obtained parameters that lead to a similar GGA-CGT performance. However, BCAP found a better vector of parameters in most runs as presented in Fig. 6.13. In this experiment, BCAP saves 20% of the calls to the target algorithm.

	Run	N	p_m	p_c	k_{ncs}	k_{cs}	B_{size}	L_s	**UI**
BCAP	1	304	0.677959	0.486372	4	2	0.301837	24	25
	2	307	0.568854	0.211471	4	2	0.400471	6	27
	3	198	0.562451	0.216652	5	2	0.442008	4	27
	4	241	0.685238	0.51544	5	2	0.236686	38	29
	5	364	0.426385	0.47666	3	2	0.184937	4	32
	6	347	0.860843	0.595558	2	2	0.307822	13	35
	7	337	0.59079	0.448577	2	2	0.369539	47	42
	8	331	0.569756	0.560826	2	2	0.362339	10	43
	9	245	0.731057	0.499798	1	2	0.252518	10	49
	10	189	0.695612	0.490825	1	2	0.127177	14	68
IRACE	1	356	0.612855	0.383007	1	4	0.174165	18	21
	2	376	0.672942	0.47128	4	2	0.344001	20	26
	3	383	0.39003	0.421521	1	4	0.179265	2	28
	4	381	0.510213	0.123572	2	3	0.407131	43	34
	5	388	0.502477	0.520349	5	2	0.498641	12	37
	6	386	0.693578	0.132053	2	1	0.277203	3	42
	7	323	0.684316	0.445518	2	2	0.445776	6	43
	8	371	0.707627	0.204171	2	1	0.255995	43	46
	9	373	0.495379	0.401472	4	1	0.24307	27	75
	10	291	0.434215	0.552431	6	1	0.183582	30	77

Table 6.6: Best configuration of each one of the ten independent runs by BCAP and IRACE when configuring GGA-CGT. Here, UI means unsolved instances, i.e., number of instances where the global optimum was not obtained.

6.6 Conclusions of the Chapter

The automated parameter tuning problem (APTP) was introduced and formulated as a bilevel optimization (BO) problems with an optimistic instance, that offers an interesting structure with useful theoretical and practical consequences. A surrogate-assisted BO technique named BCAP was presented to handle this novel BO problem. BCAP was able to deal with the computational load involved with the BO problem thanks to a surrogate model based on radial basis functions. When tackling a collection of recent and complex optimization problems, BCAP and IRACE, a *state-of-the-art* technique for parameter tuning, tuned different metaheuristics well (ABC, ECA, DE, PSO, and GGA-CGT). The Bayesian signed-rank test was used to verify the results. According to the findings, BCAP outperformed IRACE while configuring ABC, ECA, and DE, whereas similar results where obtained when tuning PSO and GGA-CGT. Compared to IRACE, BCAP required just 50% of the calls to the target algorithm for the four metaheuristics for numerical optimization used in the trials. In comparison to IRACE, BCAP required just 80% of the calls to the genetic algorithm for the bin-packing problem. Based on the above findings, we can infer that modeling the APTP problem as a BO problem and solving it using a population-based search technique with a surrogate model is a viable choice to fine-tune search algorithms.

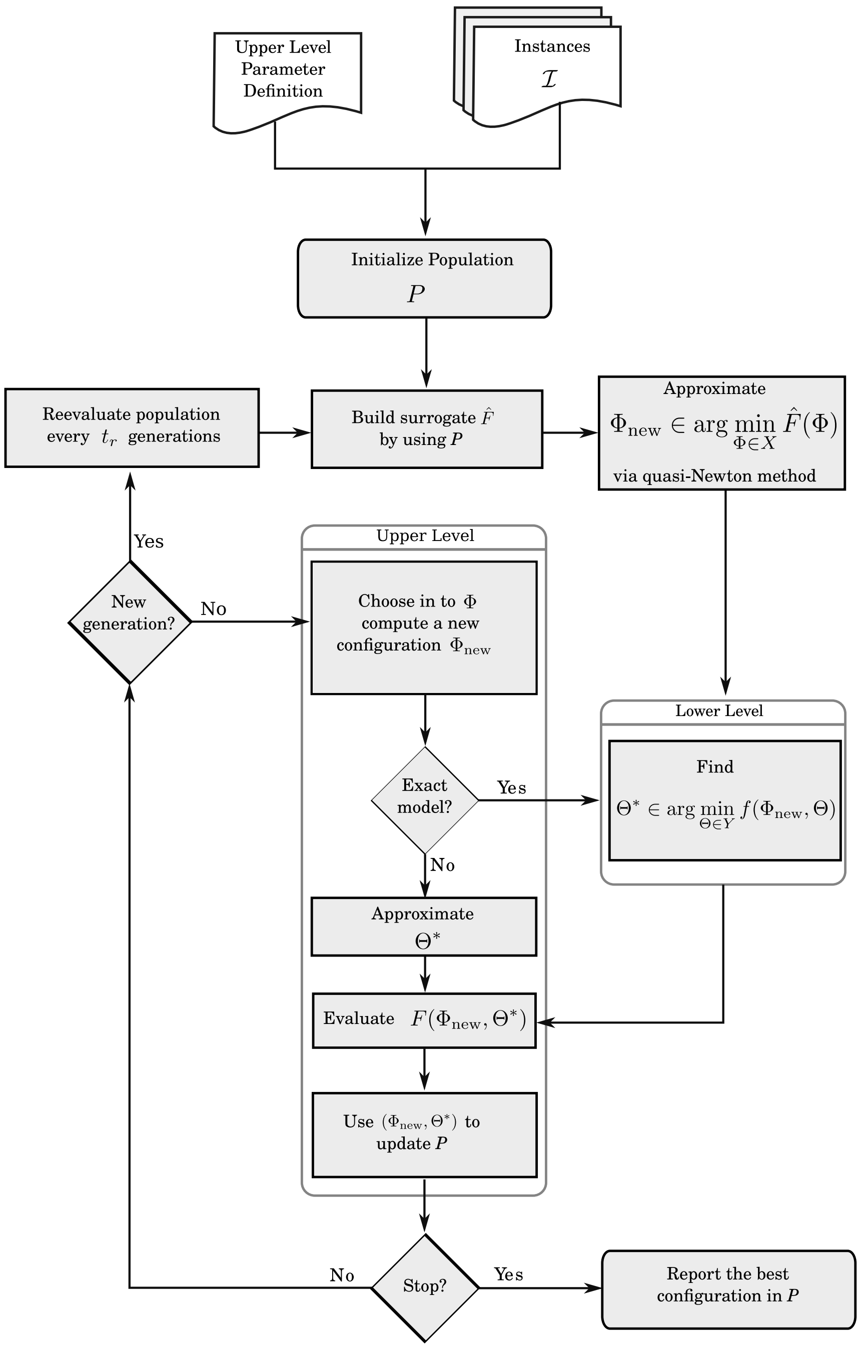

Upper Level Parameter Definition
Instances $\mathcal{I}$
Initialize Population P
Reevaluate population every t_r generations
Build surrogate $\hat{F}$ by using P
Approximate $\Phi_{\text{new}} \in \arg\min_{\Phi\in X} \hat{F}(\Phi)$ via quasi-Newton method
Yes
New generation?
No
Upper Level
Choose in to Φ compute a new configuration Φ_{new}
Exact model?
Yes
No
Lower Level
Find $\Theta^* \in \arg\min_{\Theta\in Y} f(\Phi_{\text{new}}, \Theta)$
Approximate Θ^*
Evaluate $F(\Phi_{\text{new}}, \Theta^*)$
Use $(\Phi_{\text{new}}, \Theta^*)$ to update P
No
Stop?
Yes
Report the best configuration in P

Figure 6.2: BCAP flowchart. The “approximate” step is used to save the computational cost of calling the target algorithm whilst the “reevaluation” stage is used to fix infeasible solutions.

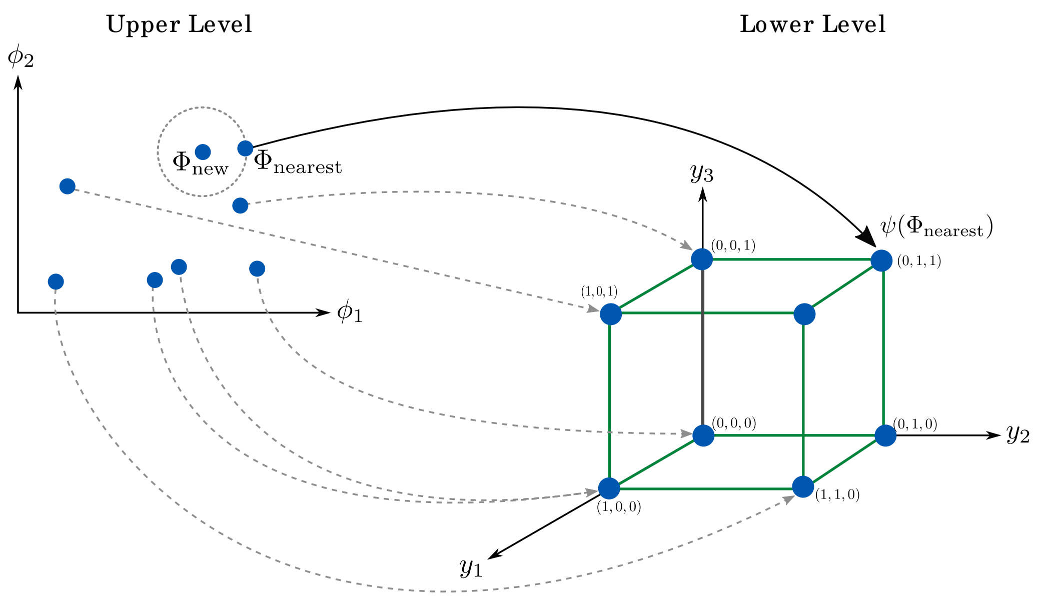


Figure 6.3: Approximation to feasible solutions. In this figure the upper level will only evaluate the instance I_1 and will then save calls to the algorithm to be fine-tuned.

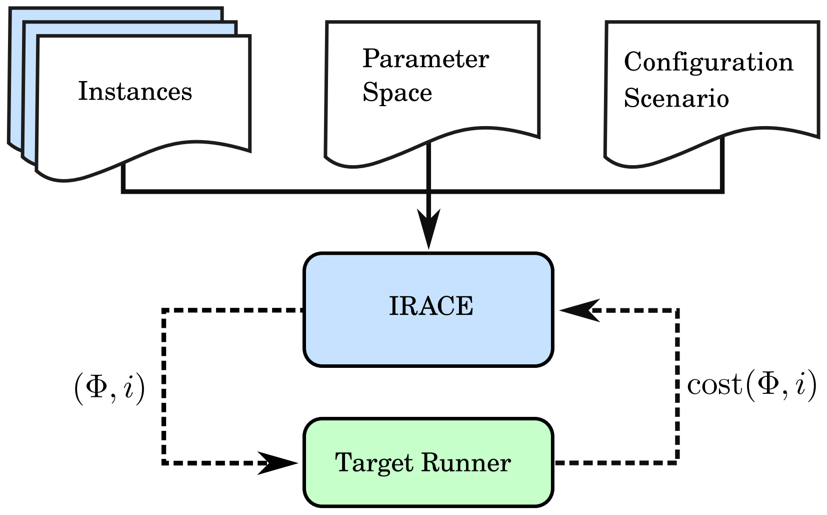


Figure 6.4: Scheme of components required by IRACE to configure an algorithm.

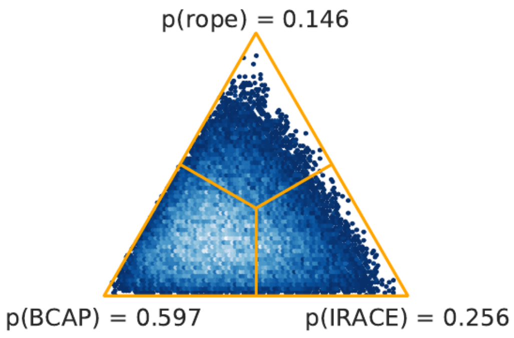


Figure 6.5: BCAP and IRACE Bayesian signed-rank test results when configuring ABC.

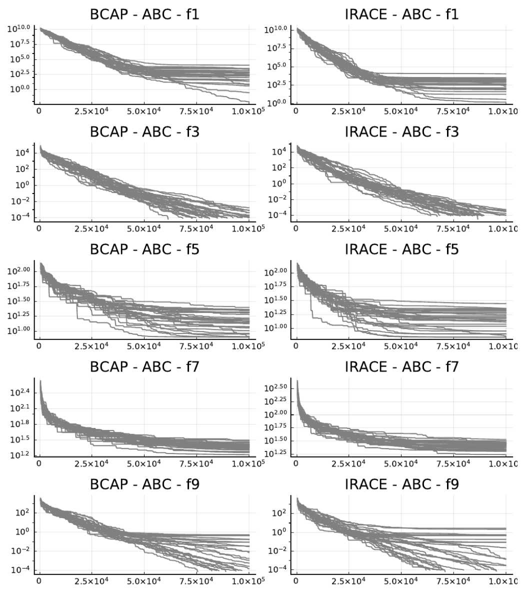


Figure 6.6: Convergence graphs at the median configuration for ABC obtained by BCAP and IRACE. y-axis shows the accuracy in log scale (for visualization purposes) whilst the x-axis shows the number of function evaluations. Each subplot contains the 31 convergence graphs related to the 31 independent runs of ABC. Here, representative test functions are considered.

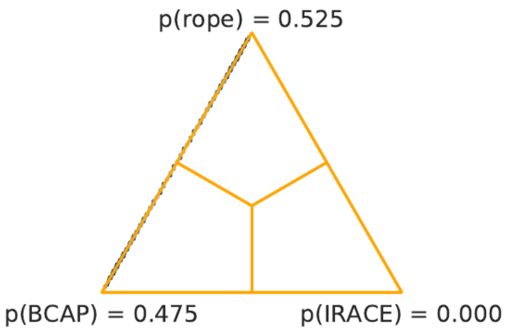


Figure 6.7: BCAP and IRACE Bayesian signed-rank test results when configuring ECA.

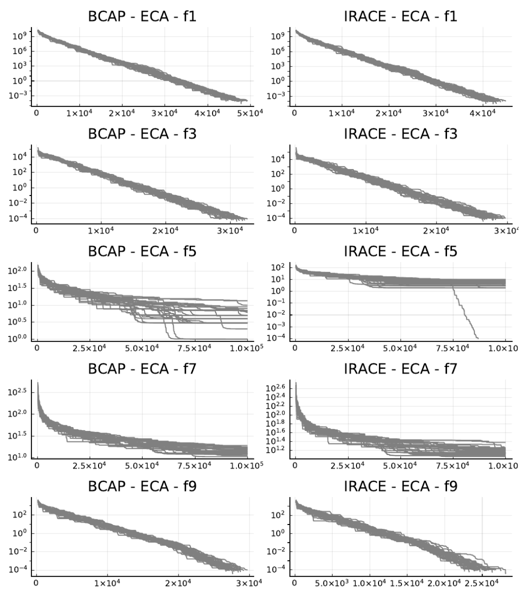


Figure 6.8: Convergence graphs at the median configuration for ECA obtained by BCAP and IRACE. Each subplot contains the 31 convergence graphs related to the 31 independent runs of ECA. y-axis shows the accuracy in log scale (for visualization purposes) whilst the x-axis shows the number of function evaluations. Here, representative test functions are considered.

