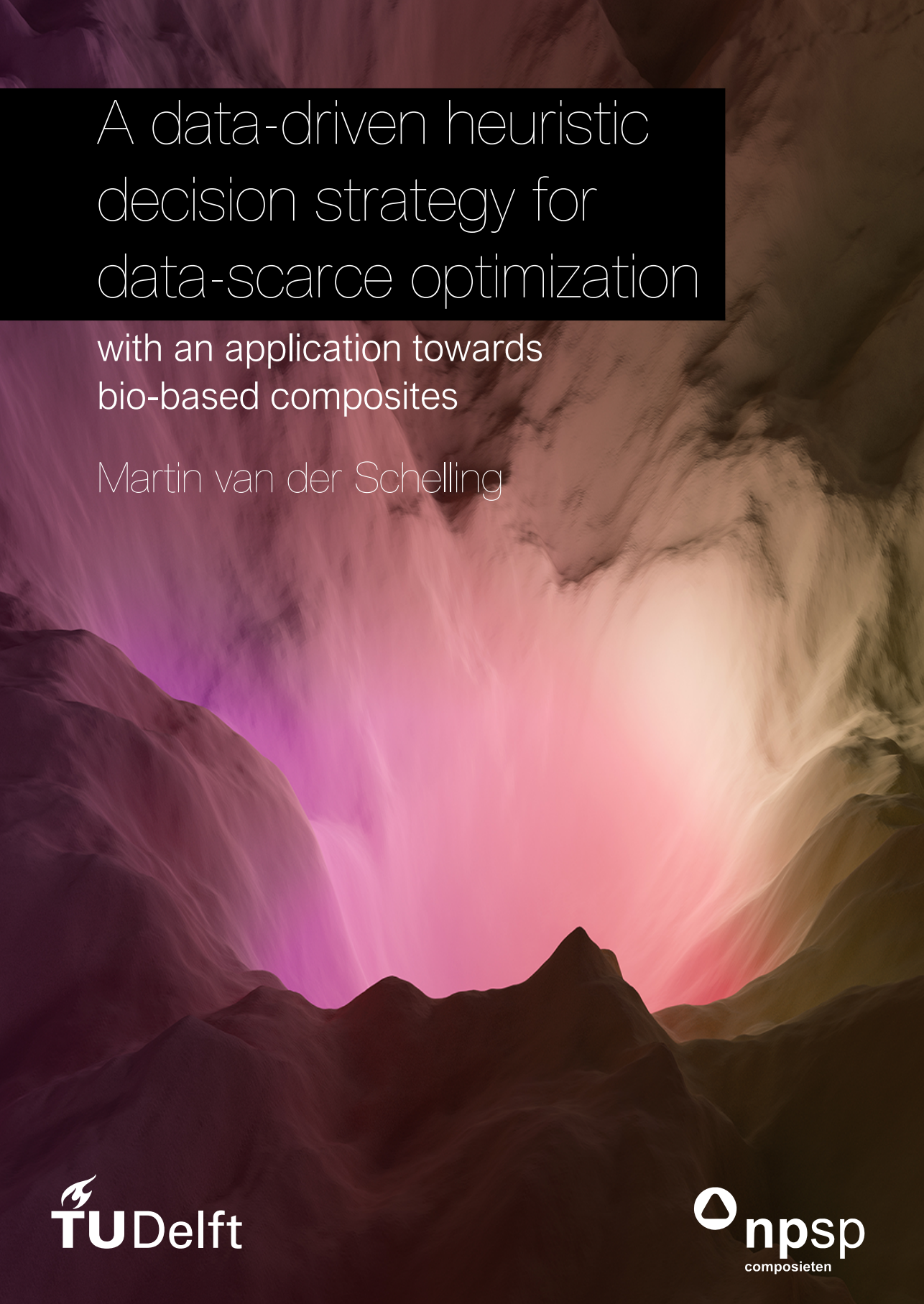
A data-driven heuristic decision strategy for data-scarce optimization

with an application towards bio-based composites

Martin van der Schelling



TU Delft

npsp composieten

A data-driven heuristic decision strategy for data-scarce optimization

With an application towards bio-based composites

by

Martin van der Schelling

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Tuesday March 9, 2021 at 3:00 PM.

Student number:	4304470	
Project duration:	March, 2020 – March, 2021	
Thesis advisor:	Dr. ir. M.A. Bessa,	TU Delft, 3ME faculty
Thesis committee:	Dr. M.H.F. Sluiter,	TU Delft, 3ME faculty
	Prof. dr. ir. C. Vuik,	TU Delft, EEMCS faculty
	Dr. ir. Z. Zarafshani,	NPSP B.V.
	Ir. W. Böttger,	NPSP B.V.

This thesis is confidential and cannot be made public until March 9, 2022.

An electronic version of this thesis is available at
http://repository.tudelft.nl/.

Cover picture: J. Ideami: "Loss Landscape created with data from the SGD-Adam training process of a convolutional network." Retrieved from: https://losslandscape.com/

TU Delft Delft University of Technology

Summary

Algorithmic optimization is a viable tool for solving complex materials engineering issues. In this study, a data-scarce Bayesian optimization model was developed to research the composition of bio-based composites. The proof-of-concept program adjusts the natural materials' weight ratios to optimize towards user-defined mechanical properties. Preliminary results show that the bio-composites proposed by the program had improved properties compared to existing bulk-moulding compounds. However, the algorithm choice is often arbitrary or based on anecdotal evidence. In parallel, this thesis proposed a data-driven framework for general data-scarce optimization problems to adapt the meta-heuristic during optimization. Guided by the 'No Free Lunch' theorem, we verified that the effectiveness over a selection of algorithms is dependent on problem-specific features and convergence. This effectiveness was captured in a unique identifier metric by optimizing a generated training set of optimization problems. The average solution quality was improved by combining several meta-heuristics in series, based on these problem-specifics. During the optimization of problems in the testing set, the same unique identifier was constructed at predefined stages in the optimization process. Subsequently, the problem was classified, and the meta-heuristic was adapted to the best-performing algorithm based on similar training samples. Experiments with various classifiers and a different number of predefined assessment stages were performed. Results show that the data-driven heuristic decision strategy outperformed the individual optimizers on the testing set. Despite the use of binarization techniques, the classification accuracy was heavily influenced by the imbalanced training set. In terms of computational resources, the various adaptions of the data-driven heuristic strategy are 2.5 times faster in runtime compared to the best-performing meta-heuristic Bayesian Optimization. Lastly, the framework was benchmarked against the 'learning to optimize' study and shows excellent performance on the logistic regression problem compared to the autonomous optimizer. In conclusion, it has been shown that even with the limited information of black-box optimization problems, data-driven optimization effectively improves the current standard of materials engineering processes.

Contents

List of Figures

List of Tables

1

Introduction

Due to the ever-increasing capabilities of modern computers, algorithmic optimization has become a viable tool for solving increasingly complex engineering issues. This has expressed itself in, among others, the field of materials engineering. Recent studies have shown that the morphology and composition can be optimized using large sets of data or simulation software [1, 2]. These marked the transition from a conventional trial-and-error approach towards computational design.

However, many material engineering issues do not comprehend large datasets or data cannot easily be acquired. These are called data-scarce problems, whether the data comes from expensive computer simulations or labour intensive experiments. In these cases, the development of meta-heuristic strategies for data-scarce black-box optimization problems has recently revealed as an interesting research area. This thesis focuses on the development of these algorithms and their application to bio-based reinforced composite design.

Bio-based composites

Since the development of composite materials in the 1940s, climate change and the energy transition have become the main incentive for sustainable research. The production of conventional composites has a negative environmental impact in terms of energy consumption. Replacing conventional petrochemical composites with bio-based alternatives represents an opportunity to develop more sustainable materials for building and construction applications as well as for consumer products.

The research and development company NPSP B.V. based in Amsterdam, the Netherlands researches the use of natural materials in composites. New biobased composites are manufactured from natural and abundant materials. These can be residual

1

flows from the clothing industry, agriculture and landscape waste streams. By applying these fibres and a bio-resin instead of a petroleum-based resin, a biobased composite with a very low CO_2-footprint is created. The products range from parts of trains to benches and traffic signs. Figure 1.1 shows a variety of products by NPSP B.V. consisting of bio-based composite parts.

(a) Front of a train.

(b) Electric scooter body.

(c) Bio-based bench.

(d) Navigation sign.

Figure 1.1: A variety of bio-based composite materials from NPSP B.V. used in mobility, design and automotive applications [3].

However, replacing existing petrochemical-based products with bio-based alternatives involves important challenges. The manufacturing and testing of bio-based composites are very labour-intensive and time-consuming. Examining all combinations of natural fibres and fillers in different ratios by hand is not viable. Therefore, this thesis aims to combine data-scarce heuristic optimization with bio-based composite research, to accelerate the search towards optimal bio-based composite recipes.

Data-scarce optimization

When looking at a more general case of data-scarce optimization, it is often unclear which algorithm should be chosen for which type of problem. The 'No Free Lunch' theorem states that different algorithms excel in solving different types of black-box optimization problems [4]. In other words, a single meta-heuristic cannot outper-

form all others for every problem. Consequently, the choice of a meta-heuristic algorithm is often arbitrary or based on anecdotal evidence.

Besides, by definition, there is no prior information known about black-box optimization problems. Instead, problem-specific features are only discovered during and after the optimization process. Therefore the heuristic choice can only be evaluated after the process ends. In this thesis, we study if information gathered from a past optimization problem could give insight into new optimization problems.

In an attempt to use the information from previous black-box optimization problems, an effort is made to generate problem-specific optimization data. By collecting problem-specific information of various optimization benchmark problems and coupling this information to the best-performing meta-heuristics, we could gain insight into the domain of solvable problems for each heuristic. The prior information from generated optimization problems is stored in a database and aids us in solving new optimization problems. The heuristic decision for new optimization problems is dependent on the most successful heuristic from similar preceding cases. This way, we develop a data-driven heuristic decision strategy and comply with the message of the 'No Free Lunch' theorem.

2

Literature review

This thesis establishes two main objectives. The first goal of this thesis is to develop a computational model that supports the development of novel bio-based composites. By varying parameters of the bio-based composite recipes and inputting the acquired mechanical testing data in the data-driven model, the ambition is to assist experimentalist in finding biocomposites with tailored properties. The second goal is to create a data-driven optimization strategy to enhance heuristic-based optimizers when solving black-box optimization problems. This will be described in section 2.4.

2.1. Bio-based composites

Fibre-reinforced composite materials have been attracting attention for years as a replacement for conventional materials such as metals. Compared to conventional materials, composites have a higher density and specific strength, exhibit excellent corrosion protection and are more resistant to fatigue at a lower maintenance cost. However, composite materials subjected to impact failure are hard to repair [5].

Recently, environmental incentives have lead researchers to study alternatives to petrochemical materials in composites. Promising sustainability results were found among life-cycle-assessment analyses of bio-based alternative feedstock [6]. Waste-streams from other material processes can be given a new functionality by being part of a sustainable composite. This leads to higher recyclability of the feedstock and reduces the high cost of traditional composites [7]. In addition to matching properties of conventional materials by biomaterials, bio-based materials can also enhance material properties. For example, natural materials can add self-healing functionality, tackling one of the weak parts of traditional composite structures [8].

Natural reinforcements

Sustainable alternatives to replace these reinforcements have been intensely investigated. Table 2.1 shows the properties of the most widely used natural reinforcements and conventional glass-fibre and carbon-fibre reinforcements.

	Density (g/cm^3)	UTS (MPa)	E-modulus (GPa)	Elongation (%)	Cost ($/kg)
Hemp	1.4	550-900	70	1.6	0.7-0.8
Wood flour	1.5	1000	40	4.4	0.2-0.5
Jute	1.5	400-800	10-30	1.8	0.8-0.9
Flax	1.4	800-1500	60-80	1.2-1.6	0.6-0.8
Sisal	1.3	600-700	38	2-3	0.7-0.8
Kenaf	1.2	295	53	2.7-6.9	0.7-0.8
Bamboo	0.9-1.1	397-713	18-55	1.9	2.4-5.7
Glass	2.6	2000-3500	70	0.5	1.2-1.8
Carbon	1.4	4000	230-240	1.4-1.8	25-38

Table 2.1: Density, Ultimate Tensile Strength (UTS), Young's modulus (E-modulus), elongation and cost of commonly used natural fibres (top) and conventional synthetic fibres (bottom) [9–12].

It is apparent that natural fibres are much lighter than glass-fibre. Since natural fibres are part of a waste-stream, the cost is drastically lower than synthetic fibres. However, researchers have observed that bio-based reinforcements have limited utilization at heavy-load applications [13]. Conventional glass-fibre reinforced composites exhibiting far better loading properties than natural fibres [14]. However, the tensile strength and Young's modulus of natural fibres allow them to replace materials for moderate-load applications.

Besides, biomaterials are often hydrophilic. Natural fibres do generally have imperfect surface adhesion to the hydrophobic matrix material. The interface interaction between fibre and matrix controls the load transfer to the fibres. Water damage and bacterial attacks are also added risks to the material [13].

Several studies of composites with cellulosic fibres such as hemp, jute and flax show that these abundant natural fibres contain the potential to substitute conventional fibres in composites [15, 16]. One of the advantages of natural fibre composites is the possibility of using conventional processing equipment of thermoplastic-based systems with low maintenance costs. This is a consequence of the abrasiveness nature of cellulosic materials. An additional advantage is that the waste-stream of cellulosic processes can be given value. However, most natural fibres will degrade above 200 °C. This limits the choice of available processing techniques, and the choice of the matrix and filler materials [15].

In a study by Miller [6], various natural materials such as hemp linen, jute and wood flour were implemented as natural reinforcement fibres for biocomposites.