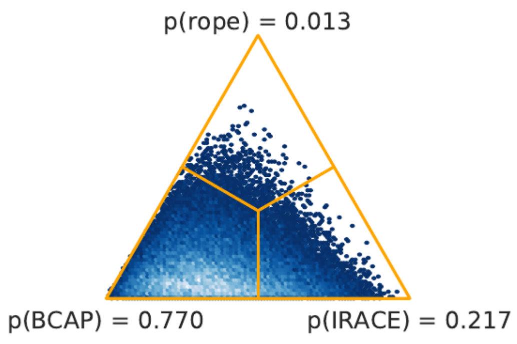


Figure 6.9: BCAP and IRACE Bayesian signed-rank test results when configuring DE.

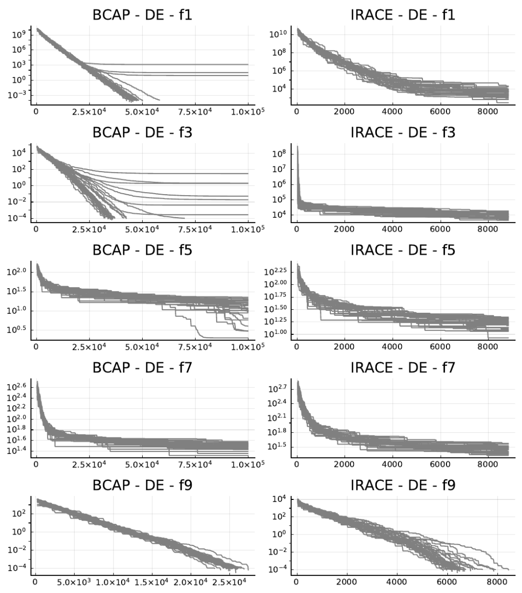


Figure 6.10: Convergence graphs at the median configuration for DE obtained by BCAP and IRACE. y-axis shows the accuracy in log scale (for visualization purposes) whilst the x-axis shows the number of function evaluations. Each subplot contains the 31 convergence graphs related to the 31 independent runs of DE. Here, representative test functions are considered.

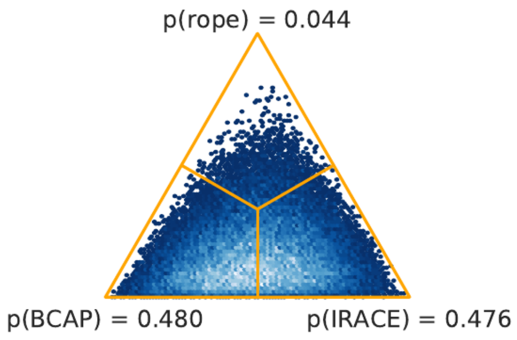


Figure 6.11: BCAP and IRACE Bayesian signed-rank test results when configuring PSO.

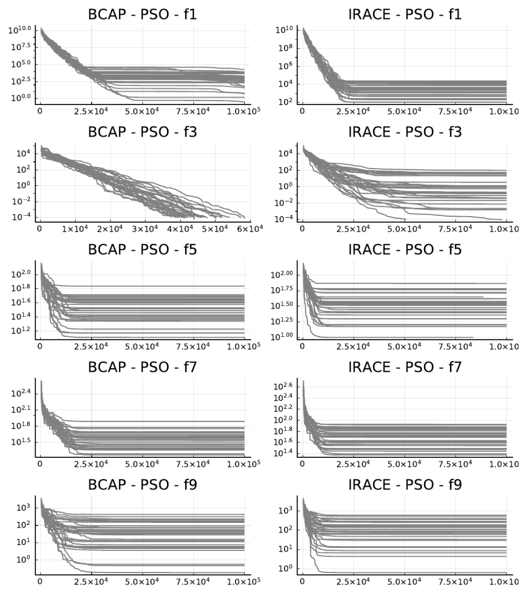


Figure 6.12: Convergence graphs at the median configuration for PSO obtained by BCAP and IRACE. y-axis shows the accuracy in log scale (for visualization purposes) whilst the x-axis shows the number of function evaluations. Each subplot contains the 31 convergence graphs related to the 31 independent runs of PSO. Here, representative test functions are considered.

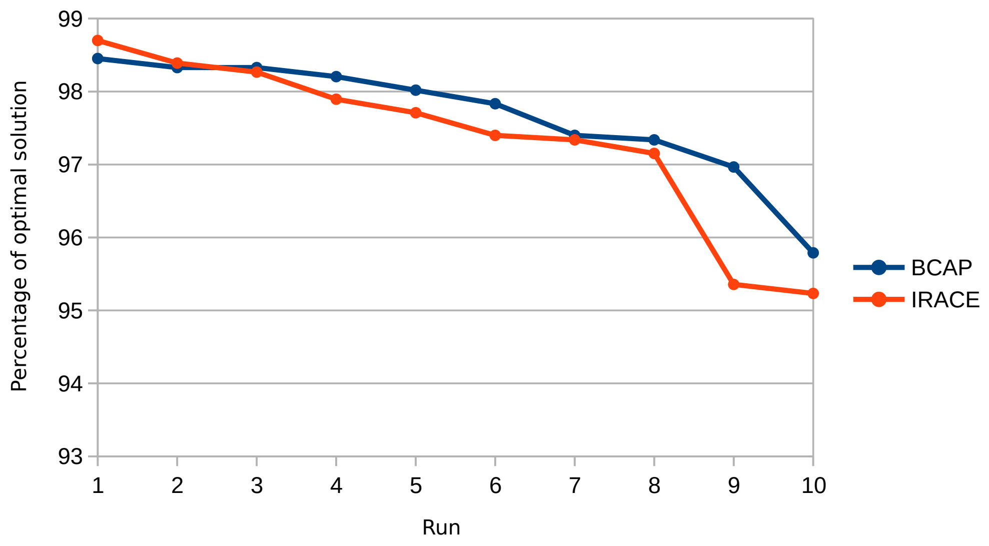


Figure 6.13: Optimal solution percentages when BCAP and IRACE independently configure GGA-CGT to solve 1615 BPP instances (10 independent runs).

Chapter 7

A Family Concept for Multi-Objective Bilevel Optimization

Information

Part of this chapter is based on the paper: Jesús-Adolfo Mejía-de-Dios, Alejandro Rodríguez-Molina and Efrén Mezura-Montes (2022, July). *A novel evolutionary framework based on a family concept for solving multi-objective bilevel optimization problems*. In Proceedings of the Genetic and Evolutionary Computation Conference Companion (pp. 348-351).
`https://doi.org/10.1145/3520304.3529045`

7.1 Introduction

Recalling from Chapter 3, there are competitive algorithms to approximate solutions for a Multi-Objective Bilevel Optimization (MOBO) problem. However, most of the research work just translate Evolutionary Multi-Objective Algorithms (EMOAs) now to deal with MOBO, but specific mechanisms are still to be proposed. In this chapter, a viable solution representation based on groups and called *families* is proposed to (1) eliminate duplication created by UL copied solutions and (2) improve solution ranking. This *family* representation uses the bilevel structure to represent a bilevel solution in an organized manner, to preserve representative and promising UL solutions as well as their associated LL solutions.

Important Concepts

Target of multi-Objective bilevel optimization: Find solutions distributed along the UL Pareto-optimal front such that the LL decision variables are optimizing the LL problem.

The following section details the proposal, defining the solution representation, and introducing the dominance criterion.

7.2 Family-based Representation

The LL of a MOBO problem comprises various solutions that the upper level must manage. As a result, the following strategy is implemented. Let us describe some basic principles for a dominance-based MOBO algorithm [37].

- *Individual*: Here, an individual is denoted by $(\boldsymbol{x}, \boldsymbol{y})$ where $\boldsymbol{x} \in X$ is an UL decision vector and $\boldsymbol{y} \in Y$ is a LL decision vector. A feasible individual is obtained when $\boldsymbol{y} \in \Psi(\boldsymbol{x})$ is an optimal solution for the LL problem parameterized by $\boldsymbol{x}$.

- *Rank*: The rank is a positive integer number indicating the front number where the individual belongs according to the non-dominated sorting in a population (see [35]).

- *Contribution*: Represented by a non-negative real number that indicates how much an individual contributes to the global front (or how much increases the diversity) composed by the non-dominated solutions in the population. Usually, larger values are preferred, e.g., crowding distance [35] or hypervolume [15].

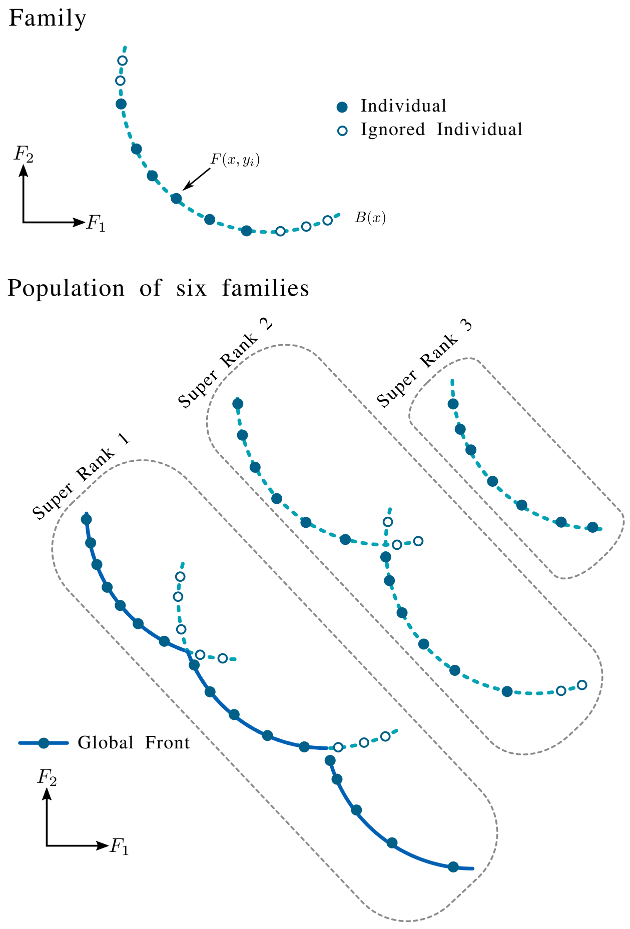


Figure 7.1: Graphical representation of six families, the corresponding super rank, and the ignored solutions.

7.2.1 Family of Solutions

Note that, for a given UL decision vector $\boldsymbol{x}$, the LL can provide multiple optimal solutions, say $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_{N_{ll}}$ associated to the same $\boldsymbol{x}$, then multiple individuals $(\boldsymbol{x}, \boldsymbol{y}_i)$ for $i = 1, 2, \ldots, N_{ll}$ are formed. However, multiple individuals share the same UL decision vector $\boldsymbol{x}$ introducing redundancy. To handle this issue, i.e., to remove redundancy and improve the solutions ranking, the concepts of *Family* and *Super Rank*, inspired by the biological classification of species [98], are introduced.

Definition 7.2.1. (**Family**) Let $\boldsymbol{x} \in X$ be an UL decision vector, then, the set $B(\boldsymbol{x}) \subset X \times Y$ defined as in Eq. 7.1:

$$B(\boldsymbol{x}) := \{(\boldsymbol{x}, \boldsymbol{y}_l) : \boldsymbol{y}_l \in \Psi(\boldsymbol{x}),\ l = 1, 2, \ldots, N_{ll}\} \tag{7.1}$$

is called a *family* concerning $\boldsymbol{x}$.

Note that a family $B(\boldsymbol{x})$ is formed by a finite number of solutions in $(\boldsymbol{x}, \boldsymbol{y}) \in X \times Y$ but with the same UL decision vector $\boldsymbol{x}$. The above is particularly useful because it suggests a data structure. Moreover, when the LL objective function values at each solution in a family correspond to a non-dominated set, i.e., $\{f(\boldsymbol{x}, \boldsymbol{y}) : (\boldsymbol{x}, \boldsymbol{y}) \in B(\boldsymbol{x})\}$ is a non-dominated set, such a property does not occur at the upper level because the set $\{F(\boldsymbol{x}, \boldsymbol{y}) : (\boldsymbol{x}, \boldsymbol{y}) \in B(\boldsymbol{x})\}$ can contain dominated solutions. Also, two different families can contain solutions that dominate each other. That is, there are solutions dominating *intra-family* and *inter-families*. When one solution is dominated by another at the upper level, that solution will be ignored because it is not in the Pareto front. The population of families must be established before comparisons can be made.

Definition 7.2.2. (**Population of families**) A population of families $\boldsymbol{P}$ is defined on a finite set of UL decision vectors:

$$\boldsymbol{P} := \{B(\boldsymbol{x}_i) : \boldsymbol{x}_i \in X,\ i = 1, 2, \ldots, N\}. \tag{7.2}$$

A family-based population includes the following properties: 1) $\boldsymbol{P}$ contains LL optimal solutions, 2) there are at most N different UL decision vectors, and 3) at most $N \times N_{ll}$ solutions in $X \times Y$. Figure 7.1 shows the objective vectors associated with the individuals in a family and also illustrates the distribution of a population of families.

The following concepts are given to compare the quality of two families so as to implement a suitable environmental selection.

7.2.2 Ranking Families

One of the most important aspects of dominance-based algorithms is ranking solutions to ensure that the better solutions remain in the population [35]. A ranking process is used to compare the dominance of two families within a population.

The family rank or *Super Rank* (SR) is based on the well-known non-dominated sorting (NDS) [35]. The main idea is to perform the NDS over every solution in

$$S = \{F(\boldsymbol{x}, \boldsymbol{y}) \in B(\boldsymbol{x}) : \forall B(\boldsymbol{x}) \in \boldsymbol{P}\}, \tag{7.3}$$

and assign the $SR(B)$ as the minimum ND rank to B as follows: $SR(B) = \min\{ND(\boldsymbol{x}, \boldsymbol{y}) : (\boldsymbol{x}, \boldsymbol{y}) \in B(\boldsymbol{x})\}$. Also, because ignored solutions are not part of the UL Pareto optimum front, deleting them is necessary to avoid memory overflow when the population size is large enough. Thus, once the SR is computed for B, the solutions $(\boldsymbol{x}, \boldsymbol{y}) \in B(\boldsymbol{x})$ such that $ND(\boldsymbol{x}, \boldsymbol{y}) > SR(B(\boldsymbol{x}))$ are removed. Algorithm 8 details the procedure to compute the super rank for each family in a population. Moreover, Figure 7.1 also shows how the suggested rank affects the distribution of six families, super rank values, and how ignored solutions are removed from each family.

The computational cost of Algorithm 8 is now analyzed. The non-dominated sorting (line 4 in Algorithm 8) over a set S of size N_S, having M-dimensional objective vectors, requires $O\left(N_S \log^{M-2} N_S\right)$ [76]. Assigning SR to each family and removing ignored solutions (lines 5-7) require $O(N \cdot N_{ll})$ with $N = |\boldsymbol{P}|$ and

$N_{ll} = |B|$ or $O(N_S)$ since $N_S = N \times N_{ll}$ in the worst-case scenario. The overall worst-case complexity of Algorithm 8 is $O\left(N_S \log^{M-2} N_S\right)$. Hence, computing super rank values only depends on the NDS complexity, i.e., no complexity is added by the super rank process.