These bio-based composites were found to have competitive flexural and thermal properties compared with short glass fibre reinforced composites. Material selection and life cycle assessments showed that natural reinforced composites were comparable in impact resistance to conventional composites. Various review papers have united the scattered research on the implementation of natural fibres in composites [15, 16]. A short overview of natural fibre research over the past two decades is shown in table 2.2. For each natural reinforcement, the compatibility and performance with different matrices were investigated.

Fibre	Matrix	Reference	Noticeable results
Hemp	PHBV	[6]	High flexural strength, low thermal conductivity, high tensile strength, high tensile modulus
	PPE	[17]	
Wood flour	PHBV	[6]	High specific tensile strength, excellent protection against weathering
	PPE	[18, 19]	
Jute	PHBV	[6]	
	Epoxy	[20]	High flexural modulus, low thermal conductivity, high water uptake
	PE	[20]	
	PP	[17, 20]	
Flax	Epoxy	[20, 21]	High specific tensile modulus, low tensile and flexural strength
	PE	[20, 22]	
	PP	[20, 23]	
Sisal	PP	[17, 24]	Moderate impact strength, high water uptake
	PHBV	[25]	
	PE	[26, 27]	
Kenaf	PP	[17, 28]	High tensile modulus, high flexural strength, limited impact strength
	PE	[28, 29]	
	Epoxy	[28, 30]	
Bamboo	PE	[26, 27]	High tensile modulus, high specific tensile strength, low density
	Epoxy	[31, 32]	

Table 2.2: Selected research of common natural fibres implemented in different matrices: thermoplastic poly(3-hydroxybutyrate-co-3-hydroxyvalerate) (PHBV), polyphenyl ether (PPE) and polypropylene (PP), whereas epoxy and polyester (PE) resins are mainly used as thermosetting resin [15, 16].

Chemical treatment of fibres

The adhesion between fibre and matrix determines the load transfer between fibres. Natural fibres generally have poor adhesion to the matrix due to their hydrophilic nature. Research has been conducted to overcome this poor adhesion by having chemical treatment done to the fibres, reducing their hydrophilic nature, resulting in better load transfer. However, chemical modification of natural fibres adds to the overall cost of the biocomposite [15, 33].

Various chemical treatments can be applied to natural fibres. A concise overview of natural fibres with different chemical treatments is shown in table 2.3. The most widely used method is to soak the natural fibres in a sodium hydroxide solution (NaOH): the alkali treatment. The hydroxyl (OH) groups present at the surface

of the cellulosic fibres are broken down, which increases the moisture resistance properties. Additionally, the alkali treatment reduces the fibre diameter and thus increases the fibre aspect ratio. An increased aspect ratio will enhance the adhesion with the matrix material [33]. Another method involves the use of silane. Silane molecules will form links between the fibre surface and the matrix material through a siloxane bridge (Si-O-cellulose). The siloxane bridges develop a mechanically interlocked coating on the fibre [33]. Fibres treated with silane will provide better tensile strength than similar fibres with alkali treatment [34]. An acetylation treatment is used to improve the plastic behaviour of the fibres. The acetyl group (CH_3CO) reacts with the hydrophilic groups and takes out existing moisture. As a result, the dimensional stability is increased as well as the moisture resistance. Lastly, benzoylation is used to improve the natural fibre's interfacial adhesion and enhance the thermal stability [33, 35].

Fibre	Matrix	Treatment	Noticeable results
Hemp	PE	Alkali	Higher flexural modulus and flexural strength [36]
Wood flour	PP	Alkali	Slight increase in tensile modulus and strength [37]
Jute	VE	Alkali	Higher flexural modulus and strength [38]
	Epoxy	Alkali	Higher tensile strength, significant decrease of impact strength [34, 39]
	Epoxy	Silane	Better adhesion, higher flexural modulus and strength [34]
	PE	Alkali	Higher tensile strength, significant decrease of impact strength [34]
	PE	Silane	Better adhesion, higher flexural modulus and strength [34, 40]
Flax	Epoxy	Alkali	Increase of tensile modulus and strength [35]
	PP	Acetylation	Improved moisture resistance, decreased impact strength [41]
	PE	Benzoylation	Increased fibre surface area [42]
Sisal	PE	Alkali	Increased tensile and impact strength [35, 43]
	PE	Acetylation	Increased flexural strength [43]
	PE	Benzoylation	Increased tensile modulus and strength [44]
	Epoxy	Alkali	Higher tensile strength, significant decrease of impact strength [39]
	PCL	Alkali	Decrease in mechanical properties due to incompatibility [45]
Kenaf	PE	Alkali	Higher flexural modulus and strength [36]
Bamboo	Epoxy	Benzoylation	Increased tensile strength and modulus, decreased water absorption [46]
	PE	Benzoylation	Increased tensile strength and modulus, decreased water absorption [46]

Table 2.3: Selected research of chemical pre-treatment on natural fibres implemented in different matrices [33].

Multi-fibre systems

The variable properties of natural fibres can be combined to yield multi-purpose systems. An adequate co-fibre can counteract the disadvantages of another fibre component. Therefore, combining natural fibres to form hybrid systems is being investigated. Conventional synthetic fibres are often combined with a natural substitute to balance cost and performance. However, research on natural multi-fibre systems is also explored. The classes of natural fibres differ significantly in mechanical properties, as seen in table 2.1. Finding a compatible match of two or more natural fibres can lead to high-end natural composites. Nonetheless, compatibility between the two fibres and the matrix is crucial to accomplish enhanced performance [47]. Table 2.4 shows selected studies that manufactured and characterized natural fibre hybrid systems.

As seen in table 2.1, the properties of natural fibres are very distinct. Also, for the

Fibres	Matrix	Noticeable results compared to the single-reinforcement composite
Hemp/Glass	PP	Increase in specific fatigue strength [48]
Wood flour/Glass	PVC	Significant improvement in impact strength [49]
Jute/Glass	PE	Increase in specific flexural properties [50]
	Epoxy	Decreased water uptake [51]
Jute/Cotton	PE	Decreased water uptake [52]
Flax/Glass	PP	Increase in recyclability [53]
Flax/Cotton	PP	Increase in recyclability [53]
Flax/Silk	Epoxy	Increase in impact strength [54]
Sisal/Glass	PE	Decrease in cost while maintaining mechanical properties [55]
Sisal/Glass	Epoxy	Increase in chemical resistance [56]
Sisal/Cotton	PE	Incompatible due to an increase in water uptake [52]
Sisal/Banana	PE	Increase in tensile and flexural properties [57]
Sisal/Oil palm	NR	Increase in compatibility and tensile modulus [58]
Kenaf/Glass	Epoxy	Similar tensile and flexural properties, but decreased impact strength [59]
Bamboo/Glass	PE	Increase in specific tensile strength [60]
	Epoxy	Increase in specific tensile strength [60]
	VE	Comparable flexural modulus at low natural fibre content [61]
	PP	Increase in thermal stability and flexural modulus [62]

Table 2.4: Selected research of multi-fibre systems implemented in different matrices [47].

same fibre, different matrix materials can provide drastically different properties to natural fibres. Even the processing conditions play a significant role in the quality of the composite. A study with sisal fibre reinforced composites concluded that the mechanical properties are greatly dependent on the fibre orientation, length, and chemical treatment [63].

Moreover, environmental conditions highly determine the properties of natural fibres. Several studies showed a decrease in mechanical properties after a great extent of moisture uptake [11, 53, 64]. The temperature of the moisture greatly determined the decay of mechanical properties [65]. Also, natural fibres are affected by UV-light. The mechanical properties of jute fibre-reinforced composites under the influence of UV-light were tested by Gassan et al. [66]. Results showed an increase in the polarity of the fibres, gaining in flexural strength. However, excessive UV-treatment caused degradation of the fibres.

Matrix

As discussed previously, the matrix selection is bound by the maximum temperature that the natural reinforcement can handle. Due to this constraint, only matrix materials that soften below this temperature are suitable for bio-based composites [67].

Several attempts have been made to incorporate both a sustainable matrix and fibre in a composite, the so-called 'Green composites'. Nowadays, most conven-

tional polymer resins from sustainable feedstocks are commercially available with bio-based content up to 100% [68]. Polylactic Acid (PLA) has been identified as a promising biopolymer. Its excellent mechanical properties and acceptable thermal properties make it a sustainable alternative for polyethylene terephthalate (PET) [69]. Besides, PLA-based biocomposites possess three times more potential for recyclability [70]. In recent years, PLA-based bio-based composites are developed and mechanically characterized for all the above mentioned natural fibres [71].

Attempts to directly use natural resins in composites have been made by Mehta et al. [72]. Vegetable oils from soybean and cottonseed can be transformed into matrix resins that will polymerize when heated in the presence of a catalyst. Bio-based composites have been manufactured by combining vegetable oils with unsaturated polyester into hemp-fibre reinforced biocomposites. The impact strength of the polyester blend composite was improved by 90%.

However, bio-based resins are relatively expensive. A bio-based filler can serve as a cost-reducing agent as well as an extra reinforcement.

Natural filler

Developing a method to use waste-stream materials as fillers is particularly promising because it tackles both the recyclability and the cost of biocomposites. Besides, natural fillers play an important role in determining the density and rheological properties of biocomposites [13].

Replacing the conventional filler calcium carbonate ($CaCO_3$) with a natural equivalent results in a considerable weight loss, as natural fillers are generally far lighter than calcium carbonate [13, 73]. A challenge when replacing calcium carbonate is the decrease in viscosity of the unpaved composite mixture. Results have shown an increase of viscosity with $CaCO_3$ content [74]. In order to mould the composite into various complex shapes, the composite must flow sufficiently. Research is being conducted on various plasticizers to increase the viscosity [75].

In the past decade, a lot of different sources of natural fillers have been investigated. However, comprehensive research on particularly promising fillers and in-depth characterization is still uncharted territory [13]. Wood pulp, almond and apricot shell showed an increase in its mechanical properties compared to the bare polypropylene (PP) matrix [76]. In separate studies, composites with both egg-shell and shellfish filler material were compared to calcium carbonate composites. Both biocomposites were comparatively better than their calcium carbonate counterpart. An increase in Young's modulus but a decrease in tensile strength was observed for the egg-shell filler, whereas the shellfish filled composites showed an increase in tensile strength [77–79]. Studies with seaweed, tuff, nanocellulose and walnut showed an increase in crystallinity with increasing filler concentration. This results in a better tensile strength. An increase in flame retardancy was noticed for the seaweed biocomposites [80, 81]. Similar research was conducted with coffee silverskin filler in bio-based polyethylene (BioPE) composites. However, biocom-

posites with coffee silverskin demonstrated a decrease in crystallinity and ductility [82]. Incorporating biochar in a PLA-based composite increased thermal stability and decreased water absorption [83].

Due to the abundant waste-stream of olive stones in the Mediterranean region, research was conducted using ground olive stone as natural filler [84]. Incorporating olive stone in unsaturated polyester resin matrix composites enhanced the flexural modulus, but at the expense of impact strength. Surface modification with mercaptosilane (MRPS) helped to reduce the water sensitivity of the composite [85].

Research has been conducted on using pecan nutshell as reinforcement fillers to manufacture PLA based biocomposites [86]. The result showed a significant increase in the viscoelastic response of PLA by enhancing the flexural strength. However, a lower resilience concerning a plain PLA composite was measured due to the lack of chemical adhesion between the PLA and the pecan nutshell filler. Parbhakar et al. [87] manufactured epoxy bio-based composites with peanut shell filler material. The composites' morphology shows better bonding between the filler and resin, thus leading to improvement of the mechanical properties. Table 2.5 shows an overview of recent studies on biofillers.

Filler	Matrix	Noticable results
Almond shell	PP	Increase in specific strength and modulus [76]
Apricot shell	PP	Increase in specific strength and modulus [76]
Biochar	PLA	Improves the thermal stability and reduces water uptake [83]
Coffee silverskin	PE	Improves the elastic modulus and reduces strain [82]
Egg-shell	Epoxy	Improves the tensile modulus but reduces the tensile strength [78]
Nanocellulose	PHBV	Increase in crystallinity and tensile strength [81]
Olive stone	PE	Increase in stiffness, but at the cost of impact strength [84, 85]
Peanut shell	Epoxy	Increase in tensile strength and modulus [87]
Pecan nutshell	PLA	Increase in thermal resistance, flexural strength and modulus [86, 88]
Seaweed	PP	Improves the matrix compatibility and introduces flame-retardancy [80]
Shellfish shell	PP	Increase in tensile strength and modulus [79]
Tuff	PHBV	Increase in crystallinity, tensile strength and modulus [81]
Walnut shell	PHBV	Increase in tensile strength and modulus [81]
Wood powder	PP	Increase in specific strength and modulus [76]

Table 2.5: Selected research of natural fillers implemented in different matrices.

All sorts of abundant waste materials can be implemented as biofillers to serve as cost-reducing and reinforcement agent in composites. The trend of bio-based filler researches expresses the opportunities of natural filler reinforcements and underlines the challenge of compatibility between the natural filler and polymer matrix. Natural fillers' performance and properties are governed by conditions like the size-distribution, shape of the filler, chemical treatment, and interaction with the matrix [13]. Analogue to natural fibres, natural fillers' properties are also sensitive to environmental conditions [89].

Bio-based composite optimization

By reflecting on the comprehensive research done on natural materials, we conclude that abundant waste-streams can provide tailored material solutions to widespread applications. The choice of recommended materials is not as limited as conventional composites, where glass, carbon and aramid fibre reinforced composites dominate the market [14]. However, systematically researching every combination of natural materials is not a viable approach as the number of natural materials is enormous and their properties are outspread. This can be derived from the enormous amount of scattered research on natural fibres and fillers. Therefore, we have to resort towards pin-pointing specific needs of applications and optimize for that specific property.