Algorithm 8: Super rank calculation.

```
1 Input: Population of families $\boldsymbol{P} = \{B(\boldsymbol{x}_i) : i = 1, 2, \ldots, N\}$.
2 Output: Super rank values for each family in $\boldsymbol{P}$.
3 $S \leftarrow \{F(\boldsymbol{x}, \boldsymbol{y}) \in B(\boldsymbol{x}) : \forall B(\boldsymbol{x}) \in \boldsymbol{P}\}$
4 Perform the non-dominated sorting on $S$ to obtain the rank of each
    individual, i.e., $ND(\boldsymbol{x}, \boldsymbol{y}),\ \forall F(\boldsymbol{x}, \boldsymbol{y}) \in S$.
5 foreach $B(\boldsymbol{x})$ in $\boldsymbol{P}$ do
6     $SR(B(\boldsymbol{x})) \leftarrow \min\{ND(\boldsymbol{x}, \boldsymbol{y}) : (\boldsymbol{x}, \boldsymbol{y}) \in B(\boldsymbol{x})\}$.
7     Remove individuals $(\boldsymbol{x}, \boldsymbol{y})$ in $B(\boldsymbol{x})$ such that their ND rank is larger
        that the super rank of the family, i.e., $ND(\boldsymbol{x}, \boldsymbol{y}) > SR(B(\boldsymbol{x}))$.
```

7.3 Proposed Method: SMS-MOBO

This section proposes a simple family-based framework that is built on a generic EMOA. Three primary components are required for a dominance-based EMOA: 1) a ranking algorithm, 2) a density estimator, and 3) operators (selection, crossover, mutation, and environmental selection). Furthermore, to handle the LL task and approximate solutions at that level, a Lower-Level EMOA (LLEMOA) is necessary.

The proposed approach (SMS-MOBO) builds families based on new UL decision vectors that are first produced at random and then reproduced using a variation operator. The super rank notion stated in Section 7.2 is used in the ranking operation. When two families have the same super rank value, a hypervolume-based density estimator is computed to compare solutions. Finally, to save the non-dominated solutions discovered thus far, an archiving process is used. Algorithm 9 summarizes the proposal by describing the main aspects of SMS-MOBO.

7.3.1 Initialization

N UL decision vectors in X are generated randomly. After that, the LLEMOA is performed to approximate a set of LL optimal solutions. Finally, the UL solutions and their corresponding LL optimal solutions create a population of families $\boldsymbol{P}$ as described in Section 7.2.

7.3.2 Reproduction at Upper Level

The following explains how a set Q (of size N_{off}) of UL offspring are created: The super rank (lower SR is better) and contribution to the front in ties are used to compare solutions in a binary tournament (see Section 7.3.5). The SBX crossover and Polynomial Mutation (PM) are adopted to generate new UL solutions once two parents have been chosen. The matching LL solutions for each UL vector in Q are then approximated to generate a new set of families. To promote a biased search at the LL, a set of LL solutions is constructed to initialize the LLEMOA (using random selection, SBX, and PM on the LL vectors in $\boldsymbol{P}$).

7.3.3 Archiving Procedure

To preserve the elite solutions discovered thus far, an archiving approach has been implemented. For this, at the end of each generation, an archive A is updated by saving non-dominated solutions in the current archive A and the population $\boldsymbol{P}$, i.e., A contains non-dominated solutions in the set $A \cup \left(\bigcup_{B \in \boldsymbol{P}} B\right)$.

7.3.4 Lower Level Optimizer

One of the most important components is the LL algorithm that needs to approximate LL optimal solutions for a given UL decision vector (say $\boldsymbol{x}$). The MOP that

Algorithm 9: MOBO: Multi-objective Bilevel Optimization framework based on a family representation.

```
1  **Input**: UL problem $(F, X)$, LL problem $(f, Y)$, population size $N$, and
     LLEMOA (LL optimizer).
2  **Output**: Final archive $A$.
3  *Initialization* (Section 7.3.1): Create UL solutions $\boldsymbol{x}_1, \dots, \boldsymbol{x}_N$ in $X$
     randomly. Optimize $f(\boldsymbol{x}_i, \cdot)$ using LLEMOA and create a family $B(\boldsymbol{x}_i)$ for
     $i = 1, 2, \dots, N$.
4  $\boldsymbol{P} \leftarrow \{B(\boldsymbol{x}_i) : i = 1, 2, \dots, N\}$.
5  *Compute the super rank* for each family in $\boldsymbol{P}$ (Section 7.2.2).
6  *Initialize Archive*: $A \leftarrow \emptyset$
7  **while** *termination criterion not fulfilled* **do**
8      *Reproduction at UL (Section 7.3.2)*: Generate a set $Q$ of new UL
         offspring using $\boldsymbol{P}$.
9      *Solve the LL problem* for each offspring in $Q$ using LLEMOA (Section
         7.3.4).
10     *Create a new set $P_{new}$ of families* using $Q$ and the corresponding
         LLEMOA solutions.
11     *Combine* $\boldsymbol{P}$ and $P_{\text{new}}$, i.e., $\boldsymbol{P} \leftarrow \boldsymbol{P} \bigcup P_{\text{new}}$.
12     *Reduce population* $\boldsymbol{P}$ (see Section 7.3.6).
13     *Update archive* $A$ using $\boldsymbol{P}$ (see Section 7.3.3).
14 **Return** archive $A$.
```

needs to be solved at the LL is

$$\boldsymbol{y}^* \in \arg\min_{\boldsymbol{y} \in Y} \{ f(\boldsymbol{x}, \boldsymbol{y}) = [f_1(\boldsymbol{x}, \boldsymbol{y}), \dots, f_m(\boldsymbol{x}, \boldsymbol{y})]^T : g_j(\boldsymbol{x}, \boldsymbol{y}), j = 1, \dots, J \}$$

where $\boldsymbol{x}$ is fixed while the $\boldsymbol{y}$'s are the decision vectors to be found.

Here, three state-of-the-art multi-objective algorithms are considered for solving this parameterized MOP problem.

NSGA-II

The Non-dominated Sorting Genetic Algorithm [35] (NSGA-II) is one of the most used methods to solve multi-objective problems. In NSGA-II, the search space is randomly seeded with a population of NP members denoted by $\boldsymbol{P}$. Non-

Dominated Sorting (NDS) is used to sort these individuals in order to find the most promising options. NP offspring are produced in each generation. A binary tournament is used to choose couples of parents, who are subsequently recombined using SBX with a given probability. Each offspring can mutate at predetermined rates utilizing PM before being accepted into the population. The NDS and the crowding distance as a density estimator are used as an environmental selection mechanism.

SPEA2

The Strength Pareto Evolutionary Algorithm (SPEA2) [146] incorporates a fitness function that combines the dominance and a density estimator. The density estimator is the closest neighbor strategy, and the optimal solutions identified at each generation are saved using an archiving mechanism. For reproduction, SPEA2 uses the same genetic operations as NSGA-II (SBX crossover and polynomial mutation), whereas for parent and environmental selection, fitness values are utilized to compare the solution quality.

SMS-EMOA

The S-Metric Selection Evolutionary Multi-Objective Algorithm (SMS-EMOA) is a metric-driven optimizer that guides the search for the Pareto front approximation using the hypervolume indicator [15]. To keep only NP members in the population, a reduction procedure is used. The NDS is used in this operator, and the worst-ranked (regarding the hypervolume contribution) front is eliminated.

7.3.5 Density Estimator

There are different ways to compare Pareto fronts in a single-level MOP, as detailed in Section 7.3.4. The super rank principles are used to assess the quality of a family. However, using a density estimator to produce solutions dispersed along the Pareto front is required when a set of non-dominated families is formed.

The hypervolume measure is used to calculate each family's contribution to the front. To quantify this contribution, the hypervolume (HV) indicator introduced in (7.4) is adopted, where A is a set of non-dominated solutions, $\boldsymbol{r}$ is a reference point, and v_k is the hypervolume between the *k-th* solution in A (mapped in the objective functions space) and $\boldsymbol{r}$.

$$HV\left(A, \boldsymbol{r}\right) = \bigcup_{k=1}^{[A]} v_k \tag{7.4}$$

$$\Delta HV(\boldsymbol{x}, \boldsymbol{y}) = HV\left(C^v, \boldsymbol{0}\right) - HV\left(C^v - \left\{F(\boldsymbol{x}_k^v, \boldsymbol{y}_k^v)\right\}, \boldsymbol{0}\right) \tag{7.5}$$

The hypervolume contribution is computed as follows. Let $\boldsymbol{L}$ be a set of families, and assume that $C = \{F(\boldsymbol{x}, \boldsymbol{y}) \in B(\boldsymbol{x}) : \forall B(\boldsymbol{x}) \in \boldsymbol{L}\}$ is non-dominated. Compute the exclusive contribution $\Delta HV(\boldsymbol{x}, \boldsymbol{y})$ of each objective vector in C as in (7.4)-(7.5) (see [15]). Here, the contribution of a family is given by

$$\Delta HV(B) = \sum_{(\boldsymbol{x},\boldsymbol{y}) \in B(\boldsymbol{x})} \Delta HV(\boldsymbol{x}, \boldsymbol{y}). \tag{7.6}$$

In other words, a family's contribution is the sum of the contributions made by its solutions to the non-dominated set C. Figure 7.2 represents how the sum of contributions (7.4)-(7.5) is calculated for three different families with same SR value.

7.3.6 Population Reduction

Let us assume that a population of families $\boldsymbol{P}$ of size $|\boldsymbol{P}| > N$ needs to be truncated to save the N best solutions. First, a ranking based on the SR values is computed. After that, the families with a minimum contribution to the front are eliminated. Algorithm 10 summarizes this procedure which is based on the hypervolume indicator.

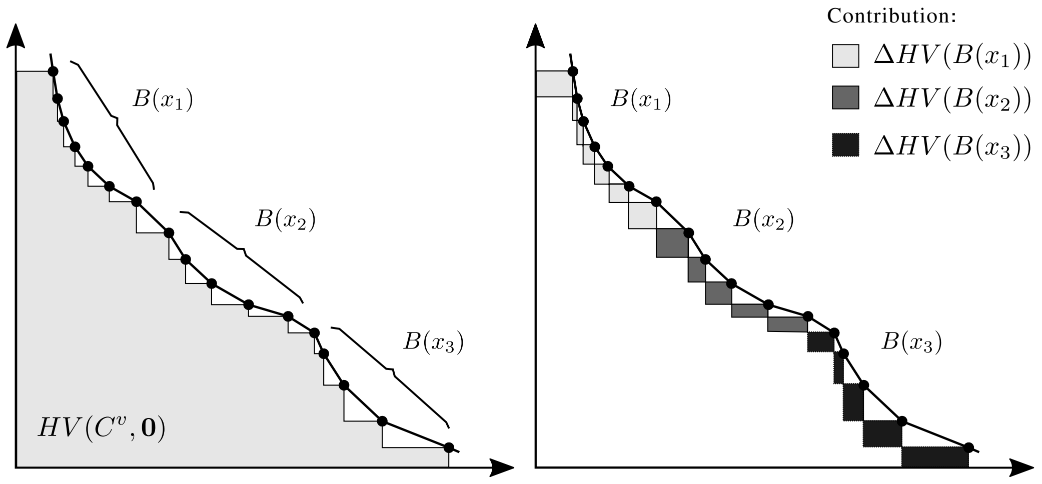


Figure 7.2: Contribution of three different families with the same SR value.

Algorithm 10: Reduce population of families that have the same SR value.

```
1  **Input**: Population $\boldsymbol{P} = \{B_i = B(\boldsymbol{x}_i) : i = 1, 2, \ldots, |\boldsymbol{P}|\}$ such that $|\boldsymbol{P}| > N$

2  **Output**: Reduced population of families with size $N$.
3  Compute SR values for each family in $\boldsymbol{P}$ using Algorithm 8.
4  Compute subfronts: $\boldsymbol{R}_j \leftarrow \{B \in \boldsymbol{P} : SR(B) = j\}$ for $j = 1, 2, \ldots, SR_{\max}$.
5  $S \leftarrow \emptyset, j \leftarrow 1$.
6  **while** $|S| \leq N$ **do**
7      $S \leftarrow S \cup \boldsymbol{R}_j, j \leftarrow j + 1$.
8  **if** $|\boldsymbol{P}| = N$ **then**
9      Return $\boldsymbol{P}$.
10 **else**
11     Put last front as $\boldsymbol{L} \leftarrow R_j$
12     Families to be chosen from $\boldsymbol{L}$: $K = N - |S|$.
13     **while** $|\boldsymbol{L}| > K$ **do**
14         For each family $B$ in $\boldsymbol{L}$, compute their contributions $\Delta HV(B)$ (see
             Section 7.3.5).
15         Delete family in $\boldsymbol{L}$ with minimum contribution $\Delta HV$.
16     $\boldsymbol{P} \leftarrow \left(\bigcup_{k=1}^{j-1} S_k\right) \cup \boldsymbol{L}$.
```

7.4 Experiments and Discussion

This section tests the proposed SMS-MOBO (Algorithm 9) by implementing three variants of the algorithm and comparing them to BLEMO on six constrained-multi-objective bilevel optimization problems [37, 36]. The hypervolume (HV) is used as the indicator to compare results.

7.4.1 Parameter Settings

Problems:

The benchmark considered here contains six constrained MOBO scenarios denoted as TP1, TP2, DS1, DS2, DS3, and DS4. TP1 is a three-dimensional problem with two inequality constraints (one at each level) and a non-convex optimum front. TP2 is a 15-dimensional problem with the follower problem as the only constraint. Because DS1 to DS4 are scalable test problems, the 10D configuration is utilized in all but DS4, which uses the 5D variant.

Algorithms:

BLEMO is the algorithm adopted for comparison purposes and its parameters are set as suggested in its original paper, i.e., the population size at the UL is set to $N_{ul} = 400$ and $N_{ll} = 40$ at the LL, $p_c = 0.9$ for the SBX crossover with $\eta_c = 20$, and $p_m = 0.1$ for the polynomial mutation with $\eta_m = 15$. The crossover and mutation parameters are the same as in SMS-MOBO, but the population size is different because our approach employs a different representation of solutions. Regarding SMS-MOBO, a population size of $N = 40$ is recommended for TP1, DS2, and DS3; $N = 10$ for TP2, and $N = 20$ for DS1 and DS4. The LL population size is set as $N_{ll} = 80$ for TP1, DS1, and DS3; $N_{ll} = 40$ for the remaining problems, but with $N_{off} = 10$ as in BLEMO for all experiments. It is worth mentioning that the parameters for SMS-MOBO have been selected using a trial-and-error strategy, suggesting that higher values for the population are recommended when multiple solutions from the lower level promote the upper-level diversity of solutions.

Here, we use the constraint handling based on the Constraint Violation Sum (CVS). That is, between two solutions, prefer the solution with the minimum CVS, if both solutions have the same CVS, then compare them using Pareto-dominance [95].

To ensure a fair comparison between SMS-MOBO and BLEMO, both algorithms have the same number of function evaluations: 80000 for TP1-TP2 and DS4, 300000 for DS1 and DS3, and 160000 for DS2.