The problem of the unpredictable behaviour of natural materials makes it less intuitive to search for an optimal recipe for a specific application. Nevertheless, finding an optimal combination of natural materials in an early stage of trial-and-error will significantly reduce the research workload and save enormous money. Here is where we connect the link with algorithmic optimization. The field of algorithmic optimization concerns selecting the best input for some set of available alternatives. Inputting different types and ratios of natural fibres, fillers and resins into an optimization algorithm and optimizing for a specific property could tailor new promising bio-based composites.

Although several studies try to model the specific behaviour of bio-based composites [90–93], we are not yet able to accurately model the complete behaviour of biocomposites. However, we can perceive the unpredictable performance as a black-box function. A black-box function is a function for which the analytical form is not known. Treating the physical and chemical behaviour of bio-based composites as a black-box function eliminates the issue of oversimplification in a conventional macro mechanical model [94]. Using a black-box optimization approach for the guidance of novel bio-based composite research is still unknown territory. It could give new insights into the advancement of bio-based composites.

Finding a suitable optimization algorithm that adapts to the bio-based composite case is a challenge since research in this field is ever-expanding. In the following section, we will dive deeper into the field of optimization.

2.2. Overview of optimization

An optimization problem consists of several components. First, we outline the objective function f that is desired to be maximized or minimized. If the problem consists of multiple objectives, a multi-objective optimization approach can be chosen. Alternatively, the objectives can all possess individual weights and be added together to reduce the multiple objectives to one single objective. [95]. Only single-objective optimization is assessed to keep the scope of the thesis concise.

The objective function depends on a number of input parameters represented as the vector $\vec{x} = (x_1, ..., x_d)$. These parameters can contain equality or inequality constraints, which reduce the feasible solution space X. For each iteration t, the

current solution $\vec{x}_t$ is altered to acquire a new solution $\vec{x}_{t+1}$. Ultimately, we want to find an optimal set of parameters $\vec{x}^*$ that will minimize or maximize the objective function $f(\vec{x})$. We can express a minimization optimization problem with the following formulation:

$$f(\vec{x}^*) \leq f(\vec{x}) \; \forall \; \vec{x} \in X \tag{2.1}$$

A minimization in $f(\vec{x})$ is the same as a maximization in $-f(\vec{x})$. For the sake of consistency, we will view the optimization as a minimization problem for f.

Solution quality and used resources

When solving an optimization problem, we are interested in the quality of the solution and computational resources used to originate the output. Preferably we want our optimization solution to be an exact solution. This will eliminate the variability of solution quality, as all other possible outcomes are considered to come up with the optimal outcome. However, exact methods are not always readily available for solving optimization problems. Furthermore, the required resources for any algorithm increase when the dimensionality of the problem increases. For larger problems, exact methods tend to demand more resources than is usually available.

In that case, there is a choice to sacrifice some of the solution quality to solve an optimization problem within the available resources [95]. One solution is to apply heuristic methods, despite not giving a guarantee of finding the exact solution of the optimization problem, they aim at returning a reasonably good solution. These methods assess the trade-off between solution quality and accessible resources.

In this context, evaluating the time complexity $\mathcal{O}$ of the optimization problem is important, irrespective of whether an exact or heuristic approach is feasible. The time complexity denotes the dependency of the problem size concerning the required resources. Several time complexity functions are shown in table 2.6.

Constant	$\mathcal{O}(c), c > 0$
Logarithmic	$\mathcal{O}(\log d)$
Linear	$\mathcal{O}(d)$
Quasilinear	$\mathcal{O}(d \log d)$
Quadratic	$\mathcal{O}(d^2)$
Polynomial (of order c)	$\mathcal{O}(d^c), c > 1$
Exponential	$\mathcal{O}(c^d), c > 1$
Factorial	$\mathcal{O}(d!)$
Super-exponential	$\mathcal{O}(d^d)$

Table 2.6: Table of various time complexities, divided into polynomial time (top) complexity and exponential (bottom) time complexity [95]. d refers to the dimensionality of the problem.

Generally speaking, the time complexity is divided into two classes. The first class consists of all the time complexity functions that are at most scaled by a polynomial expression $\mathcal{O}(d^c)$ where $c > 1$. When the problem size increases, the additional resources that are needed are manageable for most solving systems. This class is called polynomial-time algorithms, P-type for short.

If a polynomial-time algorithm exists for a specific optimization problem, we say that the problem lies in the P-type problem class. An example of a P-type problem is sorting a list of integers. Sorting algorithms have different best, worst and average time complexity expressions. For instance, the merge sort algorithm performs $\mathcal{O}(d \log d)$ as worst-case time complexity. This means that sorting a list can be done efficiently even with an increasingly large list size [96].

The second class consists of all other time complexities that scale at least exponentially with the problem size, $\mathcal{O}(c^d)$. Small problems can be solved exactly using these algorithms, but the requested resources will skyrocket when the problem size increases. This class is called non-polynomial time algorithms, NP-type for short.

A well-known NP-type problem is the Travelling Salesman Problem (TSP). This problem asks the following question: given a list of cities and the distance between them, what should be the shortest path to visit all cities? This famous NP-type problem finds its roots in 1930 and is still used as a benchmark case for heuristic optimization algorithms. Adaptations of this problem apply to vehicle routing software and GPS. Exact approaches to this problem are not able to go below $\mathcal{O}(2^d)$ and it has not been proved whereas an exact algorithm below this bound exists [97]. However, heuristic algorithms are capable of predicting a tour between millions of cities within 1% of the optimal solution [98].

Exact P-type optimization

Exact methods for P-type problems are frequently discussed in research and review articles. These exact optimization methods are effective if the problem meets one of the following strict specifications: the objective function and constraints are linearly or quadratically dependent on the input parameters, or the problem's search-space is entirely convex. More information on polynomial-time algorithms for optimization problems in this form can be found in appendix A.1.

However, using an exact optimization algorithm for the bio-based composite case is not preferable. The objective function can not be expressed linearly, as we do not know how the individual parameters will influence each other. Similarly, we do not have any information that suggests that the search space is entirely convex. Most importantly, evaluating all possible solutions is not viable. The production and testing of bio-based composite samples are very time-consuming. Evaluating a few points in the domain will take several days due to the labour-intensive process.

Heuristic NP-type optimization

Finding an exact solution for NP-type problems is not feasible, as the number of computational resources scales exponentially with the problem size. Therefore, heuristic algorithms, conventionally named heuristics, attempt to find a near-optimal solution within polynomial time. Although heuristics will not guarantee to find the optimal solution, they compromise solution quality for computational resources [95].

Due to the intense acceleration of innovation within heuristic optimization, its terminology lacks consistency. Rothlauf's book [95] tries to form modern heuristic techniques and distinguishes three forms of heuristics: standard heuristics, approximation algorithms and meta-heuristics.

- Standard heuristics are heuristic algorithms that are very problem-specific and can be divided into construction and improvement heuristics. Construction heuristics try to optimize the individual parameters separately to arrive at a single solution. In contrast, improvement heuristics start with complete solutions and adjust the entire solution to arrive at an optimum [95].
- Approximation algorithms are heuristics that guarantee quality bounds on a solution. As standard heuristics give no information about the obtained solution, this form attempts to quantify the solution quality.
- Meta-heuristics or also entitled modern heuristics are an extension to the improvement heuristics from standard heuristics. Improvement heuristics only try to improve the current solution with each iteration by exploiting the vicinity of good solutions. However, meta-heuristics differ in the fact that they also explore less-promising regions within the search space. This exploration characteristic acts as a way to break out of local minima. The duality between exploration and exploitation behaviour is an essential feature of meta-heuristics.

If no exact algorithm or approximation algorithm is known for a problem, meta-heuristics are the method of choice to tackle NP-type problems. Since meta-heuristics are designed to apply to a wide range of problems, it will broaden the chances of finding proper solutions. Therefore, the overwhelming demand for all-purpose heuristic solvers has opened up research for new and improved meta-heuristic techniques. Because, at the present time, research is mainly focused on developing meta-heuristics, we will be focusing on the meta-heuristics and refer to extensive literature reviews for the other heuristic forms [99].

Categorization of meta-heuristics

Many review papers have come up with different ways of classifying meta-heuristics. These can be summarized with the following categories:

- **Trajectory and discontinuous**: The distinction between trajectory and discontinuous methods is commonly made. With a trajectory method algorithm, consecutive solutions follow a search trajectory. A discontinuous method allows large jumps to avoid getting restricted to local minima. This distinction

can be viewed as the exploitation and exploration behaviour of the heuristic. A trajectory-based method will exploit a local minimum and iterate more solutions at a closer distance to further improve the solution quality. Discontinuous methods explore the solution space to increase confidence about the global optimum being located [100, 101].

- **Population and single-point**: The separation between population-based or single-point search methods is made. Population-based algorithms transform a group of solutions on each iteration. For single-point search methods, one single solution is altered at each step of the algorithm. Single-point search enhances the local search exploitation, whereas a population-based strategy helps the exploration behaviour of the algorithm [100, 101].
- **Nature and non-nature based**: A third way to classify meta-heuristics is based on the source of inspiration. Meta-heuristics tend to gain inspiration from biomimicry. Therefore, algorithms are categorized on nature-inspired and non-nature-inspired meta-heuristics. Salcedo-Sanz et al. [102] added the classification of meta-heuristics inspired to non-linear physical processes. He classified meta-heuristics based on, for instance, ideal gas modelling and electromagnetic theories.
- **Online and offline**: Powell [103] discussed in his research that every class of meta-heuristics is essentially categorized for either achieving the cumulative (online) or final (offline) reward. Secondly, the objective function should be considered either state-independent or state-dependent.

'No Free Lunch' theorem

It could be argued that the class of NP-type problems is so diverse that generalized algorithms which could solve many different problems should be sought. According to the 'No Free Lunch' theorem of Wolpert et al. [4], exceeding both applicability over a range of problems and maintaining high-quality solutions is not achievable. The study from 1997 benchmarks the general concept of a heuristic algorithm to a random search. A random search is a simple naive algorithm where, upon each iteration, random parameters in the search space are evaluated and compared to the best-found solution. The random search method behaves independently of the problem set and does not use problem-specific information. This states that on average, the algorithm will perform equivalently on every NP-type problem. Heuristic algorithms strive to perform better than the random search approach.

Wolpert et al. [4] demonstrated that whenever a particular heuristic algorithm gains in performance on one particular problem, it is bound to perform worse than a random search algorithm on other NP-type problems. This means that a heuristic will never perform better than random search on all NP-type problems simultaneously. Instead, different heuristics should tackle specific problem classes. Heuristics need to be designed so that problem-specific information is exploited. Making no use of problem-specifics will result in random search performance for the majority of existing problems.

As the 'No Free Lunch' theorem describes, a suitable algorithm must be found based on the problem's characteristics. However, categorizing meta-heuristics to their problem-specific traits is not straightforward. There has not yet been a systematic comparison between the exploited characteristic of a problem and the best performing algorithms on that particular problem class, to the best of our knowledge.

Currently, decisions about heuristic choices are made based on experience or performance at similar problems. Attempts have been made to optimize the meta-heuristic decision process itself. This has led to the development of hyper-heuristics.

Hyper-heuristics

Hyper-heuristics choose and alter a suitable meta-heuristic for an optimization problem and replace human expertise in the meta-heuristic decision process [104]. Hyper-heuristics can combine various meta-heuristics, in hyper-heuristic context referred to as low-level heuristics, to come up with a new high-level heuristic. In parallel, hyper-heuristics can optimize hyper-parameters in standard meta-heuristics to enhance their performance [105]. Ultimately, the hyper-heuristic algorithm analyses the currently used high-level heuristic every iteration and makes adjustments if necessary [104].

Roughly speaking, two features of a hyper-heuristic framework play an essential role in its behaviour and performance: the set of low-level heuristics and the move acceptance strategy [104].

Low-level heuristics set

The low-level heuristic set consists of several general-purpose heuristics. Each algorithm within the set can be selected, altered and combined to form a high-level heuristic. The high-level heuristic is then applied to solve the optimization problem until the move acceptance strategy requests another high-level heuristic. According to the 'No Free Lunch Theorem', the set of low-level heuristics should be diverse so that it can exploit a wide variety of different problems. The set of low-level heuristics should represent the main problem-specific categories of meta-heuristics.

Ochoa's study [106] proposed the hyper-heuristic framework HyFlex that provides the link between general-purpose meta-heuristics and specific problem domains. HyFlex is a tool for designing and comparing the performance of move acceptance policies. It has become the standard benchmark for comparing cross-domain search methods.

HyFlex does not have particular meta-heuristics implemented but does provide a framework to import low-level heuristics. Interestingly, HyFlex works with categories of low-level heuristics based on their underlying principles, unlike any other categorization paper. The categorization of low-level heuristics that are underlined are as follows:

1. **Mutational**: Mutational heuristics are heuristics that will perform a small

alteration on the solution by changing or swapping different solution components.

2. **Ruin-recreate**: Ruin-recreate heuristics will destroy some parameters of the current solution and rebuild them. Constructive heuristics can recreate the destroyed part of the solution.
3. **Hill-climbing**: Hill-climbing heuristics will iteratively improve the solution by making small changes in the direction of the more promising solution space. It will only accept non-deteriorating solutions until a local optimum is found.
4. **Crossover**: Crossover heuristics will take the parameters of two solutions and combine them to form a new solution.

Move acceptance strategy

A decision-making algorithm analyses, after every iteration, whether the current choice of heuristic has to be adjusted to improve the likelihood of converging to an optimum. This is executed bearing a 'move acceptance' strategy in mind. According to the 'No Free Lunch' theorem, the rule-set of the 'move acceptance' strategy should be based on problem-specific information gathered from past iterations.

Different move acceptance algorithms are proposed and discussed in various studies. For example, move acceptance methods could be stochastic or deterministic. The decision algorithm can also accept moves that do not necessarily move closer to the optimum. 'Improving or Equal' (IE) is a deterministic move acceptance method that only accepts moves that will make the next step's objective value better than the current solution. Whereas 'Simulated Annealing' (SA) will also accept moves to some probabilistic extent that will decrease the solution quality [105].

Generally, the underlying principle of move acceptance algorithms is a black-box optimization itself [107]. Possible moves are applied to the current state, and the decision algorithm picks a move based on the objective function outcome of the move.

2.3. Selected meta-heuristics

As discussed in section 2.2, there has been made much progress within the field of meta-heuristic. Despite the rapidly evolving innovation of meta-heuristics, this also brought negative trends in the optimization domain. The research from the beginning of this millennia contributes to a flood of metaphor-inspired heuristics. The critical review from Sörensen on these metaphor-inspired trend outlines the similar build-up of these papers [108]. A new 'revolutionary' meta-heuristic is proposed based on an inspiring biological phenomenon. After that, the heuristic is poorly benchmarked against a hand-picked set of problems so that the new heuristic comes out best. When studying these papers, the heuristics are mainly rebranded existing approaches that may have slightly altered hyper-parameters. The benchmarking process selectively displays only successful cases, which does not honour

the definition of benchmarking. Both the 'Grey Wolf Optimizer' and 'Harmony Search Algorithm' have been accused of imitative research [109, 110].

Filtering innovative research out of the trivial copies has become challenging as the amount of meta-heuristics is rising. Besides, focussing on one particular category of meta-heuristics will bias the reader. A meta-heuristic from each of the four low-level heuristic categories from HyFlex is picked and discussed to give an overview of relevant meta-heuristics research. These heuristics are selected based on their innovative principles, frequent citations and distinct features. As most of the selected algorithms have features of multiple low-level heuristic categories, they are selected based on their most prominent feature. Bayesian Optimization is included in this study because of its promising results in recent studies [111–113] and unique feature of constructing a surrogate model of the objective function. Due to this unmatched characteristic, it is not connected with one of the low-level heuristic categories.

Table 2.7 gives an overview of the heuristics that will be discussed in this section. After a brief overview and explanation of the update strategy, each heuristic performance on a benchmark problem is demonstrated.

Meta-heuristic	Category	Reference
Particle Swarm Optimization (PSO)	Crossover	[114]
Covariance Matrix Adaptation (CMAES)	Ruin-recreate	[115]
Adaptive Moment Estimation (Adam)	Hill-climbing	[116]
Bayesian Optimization (BO)		[117]

Table 2.7: Selected heuristics categorized based on their most prominent feature.

Particle Swarm Optimization (PSO)

This population-based algorithm was proposed by Kennedy & Eberhart in 1995 and gained inspiration from the movement of individuals in a school of fish or bird flock [114]. For each iteration, new improvement solutions are proposed according to rules that incorporate their previous position.

For the original particle swarm optimization (OPSO), the following update rules are applied. The new position $\vec{x}_{t+1}$ of each individual particle is determined by the position of this particle $\vec{x}_t$ plus a velocity term $\vec{V}_{t+1}$.

$$\vec{x}_{t+1} = \vec{x}_t + \vec{V}_{t+1} \tag{2.2}$$

This velocity term $\vec{V}_{t+1}$ is composed out of three terms that each serves a different purpose:

- Particle velocity $\vec{V}_t$: this is the velocity vector of the particle in the current iteration.

- Cognition $\vec{p}_t$: this term is the influence on the best position that this particle has visited in its lifetime.
- Social $\vec{g}_t$: the social term serves as a memory of the global best position considering all particles.

After each iteration, the velocity term is updated according to the following formula:

$$\vec{V}_{t+1} = \vec{V}_t + \phi_1 R_{1t}(\vec{p}_t - \vec{x}_t) + \phi_2 R_{2t}(\vec{g}_t - \vec{x}_t) \tag{2.3}$$

The cognition and social components have two real number weights ϕ_1 and ϕ_2 as hyper-parameters to tweak their respective behaviour of the optimization. Moreover, both components are multiplied with a factor R_{1t} and R_{2t}. These are vectors where their elements are random numbers distributed uniformly ($\sim U(0,1)$). Every iteration, R_{1t} and R_{2t} are computed again. This randomization acts as a variable step-size for the cognitive and social components. Particle Swarm Optimization is heavily subjected to the crossover principle, as the update rule is essentially an addition of three existing solutions: the momentum of the current position ($\vec{V}_t$), the local best particle position ($\vec{p}_t$) and the global best particle position ($\vec{g}_t$).

Over the years, many variants of PSO have been proposed. Inertia Particle Swarm Optimization (IPSO) added an inertia weight ω that scales the particle velocity term [118]. Moreover, a widely used standardized and benchmarked version SPSO2011 has been developed [119, 120]. This variant introduced a spherical distribution that is added to the velocity update term.

Particle Swarm Optimization is vulnerable as a particle approaches the solution boundary. Several ways to handle this situation are described in the paper of Padhye et al. [121].

In the original paper of Eberhart [114], the particle swarm optimization heuristic has been benchmarked against the single-objective non-linear Schaffer F6 benchmark function. This function contains a lot of local minima in a confined space and is generally challenging to optimize [122]. Appendix A.2.1 shows a visual representation of the 2D Schaffer F6 function and the performance of PSO with altering hyper-parameters and population size.

Covariance Matrix Adaptation (CMAES)

Covariance Matrix Adaptation-Evolutionary Strategy (CMAES) is an altered version of the Covariance Matrix Adaptation strategy proposed in 1996, classified as an evolutionary strategy for derivative-free global optimization [115].

Evolutionary strategy algorithms are a class of optimization methods based on the evolutionary principles found in nature. The optimization resembles natural selection, where survival of the fittest is the key principle. Its objective function evaluation value represents the fitness of an individual. We strive for the fittest individual

for a maximization problem (and the least fit for minimization).

In CMAES, each new population of search points is described by a multi-variate normal distribution $\mathbb{N}$:

$$\vec{x}_{t+1} \sim \mathbb{N}(\vec{m}_t, \mathbf{C}_t) \tag{2.4}$$

Here, $\mathbb{N}$ is a multi-variate normal search distribution, composed of a mean ($\vec{m}_t$) and a covariance matrix $\mathbf{C}_t$. These two values describe how the next generation is sampled.

Firstly, the new mean $\vec{m}_{t+1}$ is calculated by picking the weighted average of the μ best-performing points from the sampled search distribution $\vec{x}_t$.

$$\vec{m}_{t+1} = \sum_{i=1}^{\mu} \omega_i \vec{x}_{i,t}$$

whereas,

$$\sum_{i=1}^{\mu} \omega_i = 1, \omega_i > 0 \text{ for } i = 1, ..., \mu \tag{2.6}$$

where $\vec{\omega}$ is calculated by normalizing the Euclidean distances from the new mean $\vec{m}_{t+1}$. The update step for the covariance matrix is as follows:

$$\mathbf{C}_{t+1} = \sum_{\mu}^{i=1} \omega_i (\vec{x}_i^{t+1} - \vec{m}^t)(\vec{x}_i^{t+1} - \vec{m}^t)^T \tag{2.7}$$

What makes the CMAES algorithm unique is the calculation of the new covariance matrix $\mathbf{C}_{t+1}$. The covariance of each individual is calculated relative to the true mean value $\vec{m}_t$ from which the current distribution is sampled. Note that this differs from an empirical approach where the mean value of the sampled points is used ($-\frac{1}{\lambda}\sum_{i=1}^{\lambda} x_i^{t+1}$).

Secondly, we multiply the individual covariance elements by their respective weights ω_i, as calculated in equation 2.7.

CMAES incorporates a way to include information from the history of generations by the so called 'Rank-μ-Update'. After a sufficient number of generations, the mean of the covariance matrices from all generations is used to get a reliable estimate for the covariance matrix described in 2.7. A full mathematical description of the Rank-μ-Update can be found in the source material [123].

However, the sign of the steps derived from the 'Rank-μ-Update' is not used for calculating the covariance matrix. To incorporate this information, the 'evolutionary path' is introduced.

The evolutionary path is expressed as the sum of the true mean values of consecutive generations. The direction of this evolution path ($\vec{p}_c^{\,t}$) is exploited by accelerating in this direction. This enhances the behaviour of CMAES on objective functions with elongated flat valleys [123].

Combining the CMAES update step of the covariance matrix, the 'Rank-μ-update' and evolutionary path approach gives us the final update strategy of the covariance matrix:

$$\mathbf{C}_{t+1} = (1 - c_{\text{cov}})\mathbf{C}_t + \frac{c_{\text{cov}}}{\mu_{\text{cov}}}\,\vec{p}^{\,c}_{t+1}(\vec{p}^{\,c}_{t+1})^T + c_{\text{cov}}\left(1 - \frac{1}{\mu_{\text{cov}}}\right) \times \sum_{i=1}^{\mu} \omega_i \left(\frac{x_i^{t+1} - \vec{m}^t}{\sigma_t}\right)\left(\frac{x_i^{t+1} - \vec{m}^t}{\sigma_t}\right)^T \tag{2.8}$$

According to the proposed literature, the hyper-parameters μ_{cov} and c_{cov} are by default set to $\mu_{\text{cov}} = \mu_{\text{eff}}$, which is the variance effective selection mass, and $c_{\text{cov}} \approx \min(\mu_{\text{cov}}, \mu_{\text{eff}}, d^2)/d^2$, where d is the dimensionality of the optimization problem [123].

The nature of sampling from a normal distribution makes the ruin-recreate characteristic the main feature of the CMAES update rule. Existing promising solutions are selected by their objective function value and their mean and covariance matrix is built before these solutions are ruined. Subsequently, new solutions are created with the updated normal distribution parameters.

In the original paper of Hansen [115], CMAES is compared to various simple evolutionary strategy algorithms on the single-objective unimodal Rosenbrock function. This well-known test function is still a relevant benchmark for various algorithms. Appendix A.2.2 shows a visual representation of the 2D Rosenbrock function and the performance of the CMAES optimizer with varying population size.

Adaptive Moment Estimation (Adam)

Adam is classified as a stochastic gradient-descent algorithm. Stochastic gradient-based algorithms are one of the earliest approaches to heuristic optimization. Suppose one has access to the derivatives of the objective. In that case, gradient descent algorithms are relatively efficient optimization methods if the gradient's computation is as expensive as evaluating the objective function itself.

The main principle of gradient descent algorithms can be summarized in three steps. First, the optimization starts at an initial position in the search space. Then the derivative $\frac{\partial f(x_t)}{\partial x_t}$ on that particular point is evaluated. According to the derivative, the direction that has the most descending trend is determined as the moving direction. Lastly, the following evaluation is sampled in the descending direction multiplied by a step-size called the learning rate α. Stochastic Gradient Descent

(SGD) is one of the earliest approaches that is still widely used when incorporating the derivative in the optimization process.

$$\vec{x}_{t+1} = \vec{x}_t - \alpha \cdot \frac{\partial f(x_t)}{\partial x_t} \tag{2.9}$$

However, SGD becomes impractical when navigating across surface curves where one direction is much steeper than another. Improvements by adding a momentum term m_t and using past iteration in controlling this momentum term are proposed to overcome this shortcoming [124].

$$m_t = \beta m_{t-1} + (1 - \beta) \cdot \frac{\partial f(x_t)}{\partial x_t} \tag{2.10}$$

$$\vec{x}_{t+1} = \vec{x}_t - \alpha m_t \tag{2.11}$$

The hyper-parameter β controls the influence of momentum to the gradient descent.

A challenge for gradient descent algorithms is the choice of a proper learning rate α. Using a constant learning rate, the algorithm is restricted to one step-size and finetuning of this hyper-parameter is required to get useful results. Most of the time, optimizing for hyper-parameters is not feasible for new problems. Adaptive learning rate strategies try to adjust the learning rate during optimization. AdaGrad is a gradient-descent algorithm that changes the default learning rate for each parameter according to its history of computed derivatives [125]. The adaptive learning rate v_t is decreased for frequently changing derivatives and increased for dimensions that have flat slopes.

$$\vec{v}_t = \vec{v}_{t-1} + (\frac{\partial f(x_t)}{\partial x_t})^2 \tag{2.12}$$

$$\vec{x}_{t+1} = \vec{x}_t - \frac{\alpha}{\sqrt{v_t + \epsilon}} \cdot \frac{\partial f(x_t)}{\partial x_t} \tag{2.13}$$

The parameter ϵ is a small floating-point value that ensures the fraction is never divided by zero.

The learning rate update rule v_t of AdaGrad accumulates squared gradients of past iterations. Since the inverse of these squared gradients is stored, the learning rate keeps shrinking during its lifetime. Ultimately, the algorithm does not acquire additional information and is put on hold. This addresses the challenge of developing learning rate update rules that account for this aggressive decay. RMSProp is an unpublished gradient descent method developed as an extension of AdaGrad [126]. The update rule $\vec{x}_{t+1}$ takes the same form as equation 2.13. However, it restricts

the influence of the past squared gradients to a fixed window. RMSProp stores this sum of squared gradients as a decaying average of all past squared gradients. Therefore, the average at the current iteration only depends on the previous average and the current gradient.

$$\vec{v}_t = \beta \vec{v}_{t-1} + (1-\beta)(\frac{\partial f(x_t)}{\partial x_t})^2 \tag{2.14}$$

The hyper-parameter β controls the influence of previously acquired derivatives, similar to the momentum update term.