7.4.2 Experimental Results

The experiments are carried out considering the hypervolume values at the UL. To compare the performance of the proposed algorithms (SMS-MOBO variants against BLEMO), the Kruskal-Wallis rank sum test ($\alpha = 0.05$) is used to validate the results.

Due to solving multi-objective bilevel problems requiring high computational effort and resources, 11 independent runs have been performed here; however, it is recommended to perform a larger number of executions. Table 7.1 shows the median of the hypervolume values generated by SMS-MOBO in the three versions (changing the LLEMOA among NSGA-II, SPEA2, and SMS-EMOA) and BLEMO on the test scenarios.

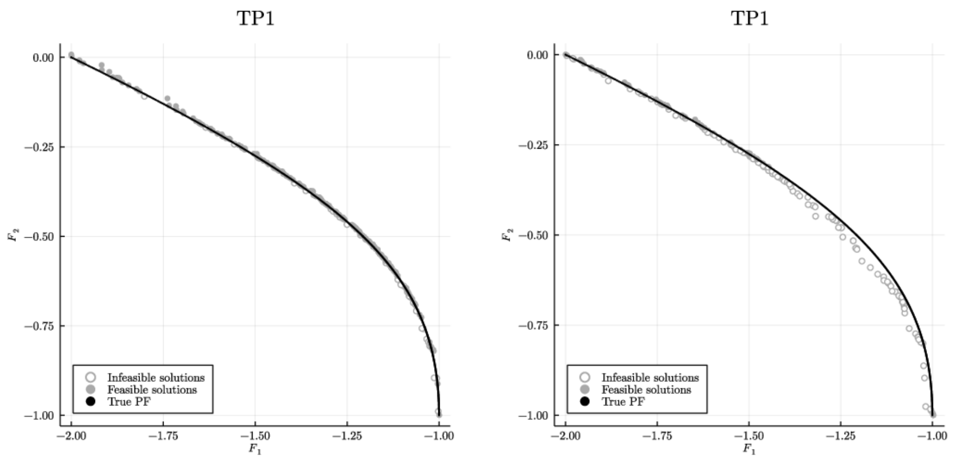
SMS-MOBO/SMS-EMOA outperforms BLEMO in TP1 and the high-dimensional problem TP2, as shown in Table 7.1, suggesting that the SMS-EMOA used as a LL optimizer can be a suitable choice if the interest problem has similar properties.

Furthermore, with the exception of test problem DS4, our approach is competitive in most test problems against BLEMO (as suggested by the Kruskal-Wallis rank sum test). In DS4, the Pareto-optimal front sits within the UL constraint's borders, making it a difficult test problem to solve because better solutions may require a larger population.

Figures 7.3-7.8 show the obtained non-dominated solutions for all problems by the SMS-MOBO/SMS-EMOA variant and BLEMO, respectively. Note that only the plots obtained by the SMS-MOBO/SMS-EMOA variant are presented here for visualization purposes.

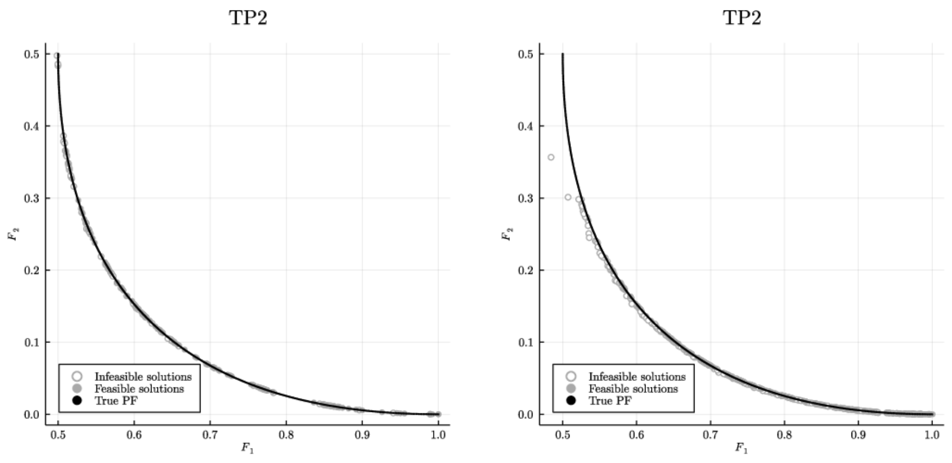
	NSGA-II	SPEA2	SMS-EMOA	BLEMO	p-value
TP1	0.302	0.302	**0.302**	0.206	1.65E-05
TP2	0.202	0.194	**0.206**	0.197	2.86E-06
DS1	0.95	0.95	0.95	0.95	0.643
DS2	0.53	0.531	0.532	0.532	0.321
DS3	1.02	1.02	1.01	1.01	0.155
DS4	0.986	0.985	0.985	**0.992**	2.18E-05

Table 7.1: Overall comparison of SMS-MOBO variants against BLEMO. Results in **bold** indicate that significant-differences are observed (concerning the Kruskal-Wallis rank sum test).



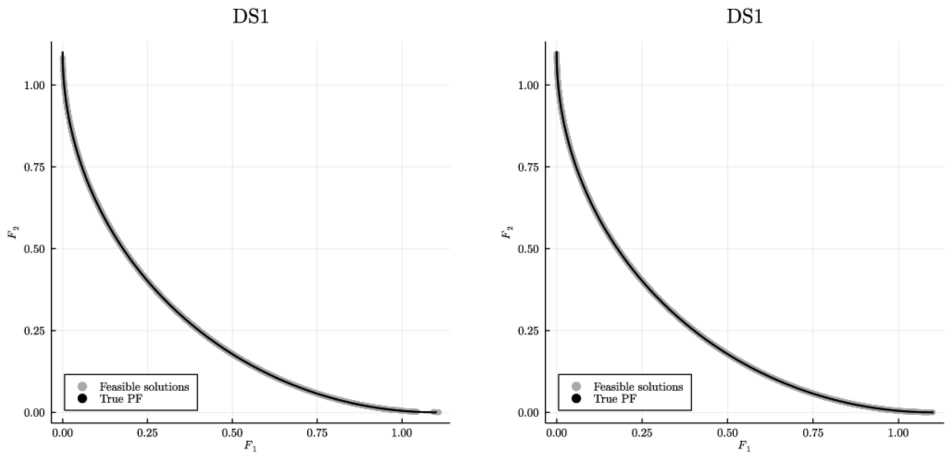
SMS-MOBO/SMS-EMOA BLEMO

Figure 7.3: Upper-level front at the median HV value by SMS-MOBO and BLEMO solving problem TP1.



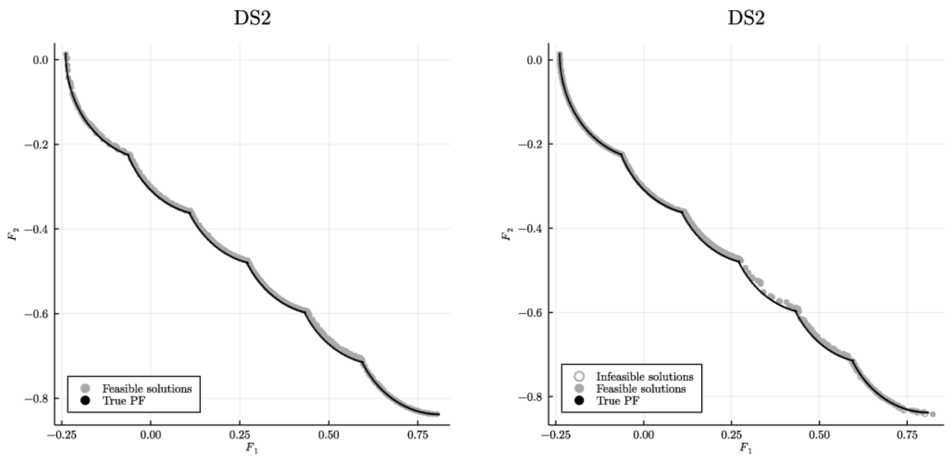
SMS-MOBO/SMS-EMOA BLEMO

Figure 7.4: Upper-level front at the median HV value by SMS-MOBO and BLEMO solving problem TP2.



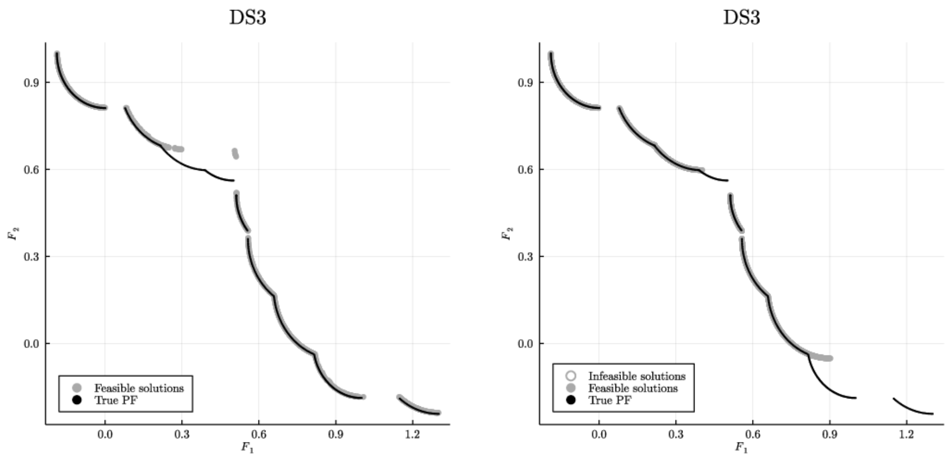
SMS-MOBO/SMS-EMOA BLEMO

Figure 7.5: Upper-level front at the median HV value by SMS-MOBO and BLEMO solving problem DS1.



SMS-MOBO/SMS-EMOA BLEMO

Figure 7.6: Upper-level front at the median HV value by SMS-MOBO and BLEMO solving problem DS2.

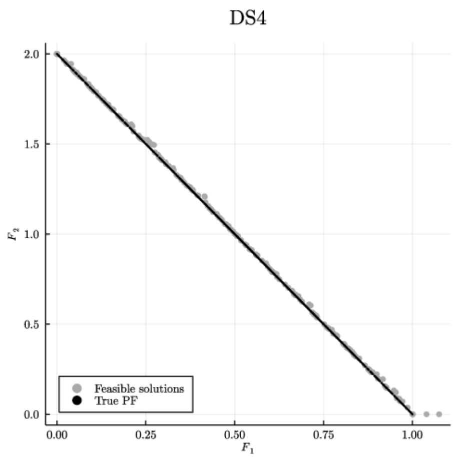


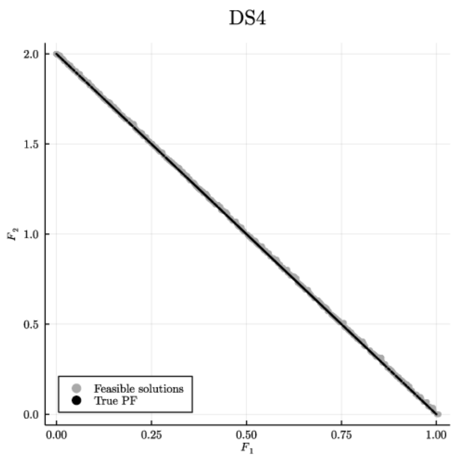
SMS-MOBO/SMS-EMOA BLEMO

Figure 7.7: Upper-level front at the median HV value by SMS-MOBO and BLEMO solving problem DS3.

7.5 Conclusions of the Chapter

In this chapter, a representation based on a family concept was proposed to solve constrained multi-objective bilevel optimization (MOBO) problems. The bilevel





SMS-MOBO/SMS-EMOA BLEMO

Figure 7.8: Upper-level front at the median HV value by SMS-MOBO and BLEMO solving problem DS4.

structure and the multiple objectives at each level were considered in this part of the experimental study. As a result, the traditional non-dominated sorting was utilized to eliminate redundancy (avoiding cloned solutions) among UL decision vectors. Besides, for the family representation, a density estimator based on the hypervolume indicator (also known as S-metric) was successfully applied to compare the quality among a set of families. On representative constrained MOBO problems, the S-metric selection MOBO algorithm (SMS-MOBO) was tested. Statistical results suggested that SMS-MOBO provided competitive results for solving multi-objective bilevel instances.

Conclusions and Future Work

This part contains the general conclusions of this thesis. The first two Chapters have been used to give the theoretical background about bilevel optimization and the solution approaches for different scenarios; the rest of the chapters were dedicated to presenting the findings. Firstly a nested approach was presented as the Bilevel Centers Algorithm (BCA) to address single-objective bilevel optimization via benchmark instances. BCA was presented as a framework that just requires three parameters, (1) the population size, (2) the size of the subset to compute the center of mass, and (3) the step size used to control the convergence acceleration. Regarding accepting hypothesis #1 (in Section 1.2), the results suggested that BCA was able to produce comparable accuracy results to those produced by state-of-the-art algorithms. However, BCA required fewer function evaluations at the upper level, making it a suitable algorithm for bilevel problems with a computationally expensive upper-level problem. Furthermore, BCA was improved by incorporating mechanisms to avoid pseudo-feasible solutions. The experimental results revealed that BCA outperformed two competitive evolutionary algorithms because BCA was able to provide feasible solutions close to the true optimum whilst pseudo-feasible solutions were spotted in most of the cases in comparison with the other approaches (which implied accepting hypothesis #2 given in Section 1.2).

Moreover, it was observed that population-based algorithms can solve unimodal problems, mainly when the objective functions at both levels are not conflicting. On the other hand, bilevel evolutionary algorithms can converge to local optima when the upper or lower level present multimodality (multiple local optima), leading them to report misleading solutions or converge to infeasible regions in the search space. The findings suggested that bilevel algorithms can converge to feasible local optima in upper-level multimodal problems (unimodal lower level); however, infeasible or pseudo-feasible solutions can be reported for multimodal lower-level problems. Hence, a suitable mechanism can be used for the upper or lower level to improve the performance of the algorithms.

The BCA framework was extended to address the automated parameter tuning problem modeled as a bilevel optimization task. The proposed method was used to deal with the computational load related to the BO problem thanks to an incorporated surrogate model based on radial basis functions. According to the findings, the proposed method statistically outperformed a popular procedure to configure algorithms known as IRACE. Moreover, this extended BCA reduced the computational usage in comparison to IRACE.

Regarding hypothesis #3, on the study on multi-objective bilevel optimization, a representation based on a family concept was proposed to solve constrained multi-objective bilevel optimization (MOBO) problems. The bilevel structure and multiple objectives at each level were considered in this part of the experimental study. As a result, the traditional non-dominated sorting was utilized to eliminate redundancy (avoiding cloned solutions) among UL decision vectors. Besides, for the family representation, a density estimator based on the hypervolume indicator (also known as S-metric) was successfully applied to compare the quality among a set of families. On representative constrained MOBO problems, the S-metric selection MOBO algorithm (SMS-MOBO) was investigated. Statistical results suggested that SMS-MOBO provided competitive results for solving multi-objective bilevel instances. These results, allowed accepting hypothesis #3 given Section 1.3. It was observed that multi-objective bilevel problems are challenging because the upper-level problem depends on the lower level optimality to reach feasibility and diversity of solutions. If the lower-level optimizer fails to converge to optimal solutions, then the upper-level optimizer will report infeasible solutions. On the contrary, if the lower level reports optimal solutions, but with low diversity of solutions, then the upper-level optimizer will be unable to generate well-distributed solutions along the upper-level Pareto-optimal front.