Adam is a stochastic gradient-based optimization technique designed to combine the advantages of AdaGrad and RMSProp. Adam is derived from Adaptive Moment estimation and was proposed in a study by Kingma and Ba in 2015 [116]. Adam uses individual adaptive learning rates for both the gradients and squared gradients. The method combines the principle of the momentum m_t from equation 2.10 and the fixed window of past squared gradients v_t from RMSProp (equation 2.14).

$$\vec{x}_{t+1} = \vec{x}_t - \frac{\alpha}{\sqrt{\hat{v}_t + \epsilon}} \cdot \hat{m}_t \tag{2.15}$$

The developers of Adam noticed that by initialization of the gradient descent, the moving averages m_t and v_t are starting out as vectors of 0's. This leads to the same learning rate shrinkage of AdaGrad. To counteract this behaviour, the exponential moving averages are bias-corrected by the following expression:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \quad , \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{2.16}$$

The exponential moving averages of the gradient ($\hat{m}_t$) and the squared gradient ($\hat{v}_t$) are controlled by the hyper-parameters β_1 and β_2 respectively. Similar to the momentum gradient descent and RMSProp, they control the influence on previously acquired derivatives.

Adam can be classified as a hill-climbing algorithm. Despite having a momentum term that gives the algorithm the ability to accept a deteriorating solution, the update rule is bound to improve the solution by making small changes in the more promising solution space.

The original paper of Kingma [116] demonstrates that Adam can be used to optimize different deep learning models, including logistic regression and multi-layer neural networks. To keep the performance demonstration over the selected heuristics in the same scope, we are looking at Adam's behaviour at single-objective functions where the analytical form is known. For example, the Beale function is a multimodal smooth objective function used for demonstrating the performance of

various gradient-based algorithms [102, 127]. Appendix A.2.3 shows a visual representation of the 2D Beale function as well as its performance and the influence of the hyper-parameters α, β_1 and β_2.

Bayesian Optimization (BO)

Bayesian Optimization is a powerful optimization strategy that is most effective if the objective function is expensive to evaluate [117]. Bayesian Optimization is built on Bayes' theorem [128]. This theorem states that the posterior probability of a model M, given information E is proportional to the likelihood of E given M multiplied by the prior probability M.

$$P(M|E) \propto P(E|M)P(M) \tag{2.17}$$

This translates to the information E being the acquired data from function evaluations in terms of an optimization problem. Consider an unknown objective function f. As we accumulate more information by function evaluations, our prior belief E of the objective function could state that the function is, for instance, noise-free and smooth. Subsequently, the objective function can be estimated by a surrogate function M that captures the prior beliefs. Based on our surrogate model, objective values that carry out high oscillations are less likely than those close to the mean value of E. This posterior probability captures our beliefs about the unknown objective function given information E.

Bayesian Optimization consists of three steps:

1. A posterior surrogate model is constructed from prior information.
2. An acquisition function is composed and evaluated within the search space.
3. The acquisition function is optimized to select the next point at which to evaluate.

Surrogate model: Gaussian process

Several ways to construct a surrogate model from data can be used for Bayesian Optimization. A popular surrogate model is a Gaussian process (GP). A Gaussian Process is an extension to a multivariate Gaussian distribution. A Gaussian distribution is specified by a mean and covariance and addresses a distribution over a random variable. Analogue to the Gaussian distribution, a Gaussian process characterizes a random function and is specified by a mean function m and a kernel k, often named the covariance function. The kernel takes two points, $\vec{x}_n$ and $\vec{x}_m$, as an input and returns a similarity measure between those points [111, 129].

$$f(\vec{x}) \sim \mathrm{GP}(m(\vec{x}), k(\vec{x}_n, \vec{x}_m)) \tag{2.18}$$

The Gaussian Process can be customized by choosing an appropriate mean function and kernel. The covariance function's choice describes the characteristics and fea-

tures of the objective function that we want to predict. Numerous different kernel functions, such as Radial basis function (RBF) or periodic kernel, can be used to cover all sorts of function characteristics [130].

When the objective function is not yet evaluated, the prior distribution does not contain any information. While evaluating more points, the Gaussian Process will be constrained on the evaluated points. By adding more information to the prior distribution, the Gaussian Process will evaluate the likelihood of objective values given the history of evaluations. This will alter the Gaussian Process by constraining it to the training points.

Acquisition function

For Bayesian Optimization, we are merely interested in finding the optimal parameters for minimizing the objective function. The acquisition function will guide the search for an optimum. The maximum value of the acquisition function will correspond to the next parameters to evaluate. This function needs to be cheap to maximize.

Several acquisition functions are proposed in the literature. The main trade-off within different acquisition function is the exploitation and exploration behaviour. The 'maximum probability of improvement' (MPI) and 'Lower Confidence Bound' (LBC) acquisition functions are greedy. It searches for parameters where the surrogate mean value is the highest. This acquisition is purely exploitative. On the other end of the spectrum, the 'Upper Confidence Bound' (UCB) acquisition function favours points where the surrogate covariance is the largest. This acquisition function behaves entirely explorative. The 'Expectation of Improvement' (EI) acquisition function balances this exploitation and exploration behaviour [112].

The Bayesian Optimization procedure is as follows. A acquisition function is constructed based on the surrogate model of the acquired information $\vec{x}_t$ and $f(\vec{x}_t)$. The next sampling point $\vec{x}_{t+1}$ is proposed by optimizing the acquisition function. Subsequently, an objective value $f(\vec{x}_{t+1})$ is acquired. Lastly, the new point is added to the acquired information and the surrogate model is updated.

The standard Branin-Hoo function is commonly used to analyze the performance of Bayesian Optimization on single-objective functions, [130]. This test function has three local optima. With an implementation of Bayesian Optimization of the Python library `GPyOpt` [131], the optimum in figure A.11b is sought. Its response-surface and performance with different kernel and acquisition functions are illustrated in appendix A.2.4

Heuristic choice for bio-based composite case

The choice of a suitable heuristic for the bio-based composite case is based on the information we have gathered before doing iterations on the problem. By performing iterations on the problem, we gather information about the search space and the heuristic response. This information may help us make a better heuristic choice, but it takes up much time due to the time-intensive nature of manufacturing bio-

based composites. As a result, there is a trade-off between the heuristic choice's accuracy and the time to acquire problem-specific information. Because we could consider the bio-based composite case as a very expensive objective function, and due to the time constraints on this project, we choose a suitable heuristic before the optimization process performed in chapter 3.

Nevertheless, we can consider that the bio-based composite problem is highly noisy due to the high variability of natural materials described in section 2.1. Therefore, the choice of using Adam, for which its update strategy is highly dependent on the gradient information of the response surface, is not justifiable.

In review papers [111–113], Bayesian Optimization is appointed as a promising novel heuristic, which has been shown to outperform other state-of-the-art global optimization algorithms on several challenging optimization benchmark problems. Bayesian Optimization handles noisy objective functions very well and is very information-efficient. It uses all the gathered information during its search to build a fitting surrogate model and suggest the next iteration location carefully. Bayesian Optimization works well with box-constrained boundaries. For CMAES and Particle Swarm Optimization, the update rule can force iterations outside of the objective function's boundaries. For Bayesian Optimization, no boundary behaviour needs to be implemented.

Bayesian Optimization has two significant drawbacks. Firstly, the time-complexity of building a surrogate model scales with the number of iterations n with $\mathcal{O}(n^3)$ due to an inversion of an $n \times n$ matrix [129]. Therefore, the time-efficiency for Bayesian Optimization reduces with the number of iterations, and generally for more than 10^4 iterations Bayesian Optimization is not a viable approach. In this thesis' context, this is not a problem. Since the manufacturing and testing of the recommended bio-based composite recipes have to be extended over multiple days, the number of iterations will not be in the order of 10^4.

Secondly, handling optimization in higher dimensions is a key challenge for Bayesian Optimization. Optimizing the kernel parameters requires many evaluations of the surrogate model and constitutes a computational bottleneck in Bayesian Optimization, although studies have been conducted to scale down the curse of dimensionality of a Bayesian Optimization problem [132]. Nevertheless, as the algorithmic implementation towards bio-based materials science has not been conducted earlier, the optimization implementation will start with low dimensionality to develop a proof-of-concept model.

In conclusion, Bayesian Optimization is chosen as the heuristic for optimizing bio-based composites so that this challenge could be addressed within an appropriate timeline for this thesis. However, in parallel, we are also interested in investigating a strategy to choose among different heuristics.

2.4. A data-driven heuristic decision strategy

Shortcomings of present heuristic optimization

When studying heuristic optimization literature, first-hand results show that meta-heuristic models such as the ones shown in appendix A.2 are impressively successful optimization tools that can solve a wide variety of problems within a short amount of time. However, behind the remarkable conclusions lie fundamental shortcomings. Namely, preliminary hyper-parameter tuning, the inconsistent classification system and the lack of a clear black-box heuristic decision strategy are other major complications in the meta-heuristic field that hold back the true capabilities of stochastic optimization.

Hyper-parameter tuning

The first problem that arises from modern heuristic literature is the choice of hyper-parameters. Hyper-parameters can drastically alter the behaviour of the algorithm performance. This has been addressed very early in the case of genetic algorithms [133]. In the following years, extensive research has been done on optimizing these hyper-parameters. By parametrizing the hyper-parameters and operating the optimizer repeatedly, the optimization performances are depicted as loss surfaces. Different approaches to optimizing these hyper-parameter loss landscapes have been proposed, including sequential model-based Bayesian Optimization and gradient-based optimization [134, 135]. To decrease computational time and improve convergence, even a random search approach on hyper-parameter tuning can be beneficial [136]. Results from several hyper-parameter tuning frameworks show that using optimal hyper-parameters in contrast to hand-picked hyper-parameters dramatically improves the performance on black-box benchmark problems [137].

When applying well-reported heuristics to new problems, if the hyper-parameters are not reported, it can be challenging to replicate the articles' results. Optimizing the hyper-parameters to specific problems is often done preliminary before reporting reasonable solutions. This process is very time-consuming and frustrating for new users that want to apply meta-heuristics to solve different problems. When adding the resources used for tweaking hyper-parameters to the algorithm's overall performance, general-purpose heuristics generally misrepresent their performance [95].

Conventional classification

Classifying the meta-heuristic algorithms in chapter 2.2 may be insufficient or impractical. For example, a problem arises if we develop a new optimization problem from which there is no P-type algorithm known. As the 'No Free Lunch Theorem' describes, different heuristics should solve diverse problem sets. Therefore, the appropriate use of a meta-heuristic is crucial to satisfy its performance. The conventional classification also does not guide us towards an appropriate choice of meta-heuristic. Unlike exact optimization, this heuristic classification system is based on the aesthetics of the algorithm and not of the problem itself [95].

In contrast, the categorization of low-level heuristics from HyFlex [106] classifies

distinct groups of meta-heuristics based on their underlying principles and disallows any aesthetics in its analysis. These distinct groups of heuristic principles could serve as a starting base for improved categorization. However, the link between meta-heuristic principles and problem-specifics is still to be made.

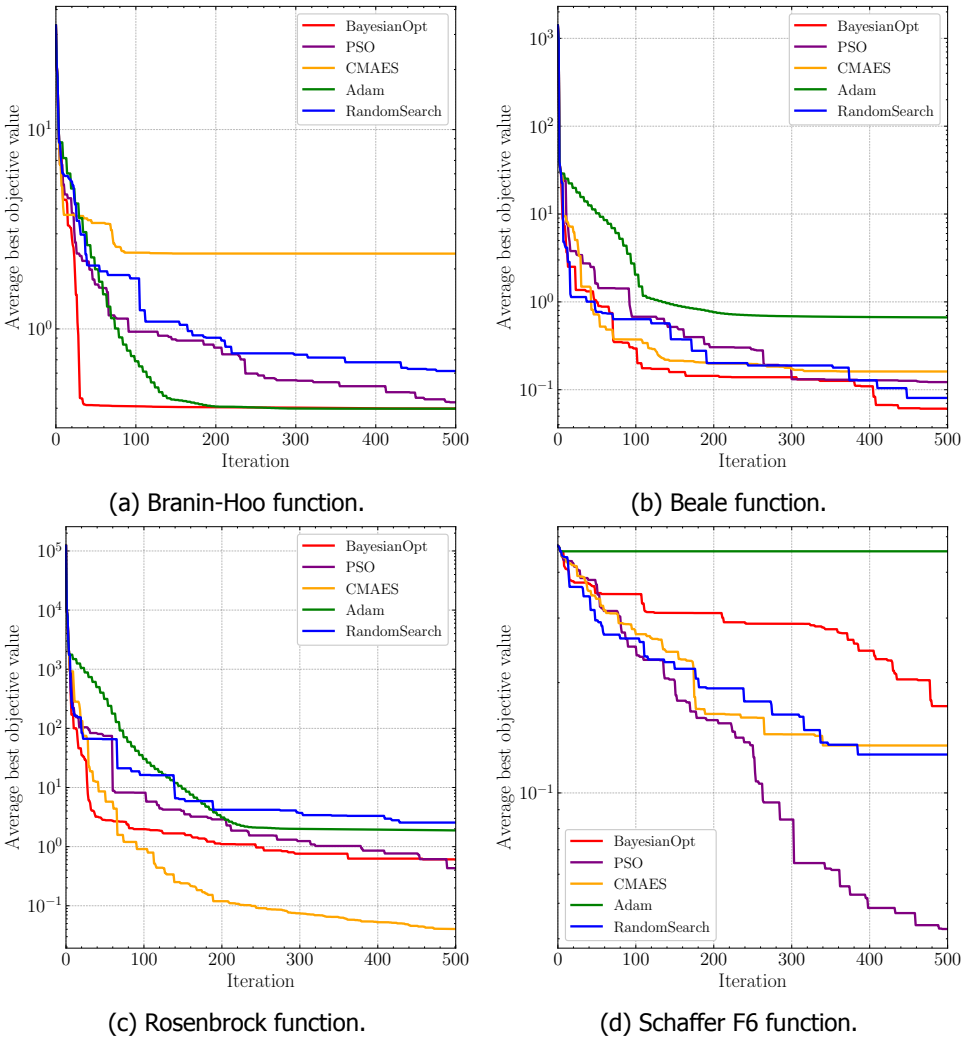
Heuristic decisions

The move acceptance strategy of a hyper-heuristic ensures that we can link the different meta-heuristics groups to different specific properties of a problem. However, the known move acceptance algorithms do only use problem-specific information in a limited manner. That is, by scaling the quality of the acquired solutions to the previously-best solutions [107]. When transitioning from one meta-heuristic to another during the same optimization problem it is also not possible to do it stochastically or deterministically. However, what is lacking in today's move acceptance strategies, is incorporating the entire history of evaluations towards the heuristic decision. Only then we can comply with the message of the 'No Free Lunch' theorem.