Future paths of research include the study and application of bilevel models and solution approaches to important concerns than contains an obvious hierarchical structure. The following list includes possible issues than can be addressed by using bilevel optimization.

- Constraint handling in single- and multi-objective bilevel problems can be improved, particularly when the lower-level optimizer provides inexact lower-level optimal solutions.
- Use the algorithm proposed in Chapter 6 to find the best parameters for machine-learning algorithms besides configuring other metaheuristics.
- The lower-level optimizer can offer a variety of solutions from a Pareto-optimal set in multi-objective bilevel problems. Improved constraint-handling techniques can be proposed to select preferred solutions by the decision-maker. Possible ideas include the usage of compromise programming at a lower-level.
- Finally, real-world problems (e.g. mechanical design, path-planing, among others) can be modeled and solved using bilevel optimization.

Bibliography

[1] B. Akay and D. Karaboga. Parameter tuning for the artificial bee colony algorithm. In *International Conference on Computational Collective Intelligence*, pages 608–619. Springer, 2009.

[2] Maria João Alves and Carlos Henggeler Antunes. A differential evolution algorithm to semivectorial bilevel problems. In Giuseppe Nicosia, Panos Pardalos, Giovanni Giuffrida, and Renato Umeton, editors, *Machine Learning, Optimization, and Big Data*, pages 172–185, Cham, 2018. Springer International Publishing.

[3] Maria João Alves, Stephan Dempe, and Joaquim J. Júdice. Computing the pareto frontier of a bi-objective bi-level linear problem using a multiobjective mixed-integer programming algorithm. *Optimization*, 61(3):335–358, 2012.

[4] M. Andersson, S. Bandaru, and A. H.C. Ng. Tuning of multiple parameter sets in evolutionary algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, pages 533–540. ACM, 2016.

[5] M. Andersson, S. Bandaru, A. H.C. Ng, and A. Syberfeldt. Parameter tuned CMA-ES on the CEC'15 expensive problems. In *2015 IEEE Congress on Evolutionary Computation (CEC)*, pages 1950–1957. IEEE, 2015.

[6] J. S. Angelo and H. J. C. Barbosa. Differential evolution to find Stackelberg-Nash equilibrium in bilevel problems with multiple followers. In *IEEE Congress on Evolutionary Computation (CEC 2015)*, pages 1675–1682. IEEE, 2015.

[7] J. S. Angelo, E. Krempser, and H. J. C. Barbosa. Differential evolution for bilevel programming. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 470–477. IEEE, 2013.

[8] Jaqueline S Angelo, Eduardo Krempser, and Helio JC Barbosa. Performance evaluation of local surrogate models in bilevel optimization. In *International Conference on Machine Learning, Optimization, and Data Science*, pages 347–359. Springer, 2019.

[9] J. M. Arroyo. Bilevel programming applied to power system vulnerability analysis under multiple contingencies. *IET generation, transmission & distribution*, 4(2):178–190, 2010.

[10] N.H. Awad, M.Z. Ali, B.Y. Q., J.J. Liang, and P.N. Suganthan. Problem Definitions and Evaluation Criteria for the CEC 2017 Special Session and Competition on Single Objective Bound Constrained Real-Parameter Numerical Optimization. Technical report, Technological University, Singapore, Tech. Rep., 2016.

[11] T. Back. Selective pressure in evolutionary algorithms: A characterization of selection mechanisms. In *IEEE World Congress on Computational Intelligence*, pages 57–62. Citeseer, 1994.

[12] Thomas Back. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, 1996.

[13] J. F. Bard. *Practical bilevel optimization: algorithms and applications*, volume 30. Springer Science & Business Media, 2013.

[14] Slim Bechikh, Rituparna Datta, and Abhishek Gupta. *Recent advances in evolutionary multi-objective optimization*, volume 20. Springer, 2016.

[15] Nicola Beume, Boris Naujoks, and Michael Emmerich. Sms-emoa: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669, 2007.

[16] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017.

[17] W. F. Bialas and M. H. Karwan. Two-level linear programming. *Management science*, 30(8):1004–1020, 1984.

[18] M. Birattari and J. Kacprzyk. *Tuning metaheuristics: a machine learning perspective*, volume 197. Springer, 2009.

[19] A. Biswas, K.K. Mishra, S. Tiwari, and A.K. Misra. Physics-inspired optimization algorithms: A survey. *Journal of Optimization*, vol. 2013, 2013.

[20] Papun Biswas. *Genetic Algorithm Based Multiobjective Bilevel Programming for Optimal Real and Reactive Power Dispatch Under Uncertainty*, pages 171–203. Springer International Publishing, Cham, 2015.

[21] Ilhem BoussaïD, Julien Lepagnot, and Patrick Siarry. A survey on optimization metaheuristics. *Information Sciences*, 237:82–117, 2013.

[22] J. P. Boyd and F. Xu. Divergence (Runge phenomenon) for least-squares polynomial approximation on an equispaced grid and Mock–Chebyshev subset interpolation. *Applied Mathematics and Computation*, 210(1):158–168, 2009.

[23] L. Brotcorne, M. Labbé, P. Marcotte, and G. Savard. A bilevel model for toll optimization on a multicommodity transportation network. *Transportation Science*, 35(4):345–358, 2001.

[24] J. Carrasco, S. García, M.M. Rueda, S. Das, and F. Herrera. Recent trends in the use of statistical tests for comparing swarm and evolutionary com-

puting algorithms: Practical guidelines and a critical review. *Swarm and Evolutionary Computation*, 54:100665, 2020.

[25] Soumitri Chattopadhyay, Aritra Marik, and Rishav Pramanik. A brief overview of physics-inspired metaheuristic optimization techniques. *arXiv preprint arXiv:2201.12810*, 2022.

[26] Y. Chen and M. Florian. On the geometric structure of linear bilevel programs: a dual approach. *Centre de Recherche Sur les Transports Publication*, (867), 1992.

[27] CA Coello Coello. Evolutionary multi-objective optimization: a historical view of the field. *IEEE computational intelligence magazine*, 1(1):28–36, 2006.

[28] Jared L Cohon. *Multiobjective programming and planning*, volume 140. Courier Corporation, 2004.

[29] B. Colson, P. Marcotte, and G. Savard. An overview of bilevel optimization. *Annals of operations research*, 153(1):235–256, 2007.

[30] R Courant and J Fritz. Introduction to calculus and analysis, vol. ii/1 and vol. ii/2, 1998.

[31] S. Damas, O. Cordón, and J. Santamaría. Medical image registration using evolutionary computation. *IEEE Computational Intelligence Magazine*, 6(4):26–42, 2011.

[32] S. Das, S.S. Mullick, and P.N. Suganthan. Recent advances in differential evolution – an updated survey. *Swarm and Evolutionary Computation*, 27:1–30, 2016.

[33] Jesús-Adolfo Mejía de Dios, Efrén Mezura-Montes, and Porfirio Toledo-Hernández. Pseudo-feasible solutions in evolutionary bilevel optimization: Test problems and performance assessment. *Applied Mathematics and Computation*, 412:126577, 2022.

[34] Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and Tanaka Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In *International conference on parallel problem solving from nature*, pages 849–858. Springer, 2000.

[35] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[36] Kalyanmoy Deb and Ankur Sinha. Solving bilevel multi-objective optimization problems using evolutionary algorithms. In *International conference on evolutionary multi-criterion optimization*, pages 110–124. Springer, 2009.

[37] Kalyanmoy Deb and Ankur Sinha. An efficient and accurate solution methodology for bilevel multi-objective programming problems using a hybrid evolutionary-local-search algorithm. *Evolutionary computation*, 18(3):403–449, 2010.

[38] S. Dempe. *Foundations of Bilevel Programming*. Springer Science & Business Media, 2002.

[39] S. Dempe and J. Dutta. Is bilevel programming a special case of a mathematical program with complementarity constraints? *Mathematical Programming*, 131(1-2):37–48, February 2010.

[40] J. Derrac, S. García, D. Molina, and F. Herrera. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1(1):3–18, 2011.

[41] A. E. Eiben and S. K. Smit. Evolutionary algorithm parameters and methods to tune them. In *Autonomous search*, pages 15–36. Springer, 2011.

[42] A. E. Eiben and S. K. Smit. Parameter tuning for configuring and analyzing evolutionary algorithms. *Swarm and Evolutionary Computation*, 1(1):19–31, 2011.

[43] Gabriele Eichfelder. Multiobjective bilevel optimization. *Mathematical Programming*, 123(2):419–449, 2010.

[44] M. A. El-Beltagy, P. B. Nair, and A. J. Keane. Metamodeling techniques for evolutionary optimization of computationally expensive problems: Promises and limitations. In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 1*, pages 196–203. Morgan Kaufmann Publishers Inc., 1999.

[45] Maha Elarbi, Slim Bechikh, Lamjed Ben Said, and Rituparna Datta. Multi-objective optimization: Classical and evolutionary approaches. In *Recent advances in evolutionary multi-objective optimization*, pages 1–30. Springer, 2017.

[46] J.F. Epperson. On the runge example. *The American Mathematical Monthly*, 94(4):329–341, 1987.

[47] E. Fernandez, C. Gomez, G. Rivera, and L. Cruz-Reyes. Hybrid metaheuristic approach for handling many objectives and decisions on partial support in project portfolio optimisation. *Information Sciences*, 315:102–122, 2015.

[48] P.J. Fleming and R.C. Purshouse. Evolutionary algorithms in control systems engineering: a survey. *Control engineering practice*, 10(11):1223–1241, 2002.

[49] Roger Fletcher. *Practical methods of optimization*. John Wiley & Sons, 2013.

[50] Jesse Frey. Introduction to stochastic search and optimization: Estimation, simulation, and control. *Journal of the American Statistical Association*, 99(468):1204–1205, December 2004.

[51] Roger Gamperle, Sibylle D. Muller, and Petros Koumoutsakos. A parameter study for differential evolution. In *WSEAS Int. Conf. on Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation*, pages 293–298. Press, 2002.

[52] Fuchang Gao and Lixing Han. Implementing the Nelder-Mead simplex algorithm with adaptive parameters. *Computational Optimization and Applications*, 51(1):259–277, May 2010.

[53] Yong-Feng Ge, Wei-Jie Yu, Jinli Cao, Hua Wang, Zhi-Hui Zhan, Yanchun Zhang, and Jun Zhang. Distributed memetic algorithm for outsourced database fragmentation. *IEEE Transactions on Cybernetics*, 51(10):4808–4821, 2021.

[54] Fred Glover. Future paths for integer programming and links to artificial intelligence. *Computers & operations research*, 13(5):533–549, 1986.

[55] P. Hansen, B. Jaumard, and G. Savard. New branch-and-bound rules for linear bilevel programming. *SIAM Journal on scientific and Statistical Computing*, 13(5):1194–1217, 1992.

[56] X. He, Y. Zhou, and Z. Chen. Evolutionary bilevel optimization based on covariance matrix adaptation. *IEEE Transactions on Evolutionary Computation*, 2018.

[57] L. Hecheng and W. Yuping. An evolutionary algorithm based on a new decomposition scheme for nonlinear bilevel programming problems. *International Journal of Communications, Network and System Sciences*, 3(01):87, 2010.

[58] Sergio Hernández, Guillem Duran, and José M. Amigó. Physics-inspired swarm optimization: The general algorithmic search. In *World Scientific Series on Nonlinear Science Series B*, pages 531–550. WORLD SCIENTIFIC, July 2021.

[59] H. H. Hoos. Automated algorithm configuration and parameter tuning. In *Autonomous search*, pages 37–71. Springer, 2011.

[60] F. Hutter and H. H. Hoos. Automatic algorithm configuration based on local search. In *Proceedings of the 22nd national conference on artificial intelligence Vol. 2*, pages 1152–1157. AAAI Press, 2007.

[61] F. Hutter, H. H. Hoos, K. Leyton-Brown, and T. Stützle. Paramils: an automatic algorithm configuration framework. *Journal of Artificial Intelligence Research*, 36:267–306, 2009.

[62] Hisao Ishibuchi, Ryo Imada, Naoki Masuyama, and Yusuke Nojima. Comparison of hypervolume, IGD and IGD+ from the viewpoint of optimal distributions of solutions. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 332–345. Springer, 2019.

[63] Hisao Ishibuchi, Ryo Imada, Yu Setoguchi, and Yusuke Nojima. Reference point specification in inverted generational distance for triangular linear pareto front. *IEEE Transactions on Evolutionary Computation*, 22(6):961–975, 2018.

[64] Md Monjurul Islam, Hemant Kumar Singh, and Tapabrata Ray. A surrogate assisted approach for single-objective bilevel optimization. *IEEE Transactions on Evolutionary Computation*, 21(5):681–696, 2017.

[65] Md Monjurul Islam, Hemant Kumar Singh, and Tapabrata Ray. Efficient global optimization for solving computationally expensive bilevel optimization problems. In *2018 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE, 2018.

[66] Md Monjurul Islam, Hemant Kumar Singh, Tapabrata Ray, and Ankur Sinha. An enhanced memetic algorithm for single-objective bilevel optimization problems. *Evolutionary computation*, 25(4):607–642, 2017.

[67] Ying Ji, Gang Ma, Ju Wei, and Yeming Dai. A hybrid approach for uncertain multi-criteria bilevel programs with a supply chain competition application. *Journal of Intelligent & Fuzzy Systems*, 33(5):2999–3008, 2017.

[68] Ying Ji, Shaojian Qu, and Zhensheng Yu. A new method for solving multi-objective bilevel programs. *Discrete Dynamics in Nature and Society*, 2017, 2017.

[69] Hongmei Jia, Bin Wang, Lin Zhang, and Yanping Liu. A bilevel time-dependent scheduling model for hazmat road transportation. *IET Road*

Transport Information and Control Conference and the ITS United Kingdom Members' Conference (RTIC 2010). Better transport through technology, 2010.

[70] Liping Jia and Yuping Wang. Genetic algorithm based on primal and dual theory for solving multiobjective bilevel linear programming. In *2011 IEEE Congress of Evolutionary Computation (CEC)*, pages 558–565, 2011.

[71] M. Jiang, Y. P Luo, and S. Y. Yang. Stochastic convergence analysis and parameter selection of the standard particle swarm optimization algorithm. *Information processing letters*, 102(1):8–16, 2007.

[72] S. Jiang, Z. Ji, and Y. Shen. A novel hybrid particle swarm optimization and gravitational search algorithm for solving economic emission load dispatch problems with various practical constraints. *International Journal of Electrical Power & Energy Systems*, 55:628–644, 2014.

[73] D. Karaboga. An idea based on honey bee swarm for numerical optimization. Technical report, Technical report-tr06, Erciyes university, engineering faculty, computer engineering department, 2005.

[74] J. Kennedy. Particle swarm optimization. *IEEE International Conference on Neural Network*, pages 1942–1948, 1995.

[75] A. Koh. Solving transportation bi-level programs with differential evolution. In *IEEE Congress on Evolutionary Computation, CEC 2007.*, pages 2243–2250. IEEE, 2007.

[76] Hsiang-Tsung Kung, Fabrizio Luccio, and Franco P Preparata. On finding the maxima of a set of vectors. *Journal of the ACM (JACM)*, 22(4):469–476, 1975.