Data-driven optimization

Optimization benchmark papers come to contrasting conclusions about the performance of different heuristics. In the study of Vesterstrom [138], several evolutionary algorithms, particle swarm optimization and differential evolution are compared on a set of benchmark problems. The author concluded that the performance of the differential algorithm heuristic is outstanding in comparison with the other algorithms tested. However, in two different studies [139, 140], CMAES was marked for being the heuristic of choice despite benchmarking on a similar set of optimization problems. Even if we evaluate the benchmark function from the selected heuristics for all the other heuristics, the results do not provide us with a clear dominance of one heuristic. The results are illustrated in figure 2.1.

Sörensen put an important recommendation in his paper about the critique on the metaphor heuristic trend: *"Perhaps a set of tools is needed, i.e., a collection of statistical programs or libraries specifically designed to determine the relative quality of a set of algorithms on a set of problem instances"* [108]. This quote guides future meta-heuristic research towards a systematic data-driven comparison.



(a) Branin-Hoo function.

(b) Beale function.

(c) Rosenbrock function.

(d) Schaffer F6 function.

Figure 2.1: Performance of all the selected heuristics and a naive random search on the Branin-Hoo, Beale, Rosenbrock and Schaffer F6 benchmark functions.

Learning to learn

A new emerging branch of science in the direction of algorithmic optimization is meta-learning or learning to learn. Current meta-heuristics are hand-engineered algorithms. Researchers learn empirically which kind of optimization problems are suitable for which meta-heuristics. With the emergence of machine learning approaches, it is possible to learn an optimization algorithm based on a wide range of learning tasks.

Automatic algorithm design is a promising development in the field of heuristic

optimization. Li and Malik's paper is one of the leading studies in which the algorithm is designed through automatic execution [141]. The idea of learning the update step for the optimization algorithm itself in a reinforcement learning environment has been independently proposed similarly by Andrychowicz et al. [142]. The model rewards a modification to the update policy if better solutions are found and penalizes modifications that result in worse solutions. In this way, an optimal update policy for a particular optimization problem is acquired. As the update policy is trained by previously acquired optimization results, the 'learning to optimize' approach is also data-driven.

Figure 2.2 explains the idea of learned optimizers illustratively. The Gradient Descent algorithm is vulnerable in parabolical, narrow valley regions where the response-surface gradient in one dimension is far steeper than in another dimension. This problem-specific feature is prominent in the response-surface of the Rosenbrock function (as seen in figure A.4a). Gradient Descent responses oscillate heavily in one direction and move very slowly towards the optimum direction. The Momentum variant of Gradient Descent proposes to overcome this shortcoming by adding a momentum term. Hence, the movement in the flat direction is accelerated. The learned optimizer learns from the algorithmic responses of this problem-specific feature by penalizing the Gradient Descent's feedback and promoting the Momentum approach in a reinforcement learning manner.

The idea of learning to learn has expressed itself in the development of recurring neural networks (RNN) that are trained by various sets of meta-problems. The study of Chen et al. [143] trained an RNN to perform black-box global optimization and compared its performance against Bayesian Optimization. The learned model has shown to be more efficient in terms of computational resources and could compete on a wide class of black-box functions. Around the same time, a meta-learning algorithm that is compatible with any model trained with gradient descent is proposed [144]. This algorithm aids the training of a meta-learning model towards finding model parameters that are sensitive to changes.

Wichrowska et al. [145] introduced a similar meta-learning RNN model that is more focussed on reducing memory and computation overhead. Their study introduces a meta-training set that consists of an ensemble of diverse loss landscapes for which an RNN captures the dependency of the gradient-descent hyper-parameters. Subsequently, popular hand-designed gradient-based optimizers are incorporated as starting points. The learned optimizer matches or outperforms Adam and RMSProp on problem types from the meta-training set.

Although most learned optimizers are focussed on single-solution derivative-based optimization, the study of Cao et al. [146] designed an RNN that handles both point-based and population-based training data. Their results showed that the meta-learning approach outperforms Particle Swarm Optimization on numerous convex and non-convex Rastrigin problems.

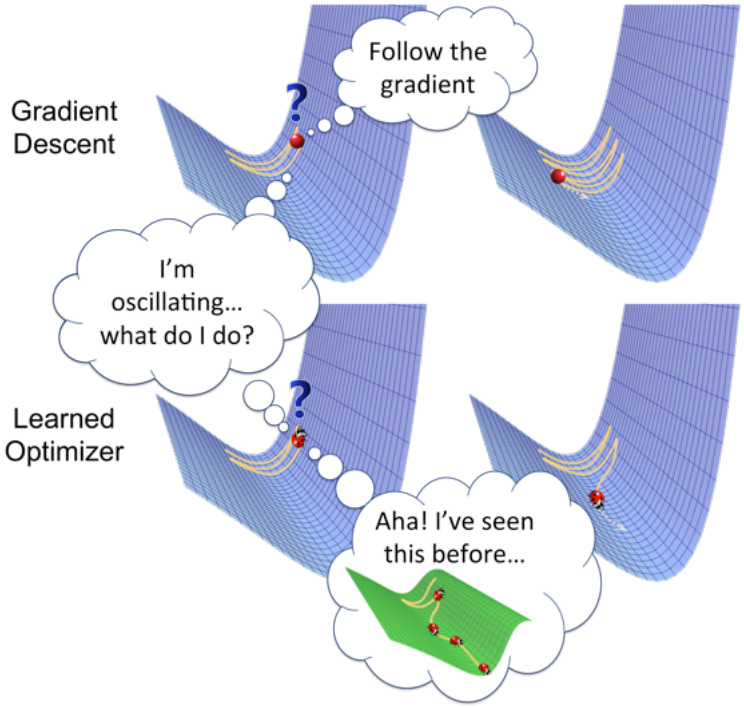


Figure 2.2: Conceptual illustration of the autonomous optimizer. The ladybug learned optimizer has knowledge of the behaviour of Gradient Descent on this problem-specific feature and therefore adapts its update strategy [141].

Aside from recurring neural networks, Wang et al. [147] developed a promising meta-learning approach for deep reinforcement learning (RL) systems. The system is learned a task by a reinforcement learning model but is tested on a completely different task. A proof-of-concept meta-learned model is developed and benchmarked against various navigational and machine learning problems. The results show that the optimizer learns the exploitable features in each new problem.

Despite the promising conclusions in these meta-learning studies, the excellent results of hand-crafted meta-heuristics should not be diminished. However, the performance of these classic meta-heuristics is limited by the 'No Free Lunch' theorem. By definition, the 'No Free Lunch' theorem could theoretically be beaten if the use of each meta-heuristic is regulated to only optimization problems which incorporate the algorithm's exploitable feature. Using a data-driven approach to handle the choice of meta-heuristic could be promising.

This thesis proposes an adaptive heuristic decision strategy that can alter the heuristic choice during black-box optimization. As the study of Cao et al., [146] population-based, as well as single solution optimizers, are implemented in the set of selected meta-heuristics. However, this set is not limited to popular gradient-based optimiz-

ers as in the study of Wichrowska et al. [145]. In addition, we introduce other popular derivative-free algorithms based on the low-level heuristic categorization of HyFlex, as discussed in section 2.3. The update step of the data-driven heuristic decision strategy will not be learned as in meta-learning studies [141–143], but carefully chosen among a set of hand-engineered meta-heuristics.

From the simple comparison of optimizers on benchmark problems in figure 2.1, it can be seen that different optimizers shine during a more global or local stage of optimization. Therefore we assess and, if necessary, adapt the heuristic choice during different convergence stages of the optimization process. These decisions will be made based on a data-driven framework of benchmark optimization runs of the various selected heuristics. By collecting problem-specific information of various optimization benchmark problems and coupling this information to the best-performing meta-heuristics, we could gain insight into the domain of solvable problems for each heuristic. The prior information from generated optimization problems is stored in a database and aids us in solving new optimization problems. The heuristic decision for new optimization problems is dependent on the most successful heuristic from similar preceding cases. In this way, we develop a data-driven heuristic decision strategy and comply with the message of the 'No Free Lunch' theorem.

3

Bio-based composite optimization

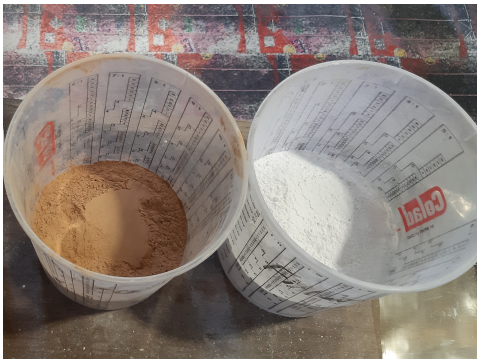
In this chapter, the development of a bio-based composite optimization model is described utilizing a Bayesian Optimization approach. In collaboration with NPSP B.V. a Python application is built that recommends bio-based bulk moulding compound (BMC) recipes. We start off with defining a bio-based composite and the production process of a bulk moulding compound, the hot press moulding and the mechanical testing of the bio-based composite plates. Subsequently, the input and output parameters of the optimization model are described using a design of experiments. After that, the optimization model is illustrated. The generated recipes and the resulting mechanical properties are described in detail in the results section. Finally, we discuss several recommendations and conclude this chapter.

3.1. Methodology

A bio-based composite recipe consists of the following materials: a natural fibre, a natural filling material, the co-filling material calcite ($CaCO_3$), a polyester resin matrix, the cross-linking initiator Trigonox C and zinc stearate as a release agent.

The natural fibre is dried for at least 2 hours in an oven at 85-100 °C before starting the mixing process. The natural filling materials are ground to a powder with a particle size between 50 and 500 microns. The natural filling material is supplemented with calcite. Both calcite and the natural filler are also dried for at least 2 hours before processing. The matrix material consists of unsaturated polyester resin in styrene. To initiate the thermosetting polymerization process, the initiator Trigonox C is added. Zinc stearate is used as a release agent; it uses its non-sticking properties to avoid the bulk moulding compound to stick to the hot press mould surface. Figure 3.1 shows the natural fibre, the natural filler and the co-filler material.

(a) Recell fibre.



(b) Peach stone (left) and calcite (right).

Figure 3.1: The natural fibre Recell and two different filler materials. The components are added to the industrial mixer in batches to ensure proper mixing.

The production process of bio-based composite plates through hot press moulding is described below. First, the polyester resin, zinc stearate and Trigonox C initiator are mixed separately in a small container, as shown in figure 3.2. The reason for this is to make sure that the initiator and release agent are well mixed with the polyester before adding the natural materials. The resulting mixture is defined as the premix. After the premix has been homogeneously mixed, it is deposited in a large industrial mixer, shown in figure 3.3.

Figure 3.2: The premix.

Figure 3.3: Industrial mixer.

Subsequently, the natural filling material and the calcite are added to the mixer in three equal parts. In between, everything is mixed until homogeneous paste forms. Next, the dried natural fibre is added, which is also split into three batches. The whole fibre is mixed for shorter times: the first batch for 2.5 minutes, the second batch for 3.5 minutes and last batch for 4.5 minutes. Extra care is taken to avoid the structure of the fibre to be affected by the mixing process. The entire mixture is stored in a vacuum-sealed bag so that the styrene does not evaporate. In total, around 2.5 kg of bulk moulding compound is manufactured for each recipe. Figure 3.4 shows the resulting bulk moulding compound.

Figure 3.4: Bio-based composite bulk moulding compound (BMC) from Recell fibre and peach stone/calcite fillers.

The second part of the production process is pressing the bulk moulding compounds into plates using hot press moulding. Under pressure of 100 bar and with an upper and lower temperature of 152 °C and 148 °C respectively, the plates are pressed for 5 minutes each. 500g of the bulk moulding compound is used for each plate, hence a maximum of 5 plates can be produced for one recipe. The resulting plates have an area of 35x25 cm^2 and are depicted in figure 3.5.

(a) Top view. (b) Side view.

Figure 3.5: Bio-based composite plate. The plates have dimensions of approximately 350x250x7 mm

Finally, the mechanical properties of the BMC specimen are tested. For this study, only the flexural strength and modulus were investigated with a three-point flexural bending test. The 'ISO 178 (2010) Plastics - Determination of Flexural Properties (Method B)' standard is used [148]. The test is performed on an Instron 5969 50kN Dual Column universal testing system. The plates are cut by a waterjet cutter to smaller test specimen with a size of 4x13x81 mm. For each composite, five three-point bending tests are performed.

3.2. Design of experiments

A bio-based composite recipe is defined by the weight ratios of these components and choice of natural materials. By varying the proportions and types of natural

materials, different bio-based composite recipes are created. The above-described process is repeated with different recipes until a bio-based composite has been manufactured with the desired properties. As mentioned before, the properties of biomaterials are unpredictable. This makes the mechanical properties for different ratios of materials in a bulk moulding compound challenging to foresee.

The current conventional way of researching is as follows: a standard recipe is used, and the weight ratio of one of the components is adjusted. Subsequently, a series of recipes are created by adjusting the weight ratio in equilateral steps. Then, another parameter is adjusted, and the process repeats. In terms of algorithmic optimization, we can speak of a quasi-random search process. Based on mechanical properties evaluation of these initial recipes for different components, new recipes are created. The recipe creation is therefore not blind-eyed, however adapting the recipes by hand is rather complicated.

As mentioned in the literature review, heuristics use the information acquired from past iterations to converge towards an optimum. By substituting the quasi-random search strategy with a heuristic, we strive to improve the solution quality with respect to the number of iterations. According to the 'No Free Lunch' theorem, choosing an efficient algorithm is not arbitrary. However, even a minimal improvement in the search process can yield significant time savings as the entire production process from mixing bulk moulding compound to testing the mechanical properties can take several days.

Input and output parameters and box-constraints need to be defined to adapt the bio-based composite recipes to an optimization problem.

Input parameters

It is essential to indicate the dimensionality and the search space boundaries to express the bio-based composite recipes as an optimization problem. The search space increases exponentially with each added dimension. Because of the extremely expensive sampling process, optimizing for a huge search space will not be beneficial. Therefore, several simplifications are considered in the bio-based composite recipes' composition to reduce the variable parameters.

We consider a three-dimensional continuous search space.

- The first parameter is the weight ratio of fibre x_{fibre}. It is decided that the composite consists of only one type of natural fibre.
- The second parameter is the weight ratio or natural filler x_{filler} concerning the total amount of filling material, which consists of natural filler and the co-filler calcite. Again, a single natural filler system is considered.
- The third parameter is the weight ratio of 'dry material' x_{dry} concerning the entire composite. The weight ratio of dry material is the natural fibre's weight ratio, filling material and co-filler calcite together. The wet material is the premix and is also fixed with this parameter.

The composition of the premix is constant as we do not want to increase the dimensionality of the model any further. Out of the premix's total weight, 92.4% is considered to be Polyester, 1.4% Trigonox C and 6.2% zinc stearate. Figure 3.6 illustrates the weight ratios and the three continuous parameters x_{fibre}, x_{filler} and x_{dry}.

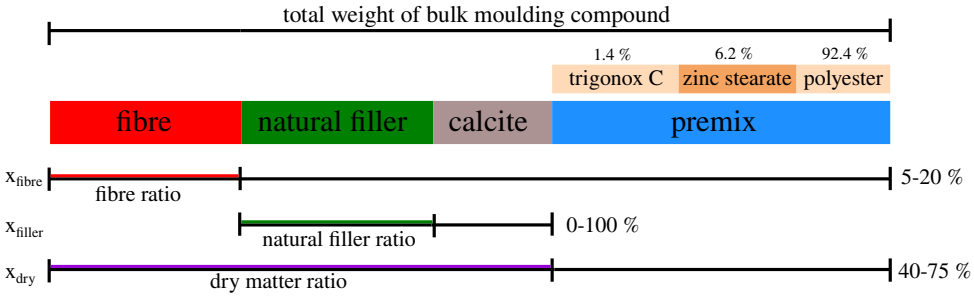


Figure 3.6: Illustration of the three continuous parameters of the bio-based composite model. The composition of the components in the premix is constant.

Designating search-space constraints for the three continuous parameters is a difficult job and is primarily based on experimentalists' empirical knowledge. The purpose of the box-constraints is to exclude any bio-based composite recipe from the search space that will result in untestable pressed plates.

In the optimization model, the search-space boundaries of each input parameter can be adjusted per optimization. The recommended values are provided below and are displayed in figure 3.6.

- The natural fibre mostly reinforces the material, and with too small an amount of fibre, the composite will not be strong enough. However, if the fibre percentage is too high, there is poor interaction with the matrix material, and the composite becomes too brittle. In addition, fibres are usually more expensive than filler material, so there is a compromise between price and properties. As a result, the fibre ratio will vary from 0.05 to 0.2.
- The weight ratio of natural filler to calcite will vary from 0 (no natural filler at all) to 1 (utterly natural filler). Calcite is a strong and inexpensive filling material, but often heavier than natural alternatives. For lightweight applications, the replacement of calcite will be desirable.
- The amount of dry material is highly dependent on the type of fibre used. If a dry fibre such as Recell[1] is used, the resulting bulk moulding compound may be too dry to press a homogeneous sheet. With flax fibre, the dry material could be increased more before the bulk moulding compound becomes too

[1]Recell is a recyclable cellulosic material obtained from production waste-streams [149].

dry. The default boundaries for the dry material parameter are set between 0.4 and 0.75.

All bio-based composite recipes are saved in an Excel file as input for the optimization model. The Excel file consists of input parameter columns describing the compounds and output columns consisting of their respective mechanical responses. The optimization model reads the type of natural fibre and filler as well as the fibre, filler and dry matter ratio for each row. Table 3.1 shows the required format of the Excel data file.

name	**type fibre**	**type filler**	**fibre ratio** (x_{fibre})	**filler ratio** (x_{filler})	**dry ratio** (x_{dry})
FlaxOli50	Flax	Olive stone	0.0995	0.1542	0.6532
ReedPeach50	Reed	Peach stone	0.0732	0.1375	0.5592
...	...	...	...	...	...

Table 3.1: Format of input columns with example data.

Output

After manufacturing and mechanical testing, the mechanical responses are recorded in the same Excel data file. Each mechanical response that has been tested is assigned to a different column. All the numerical values in one column must have the same unit of measure.

If for some reason the produced plate for a given recipe is in an unsatisfactory state to be tested, the column 'testable?' can be set to 'no'. By doing this, the optimization model will know that that combination of input parameters is not desired. Table 3.2 shows an example of the Excel file's output columns. Note that the output columns do not exhibit a strict format. Any property can be added as optimization output as long as the parameter is numerical and continuous.

testable?	**Density (kg/m^3)**	**Impact toughness (kJ/m^2)**	**Tensile strength (MPa)**	**Flexural strength (MPa)**	**Flexural modulus (MPa)**
yes	1.7187	2.1	9.7	28.8	4770
yes	1.4654	2.3	9.5	32.1	5647
...	...	...	...	...	...

Table 3.2: Format of output columns with example data.

Single-objective penalty score

As mentioned earlier, the model operates for single-object optimization. However, materials are often assessed on multiple mechanical properties. A weighted objective score is constructed to reduce the multiple objectives to a single-objective optimization.

This weighted objective score is built up as follows. Consider $\vec{y}_i = (y_0, ..., y_d)_i$ to be a vector containing the d number of mechanical properties of some bio-based composite plate i. Only the λ number of recipes known in the database for that specific

combination of fibre and filler are considered. Each element of $\vec{y}_i$ is normalized for the minimum and maximum values $y_i^{\min}$ and $y_i^{\max}$, described in equation 3.1 and 3.2.

$$y_i^{\min} = \min(y_i^0, ..., y_i^\lambda) \tag{3.1}$$

$$y_i^{\max} = \max(y_i^0, ..., y_i^\lambda) \tag{3.2}$$

It is subsequently examined for each mechanical property, whether a minimum value or maximum value is desired. For example, the density of a plate is mostly desired to be minimal. However, heavy-load applications require the flexural strength of a composite to be maximized. The weighted objective score should be able to adapt to the application needs. This is accomplished by specifying the minimization or maximization for each mechanical output with the polarization vector $\vec{a} = (a_0, ..., a_d)$.

The resulting normalized properties are multiplied by a weight vector $\vec{\omega} = (\omega_0, ..., \omega_d)$. Output properties with a greater weight value will have more influence on the total score than lower weights. In this way, the user can emphasize certain design criteria.

All individual scores are added together and divided by the sum of weights to form a penalty score s_i. The purpose of the optimization model is to minimize the penalty score. Equation 3.3 shows the calculation of the weighted penalty score s_i for one composite plate output.

$$s_i = \frac{\sum_{j=0}^{d} \omega_j \left| a_j - \frac{y_i^j - y_i^{\min}}{y_i^{\max} - y_i^{\min}} \right|}{\sum_{j=0}^{d} \omega_j} \tag{3.3}$$

The number of output parameters d, the polarization vector $\vec{a}$ and weights $\vec{\omega}$ can be adjusted before the optimization process. This allows the optimization progress to fit the requirements of the application. More information on defining these parameters in the model can be found in appendix B.1.5.

When no mechanical responses can be recorded due to the acquired composite plate's poor physical state, the 'testable?' column can be set to 'no'. Consequently, a large value will replace the weighted objective calculation. In this way, the optimizer will avoid that particular combination of input parameters for future iterations.

The calculation of the penalty score also takes missing data into account. If no data on an output property is given for a particular composite, this mechanical property will not be included in the scoring procedure for that particular plate.

3.3. Optimization model

The optimization model is written in Python 3.6 and works as follows. Before starting the optimization model, an Excel data file is required with the input and output columns described in the previous section. It is also necessary to describe the weights and polarity of the output columns to calculate the single-objective penalty scores. After this, the user is asked to select a type of natural filler and fibre. Only the data from which the natural fibre and filler are equal to the specified terms are used. The single-objective penalty score is calculated from the relevant data using the weights and the polarity of the output columns.

Now the Bayesian Optimization process starts. A Gaussian Process surrogate model is constructed with an RBF kernel, including all input data and penalty scores. An 'Expected Improvement' acquisition function is then drawn up and optimized. With this, we find the recommended recipe for the next experiment. After manufacturing and testing the bio-composite, the input and output parameters are appended to the data file, and the process can be repeated. A flow-chart of the bio-based composite model is illustrated in figure 3.7. The optimization model is open-source and can be found on GitHub.

Batches of multiple recipes

With the production capacity at NPSP, it is more efficient to make several bulk-moulding compounds on the same day. Therefore, multiple recipes can be requested from the optimization model in succession. The prediction accuracy will suffer from creating these batches since the acquisition function is not updated correctly until the next batch of tests is conducted. However, data can be gathered quicker, which will result in more reliable predictions.

The user can specify how many recipes the model generates. If the number of recipes is more than one, the remaining recipes will be sampled using the parallelization 'constant liar' strategy [150]. The constant liar strategy will duplicate the surrogate model and record the first recommended recipe with a fake objective function value equal to the lowest objective value within its 95% confidence bounds. The updated duplicate model is examined again, and a second recommended recipe is attained by optimizing the acquisition function. This procedure is repeated until the required number of recipes is obtained.

The generated recipes can be converted from the three weight-ratio parameters x_{fibre}, x_{filler} and x_{dry} to an Excel file with absolute quantities. The total mass of bulk moulding compound to be made has to be specified. The Excel gives an overview of each compound's weight per recipe and can easily be printed and carried out by the researcher. After the BMC-plates have been manufactured and tested, the resulting mechanical properties can be appended to the database and new recipes can be requested. A complete description of all features and parameters of the application and a getting-started guide are available in appendix B.1.

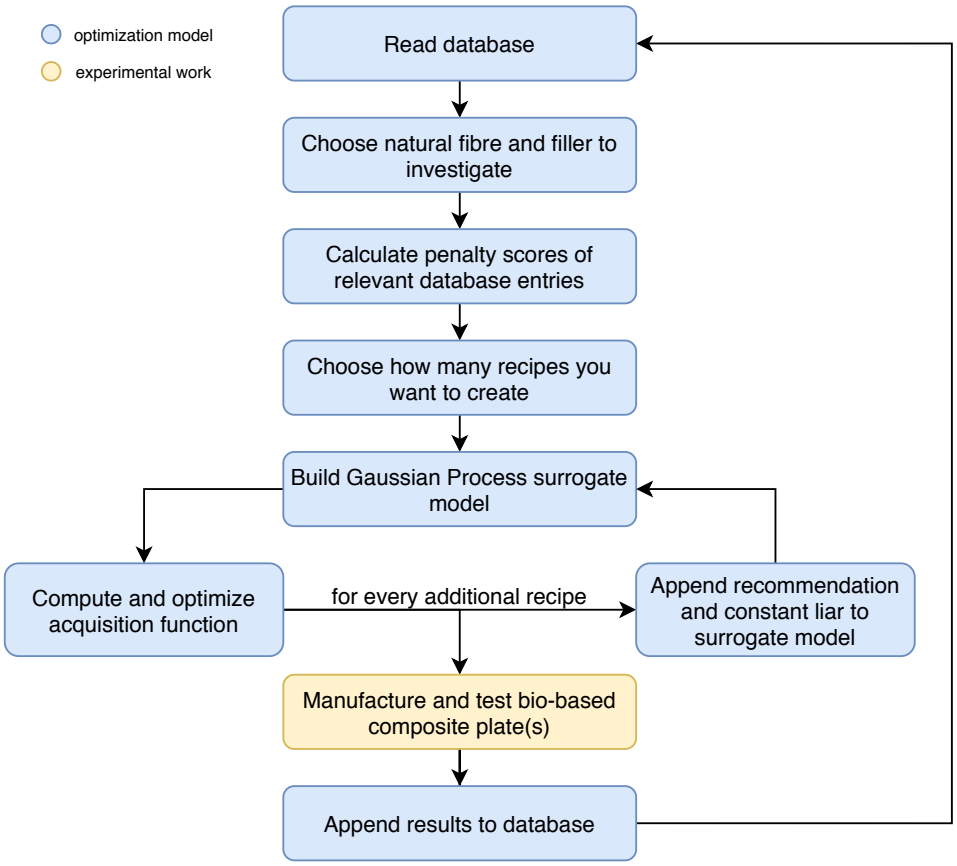


Figure 3.7: Diagram of the bio-based composite optimization model.