[77] F. Legillon, A. Liefooghe, and E. G. Talbi. Cobra: A cooperative coevolutionary algorithm for bi-level optimization. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE, 2012.

[78] A. S. Lewis and M. L. Overton. Nonsmooth optimization via quasi-Newton methods. *Mathematical Programming*, 141(1-2):135–163, 2013.

[79] Hong Li and Li Zhang. An efficient solution strategy for bilevel multiobjective optimization problems using multiobjective evolutionary algorithm. *Soft Computing*, pages 1–21, 2021.

[80] Hong Li, Qingfu Zhang, Qin Chen, Li Zhang, and Yong-Chang Jiao. Multiobjective differential evolution algorithm based on decomposition for a type of multiobjective bilevel programming problems. *Knowledge-Based Systems*, 107:271–288, 2016.

[81] Hui Li and Qingfu Zhang. Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II. *IEEE transactions on evolutionary computation*, 13(2):284–302, 2008.

[82] X. Li, P. Tian, and X. Min. A hierarchical particle swarm optimization for solving bilevel programming problems. In *International Conference on Artificial Intelligence and Soft Computing*, pages 1169–1178. Springer, 2006.

[83] Bingbing Liu, Zhongping Wan, Jiawei Chen, and Guangmin Wang. Optimality conditions for pessimistic semivectorial bilevel programming problems. *Journal of Inequalities and Applications*, 2014(1):1–26, 2014.

[84] D. C. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

[85] M. López-Ibáñez, J. Dubois-Lacoste, L. Pérez-Cáceres, M. Birattari, and T. Stützle. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58, 2016.

[86] Yibing Lv and Zhongping Wan. A solution method for the optimistic linear semivectorial bilevel optimization problem. *Journal of Inequalities and Applications*, 2014(1):1–10, 2014.

[87] Yibing Lv and Zhongping Wan. Solving linear bilevel multiobjective programming problem via exact penalty function approach. *Journal of Inequalities and Applications*, 2015(1):1–12, 2015.

[88] J.E. Marsden and A. Tromba. *Vector calculus*. Macmillan, 2003.

[89] J. A. Mejía-de Dios and E. Mezura-Montes. A physics-inspired algorithm for bilevel optimization. In *Power, Electronics and Computing (ROPEC), 2018 IEEE International Autumn Meeting on*, pages 1–6. IEEE, 2018.

[90] J. A. Mejía-de Dios and E. Mezura-Montes. A new evolutionary optimization method based on center of mass. In *Decision Science in Action*, pages 65–74. Springer, 2019.

[91] J. A. Mejía-de Dios and E. Mezura-Montes. A surrogate-assisted metaheuristic for bilevel optimization. In *Proceedings of the ACM Genetic and Evolutionary Computation Conference (GECCO)*, page in press. ACM Press, 2020.

[92] J.A. Mejía-de Dios and E. Mezura-Montes. A Metaheuristic for Bilevel Optimization Using Tykhonov Regularization and the Quasi-Newton Method. In *2019 IEEE Congress on Evolutionary Computation (CEC)*, pages 3134–3141. IEEE, 2019.

[93] Jesús-Adolfo Mejía-de Dios and Efrén Mezura-Montes. *A New Evolutionary Optimization Method Based on Center of Mass*, pages 65–74. Springer Singapore, Singapore, 2019.

[94] Jesús-Adolfo Mejía-de Dios, Efrén Mezura-Montes, and Marcela Quiroz-Castellanos. Automated parameter tuning as a bilevel optimization problem solved by a surrogate-assisted population-based approach. *Applied Intelligence*, 51(8):5978–6000, 2021.

[95] 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.

[96] Efrén Mezura-Montes, Mariana-Edith Miranda-Varela, and Rubí del Carmen Gómez-Ramón. Differential evolution in constrained numerical optimization: An empirical study. *Information Sciences*, 180(22):4223–4262, 2010.

[97] M. Mitchell. An introduction to genetic algorithms. *Cambridge, MA: MIT Press*, 1996.

[98] Royall T. Moore. Proposal for the recognition of super ranks. *TAXON*, 23(4):650–652, 1974.

[99] Ajit Narayanan and Mark Moore. Quantum-inspired genetic algorithms. In *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*, pages 61–66. IEEE, 1996.

[100] Nadia Nedjah and Luiza de Macedo Mourelle. Evolutionary multi–objective optimisation: a survey. *International Journal of Bio-Inspired Computation*, 7(1):1–25, 2015.

[101] J. Nocedal and S. Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[102] Y. S. Ong, P. B. Nair, and A. J. Keane. Evolutionary optimization of computationally expensive problems via surrogate modeling. *AIAA journal*, 41(4):687–696, 2003.

[103] Calice Olivier Pieume, Patrice Marcotte, Laure Pauline Fotso, Patrick Siarry, et al. Solving bilevel linear multiobjective programming problems. *American Journal of Operations Research*, 1(4):214–219, 2011.

[104] Calice Olivier Pieume, Patrice Marcotte, Laure Pauline Fotso, Patrick Siarry, et al. Generating efficient solutions in bilevel multi-objective programming problems. *American Journal of Operations Research*, 3(02):289, 2013.

[105] M. Quiroz-Castellanos, L. Cruz-Reyes, J. Torres-Jimenez, C. Gómez, H. J. Fraire Huacuja, and A.C.F. Alvim. A grouping genetic algorithm with con-

trolled gene transmission for the bin packing problem. *Computers & Operations Research*, 55:52–64, 2015.

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

[107] Esmat Rashedi, Hossein Nezamabadi-pour, and Saeid Saryazdi. GSA: a Gravitational Search Algorithm. *Information sciences*, 179(13):2232–2248, 2009.

[108] S. Shan and G. G. Wang. Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions. *Structural and Multidisciplinary Optimization*, 41(2):219–241, 2010.

[109] C. Shi, J. Lu, and G. Zhang. An extended Kuhn–Tucker approach for linear bilevel programming. *Applied Mathematics and Computation*, 162(1):51–63, 2005.

[110] X. Shi and H. Xia. Interactive bilevel multi-objective decision making. *Journal of the Operational Research Society*, 48:943–949, 1997.

[111] A. Sinha, Z. Lu, K. Deb, and P. Malo. Bilevel optimization based on iterative approximation of multiple mappings. *arXiv preprint arXiv:1702.03394*, 2017.

[112] A. Sinha, P. Malo, and K. Deb. Unconstrained scalable test problems for single-objective bilevel optimization. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE, 2012.

[113] A. Sinha, P. Malo, and K. Deb. Efficient evolutionary algorithm for single-objective bilevel optimization. *arXiv preprint arXiv:1303.3901*, 2013.

[114] A. Sinha, P. Malo, and K. Deb. Test problem construction for single-objective bilevel optimization. *Evolutionary computation*, 22(3):439–477, 2014.

[115] A. Sinha, P. Malo, and K. Deb. Transportation policy formulation as a multi-objective bilevel optimization problem. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 1651–1658. IEEE, 2015.

[116] A. Sinha, P. Malo, and K. Deb. A review on bilevel optimization: from classical to evolutionary approaches and applications. *IEEE Transactions on Evolutionary Computation*, 22(2):276–295, 2018.

[117] A. Sinha, P. Malo, P. Xu, and K. Deb. A bilevel optimization approach to automated parameter tuning. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, pages 847–854. ACM, 2014.

[118] Ankur Sinha, Pekka Malo, and Kalyanmoy Deb. An improved bilevel evolutionary algorithm based on quadratic approximations. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, pages 1870–1877. IEEE, 2014.

[119] Ankur Sinha, Pekka Malo, and Kalyanmoy Deb. Approximated set-valued mapping approach for handling multiobjective bilevel problems. *Computers & Operations Research*, 77:194–209, 2017.

[120] Ankur Sinha, Pekka Malo, and Kalyanmoy Deb. Evolutionary bilevel optimization: An introduction and recent advances. In *Recent advances in evolutionary multi-objective optimization*, pages 71–103. Springer, 2017.

[121] Ankur Sinha, Tharo Soun, and Kalyanmoy Deb. Using Karush-Kuhn-Tucker proximity measure for solving bilevel optimization problems. *Swarm and evolutionary computation*, 44:496–510, 2019.

[122] S. K. Smit and A. E. Eiben. Comparing parameter tuning methods for evolutionary algorithms. In *2009 IEEE congress on evolutionary computation*, pages 399–406. IEEE, 2009.

[123] M. Spivak. *Calculus on manifolds: a modern approach to classical theorems of advanced calculus*. CRC press, 2018.

[124] R. Storn and K. Price. Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. *Berkeley: ICSI*, 1995.

[125] E. G. Talbi. A taxonomy of metaheuristics for bi-level optimization. In *Metaheuristics for bi-level optimization*, pages 1–39. Springer, 2013.

[126] I. C. Trelea. The particle swarm optimization algorithm: convergence analysis and parameter selection. *Information processing letters*, 85(6):317–325, 2003.

[127] F. Van den Bergh and A. P. Engelbrecht. A study of particle swarm optimization particle trajectories. *Information sciences*, 176(8):937–971, 2006.

[128] N. Veček, M. Mernik, B. Filipič, and M. Črepinšek. Parameter tuning with chess rating system (crs-tuning) for meta-heuristic algorithms. *Information Sciences*, 372:446–469, 2016.

[129] Mark Velasquez and Patrick T Hester. An analysis of multi-criteria decision making methods. *International journal of operations research*, 10(2):56–66, 2013.

[130] L. Vicente, G. Savard, and J. Júdice. Descent approaches for quadratic bilevel programming. *Journal of Optimization Theory and Applications*, 81(2):379–399, 1994.

[131] J. Von-Neumann and O. Morgenstern. Theory of games and economic behavior. *Bull. Amer. Math. Soc*, 51(7):498–504, 1945.

[132] H. Von-Stackelberg. *Market Structure and Equilibrium*. Springer Science & Business Media, 2010.

[133] J. Y. T. Wang, M. Ehrgott, K. N. Dirks, and A. Gupta. A bilevel multi-objective road pricing model for economic, environmental and health sustainability. *Transportation Research Procedia*, 3:393–402, 2014.

[134] Xiaoli Wang and Yuping Wang. An energy and data locality aware bi-level multiobjective task scheduling model based on mapreduce for cloud computing. In *2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, volume 1, pages 648–655, 2012.

[135] Y. Wang, Y.-C. Jiao, and H. Li. An evolutionary algorithm for solving nonlinear bilevel programming based on a new constraint-handling scheme. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 35(2):221–232, 2005.

[136] Lyndon While, Philip Hingston, Luigi Barone, and Simon Huband. A faster algorithm for calculating hypervolume. *IEEE transactions on evolutionary computation*, 10(1):29–38, 2006.

[137] Jianlin Jiang Yibing Lv, Tiesong Hu. Penalty method-based equilibrium point approach for solving the linear bilevel multiobjective programming problem. *Discrete & Continuous Dynamical Systems - S*, 13(6):1743–1755, 2020.

[138] Zhongping Wan Yibing Lv. Linear bilevel multiobjective optimization problem: Penalty approach. *Journal of Industrial & Management Optimization*, 15(3):1213–1223, 2019.

[139] E. A. Youness, O. E. Emam, and M. S. Hafez. Fuzzy bi-level multi-objective fractional integer programming. *Applied Mathematics & Information Sciences*, 8(6):2857–2863, November 2014.

[140] B. Yuan and M. Gallagher. Statistical racing techniques for improved empirical evaluation of evolutionary algorithms. In *International Conference on Parallel Problem Solving from Nature*, pages 172–181. Springer, 2004.

[141] B. Yuan and M. Gallagher. Combining meta-eas and racing for difficult ea parameter tuning tasks. In *Parameter Setting in Evolutionary Algorithms*, pages 121–142. Springer, 2007.

[142] Guangquan Zhang, Jie Lu, and Tharam Dillon. Decentralized multi-objective bilevel decision making with fuzzy demands. *Knowledge-Based*

Systems, 20(5):495–507, 2007. Intelligent Knowledge Engineering Systems.

[143] Tao Zhang, Tiesong Hu, Jia-wei Chen, Zhongping Wan, and Xuning Guo. Solving bilevel multiobjective programming problem by elite quantum behaved particle swarm optimization. In *Abstract and Applied Analysis*, volume 2012. Hindawi, 2012.

[144] Y. Zheng, Z. Wan, and G. Wang. A fuzzy interactive method for a class of bilevel multiobjective programming problem. *Expert Systems with Applications*, 38(8):10384–10388, 2011.

[145] A. Zhou, B.Y. Qu, H. Li, S. Z. Zhao, P. N. Suganthan, and Q. Zhang. Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation*, 1(1):32–49, 2011.

[146] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. Spea2: Improving the strength pareto evolutionary algorithm. *TIK-report*, 103, 2001.

[147] Eckart Zitzler and Lothar Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE transactions on Evolutionary Computation*, 3(4):257–271, 1999.

Appendix A

Optimization

This appendix is dedicated to describing what optimization problems are, including the solution methodology from classical to approximate approaches such as evolutionary algorithms and other metaheuristic methods. Here, single- and multi-objective problems are briefly introduced to give important concepts to further introduce bilevel optimization problems comprehensively.

A.1 Single-Objective Optimization

In this section, the single-level optimization problems are described by giving important definitions and theoretical results that involve theorems on optimality conditions that guarantee the existence of solutions for some optimization tasks. Optimizing a single-objective problem means that it is required to find arguments for the real-valued function such that the minimum is reached on those arguments. Thus, an optimization problem is defined, without loss of generality, as finding the set:

$$X^* = \underset{\boldsymbol{x} \in X}{\operatorname{argmin}} f(\boldsymbol{x}) = \{\boldsymbol{x}^* \in X \ : \ f(\boldsymbol{x}^*) \leq f(\boldsymbol{x}),\ \forall \boldsymbol{x} \in X\}, \tag{A.1}$$

where f is a bounded below function (called objective function), i.e., $f(x^*) > -\infty$. X is a D-dimensional parameter space (search space), usually $X \subset \mathbb{R}^D$ is the domain for $\boldsymbol{x}$ representing constraints on allowable values for $\boldsymbol{x}$. Equation A.1 may be read as: X^* is the set of values (arguments) $\boldsymbol{x} = \boldsymbol{x}^*$ that minimizes $f(\boldsymbol{x})$ subject to X^* (see Figure A.1).

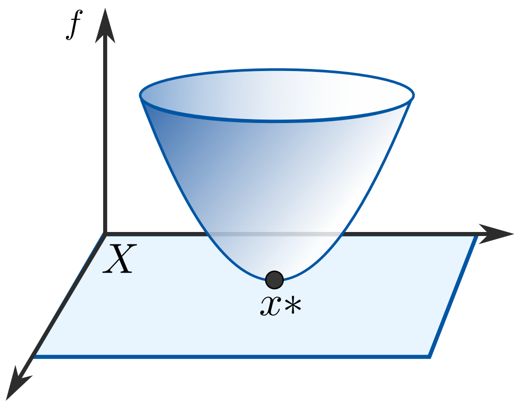


Figure A.1: A single-objective optimization problem is represented. Note that X represents the search space, and $\boldsymbol{x}^*$ is minimizing f on X.

From the above description, different classes of optimal solutions emerge depending on the properties of the objective function f. For instance, if f is oscillating on a range, then local or global optimal solutions can appear. In practice, it is common to find that f contains multiple optimal solutions. The main idea is to compute global optima.

Given a function $f : X \subset \mathbb{R}^n \rightarrow \mathbb{R}$ with $X \neq \emptyset$, for $\boldsymbol{x}^* \in X$ the value $f^* := f(\boldsymbol{x}^*) > -\infty$ is called a **global minimum** if and only if

$$\forall \boldsymbol{x} \in X, \quad f(\boldsymbol{x}^*) \leq f(\boldsymbol{x}).$$

Then, $\boldsymbol{x}^*$ is a global minimum point, f is the objective function, and the set X is the feasible region.

An optimization problem is solved only when a global minimum is found. However, global minimums are, in general, difficult to find because objective functions can be hard to analyze in real-world problems. Therefore, in practice, a local minimum [106, 12] that represents a better solution than an *a priori* known solution will be acceptable. Optimization problems are usually ill-posed problems because existence of optimal solution is not guaranteed in general, and when solutions exist, they can not be unique.

More complicated optimization problems can contain some constraints inherent to the task being modeled, adding a difficulty layer to the resolution strategy. Those constraints are determined by equality and inequality equations besides that objective function. That is, a constrained optimization problem restricts the global search space into regions where the function arguments can be a candidate to be an optimum (see Definition 1.1).

Definition 1.1

A general **Constrained Optimization Problem** (COP) can be defined as follows:
Minimize:

$$f(\boldsymbol{x}),\ \boldsymbol{x} \in S \subseteq \mathbb{R}^D \tag{A.2}$$

subject to:

$$g_i(\boldsymbol{x}) \leq 0, \qquad i = 1, \ldots, p \tag{A.3}$$

$$h_j(\boldsymbol{x}) = 0, \qquad j = p+1, \ldots, m \tag{A.4}$$

where $S = \prod_{k=1}^{D} [x_{k,\min},\ x_{k,\max}]$ i.e. $x_k \in [x_{k,\min},\ x_{k,\max}]$ for $k = 1, 2, \ldots, D$. The problem is subject to p inequality constraints and $m - p$ equality constraints. If $\boldsymbol{x}$ satisfies $g_i(\boldsymbol{x}) \leq 0$, for $i = 1, \ldots, p$ and $|h_j(\boldsymbol{x})| \leq \varepsilon$, for $j = p+1, \ldots, m$ with $\varepsilon > 0$ a small value; then $\boldsymbol{x}$ is regarded feasible.

As mentioned before, COPs are, in general, complicated to solve since the constraints can add multiple sources of difficulty in the search space. For example, when the constraints introduce non-convex or disconnected feasible regions, finding optimal feasible solutions become a challenging task.

To illustrate the main properties of an optimization problem, the following example is given.

Example 1.1

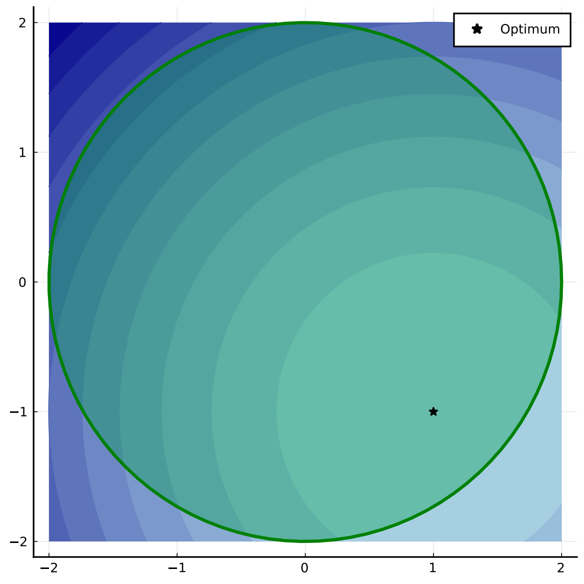
The Sphere function constrained by a disk:

$$f(x_1, x_2) = (x_1 - 1)^2 + (x_2 + 1)^2$$

subject to:

$$x_1^2 + x_2^2 \leq 2,$$

where $-2 \leq x_1, x_2 \leq 2$.



As you can note, this nonlinear problem contains two decision variables and an inequality constraint.

The following section gives important results on optimal solutions' existence in optimization problems and optimality conditions to determine which optimization problems have at least a local optima.

A.1.1 Optimality Conditions

Optimality Conditions have been developed from theoretical analyzes to prove that optimal solutions exist for some classes of optimization problems. Usually, classical algorithms to solve optimization problems are based on optimality conditions.

Gradient-based optimality conditions assume that $f(\boldsymbol{x})$ is continuous and differentiable, i.e., partial derivatives exist.

$$\nabla f(\boldsymbol{x}) = \left(\frac{\partial f(\boldsymbol{x})}{\partial x_1}, \frac{\partial f(\boldsymbol{x})}{\partial x_2}, \ldots, \frac{\partial f(\boldsymbol{x})}{\partial x_n} \right)^T$$

is the vector of partial derivatives known as gradient vector (gradient of $f(\boldsymbol{x})$). This gradient vector is very useful to determine the direction where f takes the largest values, suggesting then a search direction. Moreover, the gradient vector becomes the zero vector on optimal solutions. This fact introduces a necessary optimality condition (but not sufficient). The following theorems use the gradient concept to give necessary and sufficient optimality conditions.

Theorem 1.1

If the objective function $f(\boldsymbol{x})$ has maximum or minimum (also known as extreme point) at $\boldsymbol{x} = \hat{\boldsymbol{x}}$ and the first partial derivates of $f(\boldsymbol{x})$ exists, then $\nabla f(\hat{\boldsymbol{x}}) = (0, 0, \ldots, 0)^T = \mathbf{0}$.

The proof for this theorem is detailed in [106].

As you can see in Theorem 1.1, optimal solutions satisfy some interest properties that help to characterize solution nature, letting us know if the point $\boldsymbol{x}$ could be a minimum or maximum point. The above theorem states that any optimal point cancels the gradient vector. However, other solutions could also cancel it. It is necessary to determine whether a solution is optimal, then the necessary condition is required.

Theorem 1.2

If $\boldsymbol{x}$ is a stationary point, then $\hat{\boldsymbol{x}}$ is a local optimum and the Hessian matrix H (second derivate for $f(\boldsymbol{x})$) evaluated at $\hat{\boldsymbol{x}}$. Then: 1) $\hat{\boldsymbol{x}}$ is a local minimum point if $H(\hat{\boldsymbol{x}})$ is positive-definite, or 2) $\hat{\boldsymbol{x}}$ is a local maximum point if $H(\hat{\boldsymbol{x}})$ is negative-definite.

The proof for this theorem is detailed in [106].

The necessary condition has been developed for twice differentiable functions f at a solution $\hat{\boldsymbol{x}}$ that satisfies Theorem 1.2, i.e., $\nabla f(\hat{\boldsymbol{x}}) = \mathbf{0}$ and the Hessian matrix is definite positive or negative, then $\hat{\boldsymbol{x}}$ is a strict local optima [106].

As you can see from the above conditions, constrained problems have not been considered due to more considerations have to be done according to the nature of equality and inequality constraints. Therefore, different conditions have been studied for constrained optimization problems, such as the Karush-Kuhn-Tucker conditions addressed in the following part.

A.1.2 Karush-Kuhn-Tucker Conditions

The Karush-Kuhn-Tucker (KKT) conditions are used to handle equality and inequality constraints in nonlinear optimization problems as a generalization of the Lagrange Multiplier method (which only considers equality constraints) [17, 26, 109]. Note that the KKT conditions are also called Kuhn-Tucker conditions or first-order necessary conditions.

The main idea is to transform the constrained problem from the Definition 1.1

into the Lagrangian function:

$$L(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \boldsymbol{\mu}^\top \boldsymbol{g}(\boldsymbol{x}) + \boldsymbol{\lambda}^\top \boldsymbol{h}(\boldsymbol{x})$$

where $\boldsymbol{g}(\boldsymbol{x}) = (g_1(\boldsymbol{x}), \ldots, g_p(\boldsymbol{x}))^\top$, $\boldsymbol{h}(\boldsymbol{x}) = (h_1(\boldsymbol{x}), \ldots, h_m(\boldsymbol{x}))^\top$ denote the equality and inequality constraints, respectively. Thus, the following result determines whether a point is an optimal solution for the problem in Definition 1.1.

Theorem 1.3

Assume that the objective f and constraint functions g_i, h_j are continuously differentiable (the derivative exists and is itself continuous). If $(\boldsymbol{x}^*, \boldsymbol{\mu}^*)$ satisfy that:

$$L(\boldsymbol{x}^*, \boldsymbol{\mu}^*) = \nabla f(\boldsymbol{x}^*) + \sum_{i=1}^{m} \mu_i \nabla g_i(\boldsymbol{x}^*) + \sum_{j=1}^{J} \lambda_j \nabla h_j(\boldsymbol{x}^*) = \boldsymbol{0}$$

where $\lambda_j \geq 0$ and $\lambda_j g_j(x, y) \leq 0$. Then $\boldsymbol{x}^*$ is an optimal solution.

The proof for this theorem is detailed in [106].

Theorem 1.3 is one of the most important results in the optimization area. The KKT conditions are useful to handle equality and inequality constraints and characterize optimal feasible solutions by using a Lagrangian function. Different algorithms have been proposed to solve constrained optimization problems by implementing the KKT conditions.

A.1.3 Solutions Strategies

The specialized literature reports different strategies for solving optimization tasks, such as exact methods derived from mathematical procedures where the characteristic of problems is known [101, 106]. However, when the information on the optimization problem is not fully available, approximate strategies such as *metaheuristic* algorithms become relevant as they can approximate solutions for highly complicated problems.

Classical Approaches

Classical approaches to solving optimization problems are always the first option when the information on the objective function and constraints is given *a priori*, and the optimization problem behaves well. As mentioned before, exact algorithms can solve well-behaved problems such as linear, quadratic, convex, and non-convex programming problems. For instance, Newton-like or quasi-Newton algorithms are based on the first-order derivate information of the objective function [106].

Metaheuristic Approaches

The *metaheuristic* concept was introduced in [54] to describe an algorithm which uses upper-level heuristics[1] to solve complex optimization problems [125]. The most popular metaheuristics are based on Darwin's theory of natural evolution, known as Evolutionary Algorithms (EAs)

- *Evolutionary Algorithms.* EAs have been successfully used to solve real-world optimization problems [21, 48, 31]. EAs are methods with stochastic operators such as selection, recombination, and mutation, inspired by the natural selection of species (evolution). Some of the most popular EAs are Genetic Algorithms [97], Evolutionary Programming, Evolution Strategies, and Genetic Programming [12, 50]. Nowadays, one of the most popular EA for real-parameter optimization, particularly in constrained search spaces, is DE [96] which was introduced by Storn and Price [124] for global optimization over continuous search spaces. Evolution-based algorithms are commonly used when traditional/exact techniques can not be applied (or the properties of problems are not available) because EAs do not require strong assumptions.

[1]Discover new strategies to solve problems.

- *Physics-Inspired Algorithms.* Physics-inspired algorithms are based on models from physical phenomena such that gravitational law [107], the center-of-mass concept [90], quantum physics [99], among others [21]. Those methods can work on a set of solutions but are not necessarily based on stochastic principles (as in EAs) to successfully solve complex optimization problems. The reader is referred to [19, 21] for finer details on physics-inspired methods that solve optimization problems.

- *Hybrid Algorithms.* When mathematical properties of the optimization problem are known, classical techniques can be applied to obtain exact local optima. However, when accurate enough global solutions are required, hybrid methods have been proposed. They combine metaheuristic methods (such as global search); and classical techniques (local search) to improve the approximation accuracy [47, 53]. Also, combinations of metaheuristics from different inspirations have been considered to improve results on complicated problems when traditional frameworks are unable to provide desired solutions [72]. Thus, combining different algorithms to propose a new algorithm is referred to as hybrid algorithms.

A.2 Multi-Objective Optimization

Optimization problems simultaneously dealing with more than one objective function are known as multi-objective problems if such objectives are in conflict. Moreover, different real-world problems can be modeled as multi-objective optimization tasks, providing a most representative mathematical model to the problem of interest.

Important Concepts

- **Conflicting Objectives:** When an objective takes smaller values, another one takes larger values.

- **Multi-objective problem:** Optimization problems with multiple conflicting objectives.

Without lost of generality, a multi-objective optimization problem can be formulated as follows [28, 14, 145]:

$$\min_{\boldsymbol{x} \in X} F(\boldsymbol{x}) = \begin{pmatrix} F_1(\boldsymbol{x}) \\ F_2(\boldsymbol{x}) \\ \vdots \\ F_M(\boldsymbol{x}) \end{pmatrix} \tag{A.5}$$

$$g_j(\boldsymbol{x}) \leq 0, j = 1, 2, \ldots, J \tag{A.6}$$

$$h_k(\boldsymbol{x}) = 0, k = 1, 2, \ldots, K \tag{A.7}$$

$$\boldsymbol{x} \in X. \tag{A.8}$$

where M is the number of conflicting objectives, g_j and h_k respectively denote inequality and equality constraints. X is the search space or the function domain.

Here, the optimality definitions are quite different from those in single-objective optimization since $F : X \subset \mathbb{R}^D \to R^M$ is not a real-valued objective function. Therefore, it is necessary to introduce concepts of dominance to discriminate the quality of a solution.

Definition 1.2

A solution in $\boldsymbol{x}^* \in \Omega$ is **Pareto Optimal** if $\forall \boldsymbol{x} \in \Omega - \{\boldsymbol{x}^*\}$, $F_i(\boldsymbol{x}^*) \leq F_i(\boldsymbol{x})$ is obtained for $1 \leq i \leq M$ and there exists at least one $1 \leq j \leq M$ such that $F_j(\boldsymbol{x}^*) < F_j(\boldsymbol{x})$.

Once the optimality is described in this context, the Pareto Optimal Set $P^* \subset \Omega$ is defined, such as any element in P^* is a Pareto optimal solution. Another important concept is the Pareto-Optimal front given by

$$PF^* = \{F(\boldsymbol{x}) : \boldsymbol{x} \in P^*\}.$$

Finding solutions distributed along the Pareto-optimal front is a complicated task. Here, optimality is hard to meet, and a well-distribution of solutions is required when *a posteriori* decision-making is performed.

Other important concepts in multi-objective optimization are the Ideal and Nadir points analytically defined as:

- Ideal point: $\boldsymbol{z} = (z_1, z_2, \ldots, z_M)$, where $z_m = \min_{\boldsymbol{x}\in\Omega} F_m(\boldsymbol{z})$.
- Nadir point: $\boldsymbol{w} = (w_1, w_2, \ldots, w_M)$, where $w_m = \max_{\boldsymbol{x}\in\Omega} F_m(\boldsymbol{w})$.

Definition 1.3

A solution $\boldsymbol{x}_1 \in \Omega$ dominates $\boldsymbol{x}_2 \in \Omega$ if and only if $F_i(\boldsymbol{x}_1) \leq F_i(\boldsymbol{x}_2)$ for $1 \leq i \leq M$ and there exists at least one $1 \leq j \leq M$ such that $F_j(\boldsymbol{x}_1) < F_j(\boldsymbol{x}_2)$.

The above concepts are important to compare two different solutions in terms of optimality. A representation is given in Figure A.2 which is used to visualize how optimal solutions are distributed.

Multi-objective problems can be solved by using classical approaches from mathematical programming, and fuzzy logic, among others. Evolutionary multi-objective algorithms are proposed to approximate solutions along the Pareto-optimal front. Traditional methods aggregate the different objective functions into a single one (weighted sum of objective values) [14]. However, those methods make strong assumptions about the objective functions limiting their applicability. Here, exact algorithms are not addressed to extend the discussion on multi-objective evolutionary algorithms. The reader is referred to [129] for further details on classical approaches to solving multi-objective problems.

A.2.1 Multi-Objective Evolutionary Algorithms

High interest from the Evolutionary Computing community has been observed because evolutionary algorithms are highly competitive to solve multi-objective optimization problems. MOEAs are based on different ideas such as Pareto dominance, decomposition, and indicators, among others, [14, 100, 27]. The most representative algorithms are listed and described as follows:

- Non-dominated sorting Genetic Algorithm II (NSGA-II) is one of the most popular MOEA thanks to its simplicity and low computational complexity. NSGA-II Uses a non-dominates sorting and crowding distance to keep a promising distribution of solutions [34].

- Strength Pareto Evolutionary Algorithm (SPEA/SPEA2) uses the main population for the optimization process and an archive population to save non-dominated solutions found via the main population [147]. Here, the strength Pareto indicates the percentage of solutions in the current population that a solution is Pareto-dominating.

- Multi-objective Evolutionary Algorithm Based on Decomposition (MOEA/D)

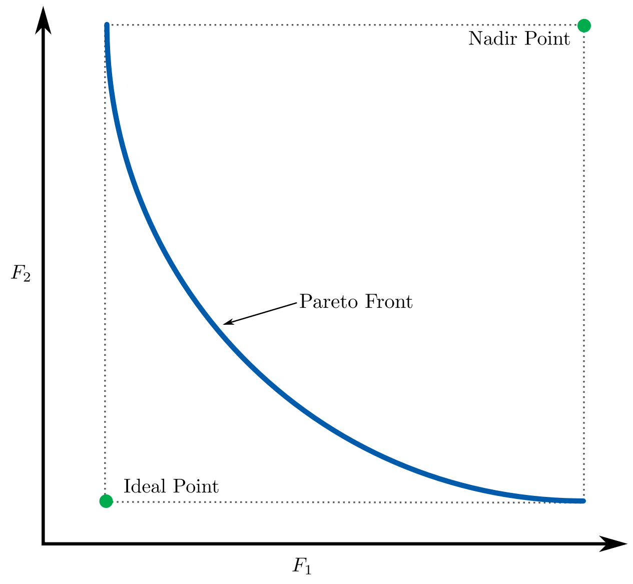


Figure A.2: Illustrating the distribution of a Pareto-optimal front, the location of the nadir, and ideal points. Also, it can be observed that the Pareto-optimal front determines a convex region. Minimization is assumed in this figure.

is one of the most popular decomposition-based MOEA. MOEA/D simultaneously solves as many sub-problems (optimizes single objective functions known as scalarizing functions) as elements in the population [81]. It is worth mentioning that each sub-problem is defined by a scalarizing function parameterized by a weight vector, and the distribution of the weight vectors helps the optimizer to control the diversity of solutions.

- S-Metric Selection Evolutionary Multi-Objective Algorithm (SMS-EMOA) is a metric-driven optimizer that uses the hypervolume indicator to guide the search for the Pareto front approximation [15]. A reduction operation is adopted to maintain only NP individuals in the population.

Assessing the algorithms' performance to highlight their pros and cons of is important. The next section is devoted to describing some of the most used indicators.

A.2.2 Performance Indicators

Different performance indicators have been proposed to evaluate MOEAs' performance [14, 45]. The most common indicators are used to quantify how close the approximate Pareto front and true Pareto Front are. The most commonly used performance indicators are then described as follows [45, 62, 63]:

- The Generational Distance (GD) estimates how close is the Pareto front found by the MOEA from the true Pareto front in a given problem [62].

$$GD(PF, PF^*) = \sqrt{\sum_{i=1}^{N} d_i^2} \Big/ |PF^*| \tag{A.9}$$

  where N is the number of non-dominated solutions found by the MOEA and d_i is the (euclidean) distance between the i-th element in PF and the closest to it in PF^*.

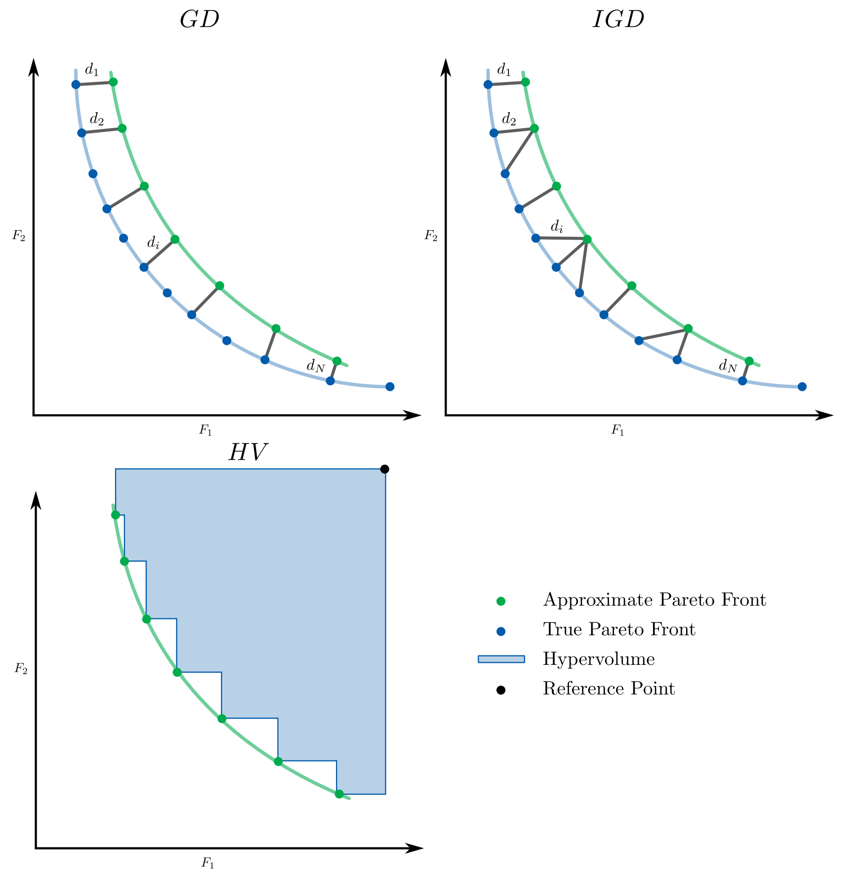


Figure A.3: Main aspects when the generational distance (GD), inverted generational distance (IGD), and hypervolume indicator (HV) are computed.

- Inverted Generational Distance (IGD) is a variant of GD, but in this case, each element in the true Pareto front is compared concerning the front found by the MOEA [62].

- Hypervolume (HV) is used to approximate the area/volume, which is dominated by the Pareto front obtained concerning a given reference point [136].

GD and IGD are not related to the Pareto-optimality definitions (Pareto-compliant), but they are widely used to numerically determine the distance between the Pareto-optimal front and the obtained by an evolutionary algorithm. The hypervolume indicator is Pareto-compliant, and it can be used to determine how close a set of non-dominated solutions is to the Pareto-optimal front but also to quantify the distribution along the front. Figure A.3 shows how the GD, IGD, and hypervolume are calculated.

Appendix B

Proofs of Theorems

This part contains the corresponding proofs for Theorem 2.3 and Corollary 2.1.

Proof B.1

Let $(F,\ f,\ X,\ Y,\ \mathbb{R})$ be a BO problem that satisfies the hypothesis in Theorem 2.3. Since f is non-injective, then, there exists $(\boldsymbol{w},\boldsymbol{z}) \in X \times Y$ such that $f(\boldsymbol{x},\boldsymbol{y}^*) = f(\boldsymbol{w},\boldsymbol{z})$, then $F(\boldsymbol{w},\boldsymbol{z}) = Q(f(\boldsymbol{w},\boldsymbol{z})) = Q(f(\boldsymbol{x},\boldsymbol{y}^*)) = F(\boldsymbol{x},\boldsymbol{y}^*)$ with $Q : \text{Img}(f) \subseteq \mathbb{R} \to \mathbb{R}$. $\square$

The following proof is associated to Corollary 2.1.

Proof B.2

If f is non-injective at $(\boldsymbol{x}^*,\ \boldsymbol{y}^*)$ and $(\boldsymbol{x},\ \boldsymbol{y})$ with $\boldsymbol{y} \in Y - \Psi(\boldsymbol{x})$. The result is fulfilled by applying Theorem 2.3. $\square$

Appendix C

Test Problems

C.1 PMM Test Suite

PMM1

Here, q_1, r_1 and r_2 are convex functions. The LL problem is differentiable on Y. Those functions are defined as follows:

$$q_1(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{k} \left(y_i - \frac{x_i^3}{100} \right)^2, \qquad r_1(\boldsymbol{x}) = -\sum_{i=1}^{k} x_i^2,$$

$$q_2(\boldsymbol{y}) = 10 + y_{k+1} + 10^6 \sum_{i=k+2}^{D_{ll}} y_i^2, \qquad r_2(\boldsymbol{x}) = -\sum_{i=k+1}^{D_{ul}} x_i^2.$$

A feasible solution is obtained when $y_i = (x_i^3)/100$ for $i = 1, 2, \ldots, k$; $y_{k+1} = 10$ and $y_{k+2} = \cdots = y_{D_{ll}} = 0$. The upper level is a convex task on feasible solutions.

PMM2

In this test problem, q_1 and r_2 are shifted Ben Cigar functions, q_2 is a shifted Zakharov function, and r_1 is the sphere function. Hence, the upper-level optimization problem is a convex task on feasible solutions. Those functions are defined as follows:

$$
\begin{aligned}
q_1(\boldsymbol{x}, \boldsymbol{y}) &= (y_1 - x_1 \sin(x_1))^2 + 10^6 \sum_{i=2}^{k} (y_i - x_i \sin(x_i))^2, \\
r_1(\boldsymbol{x}) &= -\sum_{i=1}^{k} x_i^2,
\end{aligned}
$$

$$
\begin{aligned}
q_2(\boldsymbol{x}) &= \sum_{i=k+1}^{D_{ll}} (y_i - \sqrt{i})^2 + \left(\sum_{i=k+1}^{D_{ll}} 0.5(y_i - \sqrt{i}) \right)^2, \\
&\quad + \left(\sum_{i=k+1}^{D_{ll}} 0.5(y_i - \sqrt{i}) \right)^4, \\
r_2(\boldsymbol{x}) &= -x_{k+1}^2 - 10^6 \sum_{i=k+2}^{D_{ul}} x_i^2.
\end{aligned}
$$

A feasible solution is obtained when $y_i = x_i \sin(x_i)$ for $i = 1, 2, \ldots, k$, and $y_i = \sqrt{i}$, for $i = k+1,\ k+2, \ldots, D_{ll}$.

PMM3

In this test problem q_1 is a shifted sphere function, q_2 is the Rosenbrock's function, r_1 and r_2 are sphere functions. Those functions are defined as follows:

$$q_1(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{k} \left(y_i - \frac{x_i^3}{100} \right)^2, \qquad r_1(\boldsymbol{x}) = -\sum_{i=1}^{k} x_i^2,$$

$$q_2(\boldsymbol{y}) = \sum_{i=k+1}^{D_{ll}-1} [100(y_i^2 - y_{i+1})^2 + (y_i - 1)^2],$$
$$r_2(\boldsymbol{x}) = -\sum_{i=k+1}^{D_{ul}} x_i^2.$$

A feasible solution is obtained when $y_i = (x_i^3)/100$ for $i = 1, 2, \ldots, k$, and $y_i = 1$ for $i = k+1,\ k+2, \ldots, D_{ll}$. As we can see, the lower level has a multimodal objective function.

PMM4

For this test problem q_1 is a shifted sphere function, q_2 is the Rastrigin function, r_1 and r_2 are sphere functions. Moreover, this problem incorporates a relationship such that the approximation to the mapping Ψ might fail in some cases due to the Runge's phenomenon [46, 22]. Those functions are defined as follows:

$$q_1(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{k} \left(y_i - \frac{10}{1 + 2.5x_i^2} \right)^2, \qquad r_1(\boldsymbol{x}) = -\sum_{i=1}^{k} x_i^2,$$

$$q_2(\boldsymbol{y}) = 10(D_{ll} - k) + \sum_{i=k+1}^{D_{ll}} y_i^2 - 10\cos(2\pi y_i),$$
$$r_2(\boldsymbol{x}) = -\sum_{i=k+1}^{D_{ul}} x_i^2.$$

A feasible solution is obtained when $y_i = 10/(1 + 2.5x_i^2)$ for $i = 1, 2, \ldots, k$, and $y_i = 0$ for $i = k+1,\ k+2, \ldots, D_{ll}$.

PMM5

In this test problem q_1 is a shifted Rastrigin function, q_2 is the sum of different power functions, r_1 and r_2 are sphere functions. The lower-level optimization task is multimodal and non-differentiable. Those functions are defined as follows:

$$\begin{aligned} q_1(\boldsymbol{x}, \boldsymbol{y}) &= 10k + \sum_{i=1}^{k} (y_i - x_i)^2 - 10\cos(2\pi |y_i - x_i|), \\ r_1(\boldsymbol{x}) &= -\sum_{i=1}^{k} x_i^2, \end{aligned}$$

$$q_2(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=k+1}^{D_{ll}} |y_i|^{i-k+1}, \qquad r_2(\boldsymbol{x}) = -\sum_{i=k+1}^{D_{ul}} x_i^2.$$

A feasible solution is obtained when $y_i = x_i$ for $i = 1, 2, \ldots, k$, and $y_i = 0$ for $i = k+1,\ k+2, \ldots, D_{ll}$.

PMM6

This final test problem is the hardest one to solve of this test suite, since q_1 is a shifted Griewank's function, q_2 and r_2 are modified Rastrigin functions which are multimodal and r_1 is the sphere function. Those functions are defined as follows:

$$\begin{aligned}
q_1(\boldsymbol{x},\boldsymbol{y}) &= 1 + \frac{1}{4}\sum_{i=1}^{k}(y_i - x_i)^2 - \prod_{i=1}^{k}\cos\left(\frac{10(y_i - x_i)}{\sqrt{i}}\right),\\
r_1(\boldsymbol{x}) &= -\sum_{i=1}^{k} x_i^2,\\
q_2(\boldsymbol{x}) &= 10(D_{ll} - k) + \sum_{i=k+1}^{D_{ll}} y_i^2 - 10\cos(2\pi y_i),\\
r_2(\boldsymbol{x}) &= -\left[10(D_{ul} - k) + \sum_{i=k+1}^{D_{ul}} x_i^2 - 10\cos(2\pi x_i).\right]
\end{aligned}$$

A feasible solution is obtained when $y_i = x_i$ for $i = 1, 2, \ldots, k$, and $y_i = 0$ for $i = k+1,\ k+2, \ldots, D_{ll}$.