3.4. Results & Discussion

Due to the measures concerning the COVID-19 pandemic, the model has been put in practice to a limited extent. After some experiments where recipes were composed in the traditional quasi-random method, one iteration with three bulk moulding compound recipes was performed. Due to time limitations, these results should be regarded as a proof-of-concept and should inspire researchers to use this model in the near future. A Recell natural fibre [149] with a Peach stone natural filler material was chosen. This choice was based on the amount of information from preceding experiments with this particular combination of natural materials. The available data on Recell and Peach stone composites is given in table 3.3.

name	type fibre	type filler	fibre ratio	filler ratio	dry ratio	Density (kg/m^3)	Flexural strength (MPa)	Flexural modulus (MPa)
RecellCal100	Recell	Calcite	0.0925	0.0000	0.7141	2.003	42.45	4770
Sto2.2	Recell	Calcite	0.1111	0.0000	0.7272	1.9750	42.00	
RecellCalGly	Recell	Calcite	0.0889	0.0000	0.6866	1.9867	39.76	7516
RecellPea50	Recell	Peach stone	0.0925	0.5000	0.5671	1.4676		
RecellPea75	Recell	Peach stone	0.0925	0.7500	0.5674	1.3183		
Sto2.1	Recell	Calcite	0.0808	0.0000	0.7272	2.0556	44.30	
Sto2.15	Recell	Calcite	0.0918	0.0000	0.7245	2.0220	40.40	7080

Table 3.3: Mechanical properties of Recell and Peach stone bio-based composites.

Regarding the objective of the optimization, it was decided to make a composite that is very stiff, strong and light. In terms of polarization, weights and objectives vectors, the following values have been chosen. The density is minimized with a weight of 0.3. We want to maximize the flexural strength, which is why the value of $a = 1$ is chosen. This quantity also has a weight value of 0.3. The choice was made to give more importance to the composite's stiffness and therefore the flexural modulus and heavier weight of 1.0 has been given. This quantity is minimized to strive for a stiff composite. The objective values are given in table 3.4.

	Density	Flexural strength	Flexural modulus
weight value ω	0.3	0.3	1.0
polarity value a	0	1	0

Table 3.4: The output properties, weights vector $\vec{\omega}$ and polarity vector $\vec{a}$ for the Recell and Peach stone composites.

As the density is proportional to the compounds' weight ratio in the material, the density can also be deducted by the input parameters itself. Therefore, the density could also be considered a constraint on the input parameters. However, it is decided that the composite's density will be considered an objective to the material. Although a maximum density of a material is often a strict requirement to the application, it could also be imposed as an objective to strive for a lightweight material.

Based on the objective formulation in table 3.4 and the available data from table 3.3, the following penalty scores are calculated in table 3.5.

name	testable?	Density	Flexural strength	Flexural modulus	Total penalty score
RecellCal100	yes	0.1742194	0.0764042	0.0000000	0.2506236
Sto2.2	yes	0.4453433	0.2533040		0.6986473
RecellCalGly	yes	0.1653896	0.1875000	0.6250000	0.9778896
RecellPea50	yes	0.2024915			0.2024915
RecellPea75	yes	0.0000000			0.0000000
Sto2.1	yes	0.5000000	0.0000000		0.5000000
Sto2.15	yes	0.1789531	0.1610683	0.5257647	0.8657861

Table 3.5: Calculation of the penalty scores for the Recell-Peach stone composites.

Subsequently, three recipes have been generated with the intention that three bulk moulding compounds can be mixed in one working day. With a three-point bending test, the flexural strength and modulus are obtained. The complete stress-strain curves can be found in appendix B.2. The compositions and average mechanical properties of the generated recipes are shown in table 3.6.

name	type fibre	type filler	fibre ratio	filler ratio	dry ratio	Density (kg/m^3)	Flexural strength (MPa)	Flexural modulus (MPa)
RecPea24	Recell	Peach stone	0.2411	0.1258	0.6923	1.3178	40.2	4400
RecPea12	Recell	Peach stone	0.1236	0.2198	0.4653	1.4511	44.6	4870
RecPea09	Recell	Peach stone	0.0925	1.0000	0.5688	1.5897	32.4	5500

Table 3.6: Mechanical properties of the generated Recell and Peach stone bio-based composites.

Upon calculating the penalty scores after the three generated recipes have been added to the database, we can see that the new recipes made adequate composites. The results can be seen in table 3.7. Notice that the normalization boundaries have changed and thus the absolute values of the penalty scores.

name	testable?	Density	Flexural strength	Flexural modulus	Total penalty score
RecellCal100	yes	0.1742287	0.0330430	0.0742137	0.2814855
Sto2.2	yes	0.4453818	0.1065574		0.5519392
RecellCalGly	yes	0.1654052	0.0743852	0.6250000	0.8647904
RecellPea50	yes	0.2030535			0.2030535
RecellPea75	yes	0.0007047			**0.0007047**
Sto2.1	yes	0.5000000	0.0122951		0.5122951
Sto2.15	yes	0.1789591	0.0645492	0.5375481	0.7810564
RecPea24	yes	**0.0000000**	0.0676230	**0.0000000**	0.0676230
RecPea12	yes	0.0338848	**0.0000000**	0.0942715	0.1281563
RecPea09	yes	0.0690956	0.1875000	0.2206354	0.4772311

Table 3.7: Calculation of the penalty scores for the Recell-Peach stone composites after appending the three generated composite plate results.

The penalty scoring system does not penalize the absence of data. This can be seen because the best scoring recipe is still the *'RecellPea75'*, despite only the density being known for this composite.

3.5. Recommendations

The bio-based composite optimization model results described above are just the beginning of the use of algorithmic optimization for novel materials research. Several recommendations are mentioned in this chapter to give the project the right follow-up direction.

Collecting more data

First, more experimental data should be made available to the model. We only tested three new recipes, which is insufficient. To make a good recommendation, relevant data must be available for the specific natural fibre and filler combination to create a surrogate model. If this data is absent, the model will not perform better than a random search strategy. Besides, faulty manufactured BMC-plates that are not up for testing should also be added to the database. This helps the model to map out areas in which the model should refrain from searching. All this information will have to be stored centrally in the Excel data file. It is therefore essential that this data file is regularly updated with all experiments performed.

Controlling the optimization search-space

Manufacturing a well-built composite of natural materials is not trivial. As discussed earlier in the literature review, the interaction between the natural fibre, filler and matrix material is the bottleneck in making reinforced natural composites. By improving this interaction, a better load transfer is created, which improves the mechanical properties. Research has shown that the chemical modification of these natural materials could reduce their hydrophilic nature.

When the research results of a chemically modified fibre material are added to the optimization model, they are seen as a new type of natural fibre. Despite having the underlying properties of the unmodified natural fibre, no relationship with the modified fibre is established. In other words, the type of fibre or natural filler in the model is fixed as a discrete parameter, and there does not exist a relative continuous scale between different inputs. This choice was made to keep the model simple and decrease the search-space size but could leave out the influence of chemical modification on natural fibres and filler. Similarly, multi-fibre systems or different co-filler combinations induce continuous parameter traits to the discrete variables fibre and filler selection and are not supported by the optimization model.

In addition to chemical modification, adding additional compounds to the bio-based composite recipe is also common. Incorporating these so-called additives will help to increase the compatibility of the composite. Research is conducted in adding glycerol as a plasticizer to the resin, which increases the interfacial adhesion between the natural fibre and the matrix [151]. Besides that, the fibre ratio upper boundary could be raised if previously unsuitable recipes are becoming compatible with the introduction of glycerol. On the other hand, introducing additives to the input parameters will drastically increase the optimization search-space. Even if the glycerol input is added as a discrete parameter (either no glycerol or a constant

weight ratio of glycerol added in the recipe), the search-space volume is still doubled. The recurring consideration for the optimization model is, on the other hand, refining the bio-composite recipe by incorporating appropriate additives with the drawback of opening up the search-space too much to handle proper exploration of the search space. Therefore, the search-space should only be extended by introducing additive parameters if the research facility and time-management allow an increase in research production.

The above-described recommendations will increase the search space of the optimization model. Comprehensive characterization of the natural materials could aid to reduce the search space. Although the optimization model does not know the physical and mechanical traits of bio-based materials, the similarity between natural materials can be induced to combine data. For example, suppose the mechanical properties, density and particle size of two different natural fillers are alike. In that case, their recipes and penalty scores could be merged to give a rough estimation of a promising region within the search-space of either type of filler material. Extra care should be taken in combining the data-sets of different natural materials. However, when data about a specific new filler material is missing, the data from similar filler materials could guide the first iterations.

Implementing measurement uncertainty

Natural materials are more unpredictable than conventional composites. Even within the same batch of composites, it is possible that the bulk moulding compound is not entirely homogeneously mixed. The resulting bio-based composite plates could therefore exhibit distinctive mechanical responses. In practice, this can also be observed within the three-point flexural bending test data in appendix B.2. Some variation is present in the flexural strength and modulus. The relative standard deviation of the flexural strength is between 3.5% and 7.5% and between 2.0% and 2.5% for the flexural modulus. The model takes the average response of the five individual measurements and then the Gaussian process is assumed to be noiseless. The variance of the measurements is hence not included in the construction of the single-objective penalty scores. To consider the uncertainty of natural materials into the optimization model, we can also include this as a hyper-parameter and learn the noise level. When constructing a Gaussian Process surrogate model, this standard deviation can be used to fit the regression response surface. Implementing this so-called heteroscedastic Gaussian Process regression will realize confidence intervals on penalty scores and take uncertainty in mechanical properties into account.

The variability of mechanical properties for natural composites must be handled with care. That is why the recipes must be reproducible. The production of specific adequate recipes will have to be repeated to verify the mechanical properties and the associated uncertainty measurement.

4

Data-driven optimization

In the previous chapter, we have generated new bio-based composite recipes using Bayesian Optimization and contributed to the optimization of the mechanical properties of these novel materials. The idea of implementing data-scarce optimization in this field of research can be of great value.

The choice to use Bayesian Optimization for this specific case is based on the results and recommendations discussed in the literature review. Bayesian Optimization is competent for noisy problems with low dimensionality and where each function evaluations is expensive.

However, according to the 'No Free Lunch' theorem, we should base the algorithm choice on the problem's characteristics to be optimized. As seen in the literature review, the performance of optimization algorithms strongly depends on the type of problem-specific features. The specific response-surface features might be entirely distinctive between different natural materials. For example, Bayesian Optimization could work very well on optimizing the flax fibre and almond shell natural composites search-space. However, perhaps the response surface of Recell fibre and peach stone is better suited to solve with gradient-based optimization. The 'No Free Lunch' Theorem also indicates that different heuristics are only better than a random search for a select group of problems. Therefore, Bayesian Optimization would solve many problems well if they have low dimensionality, but this does not apply to all low-dimensional problems.

It is crucial to consider which algorithm is suitable for which problem to increase its success rate. The difficulty is that the characteristics of response surfaces are not yet associated with the competence of algorithms. Besides, these problem-specific features are not known before initiating the optimization, as black-box optimization is considered. However, we can learn from the performance and behaviour of preceding optimization instances. This information could enable the optimization pro-

cess to adapt to a more promising algorithm during the current optimization.

We develop a heuristic decision strategy based on data from previous optimization processes for this part of the research. The optimization process is split into an offline and online process. In the offline process, optimization results are acquired from a diverse generated problem set. We use the acquired data to identify similar problems in a database of previous optimization results during the online process. The currently used heuristic is then adjusted to the best performing heuristic from similar problems.

This chapter is built up as follows. First, we construct a set of optimization problems in section 4.1 that are being solved by a selection of meta-heuristics explained in section 4.2. In section 4.3 we compute a performance metric on the optimization results and determine a heuristic strategy by combining several algorithms in series in section 4.4. Next, we develop a heuristic metric that characterizes the problem-specific behaviour of a heuristic on a particular optimization problem in section 4.5.1. This information is stored in a database, which will be discussed in section 4.5.2 During the online optimization, we build this heuristic metric and compare it to the previously acquired signatures from the offline process. With classification, we will extract the best performing heuristic of the optimization problem by comparing the performance on similar problems in section 4.5.3. For the upcoming iterations, we will switch to the best performing heuristics and repeat the process. After optimization with the data-driven heuristic strategy, we will evaluate its performance in section 4.6 and give recommendations in section 4.7.

4.1. Optimization problems

Only box-constrained single-objective continuous problems will be considered for this research. As many benchmarking and real-world problems do not adhere to these criteria, it is decisive to limit the scope of this study. Because the current research trend lies more in the development of general optimizers, algorithms are not only used in this field of optimization problems. For instance, the gradient-based optimizers have been primarily used for training neural networks and machine learning applications [152]. Genetic algorithms are frequently used to solve unconstrained discrete spaces [153, 154]. However, both algorithm classes are also considered viable choices on continuous single-objective problems [155]. This means that particular conclusions about the algorithms' performance on this distinct set of problems are only related to continuous box-constrained single-objective optimization.

Problem-specific features

The 'No Free Lunch' theorem declares that algorithms exploit specific features of a problem in the search for a global optimum. If the exploitable feature of a problem associated with a specific algorithm is obsolete, this heuristic performance will be reduced. To study one heuristic performance, we need a collection of optimization problems that include this exploitable feature and problems where the feature is

not present. In the same way of comparing several heuristics, the collection of optimization problems requires a selection of benchmark functions that vary significantly in their problem-specific features. Only then it is meaningful to investigate the effectiveness of an algorithm objectively. The most used problem-specific characteristics are the presence or absence of local optima, the size of the global optimum area of attraction, the separability of the problem and the stochastic nature of the output.

Number of optima

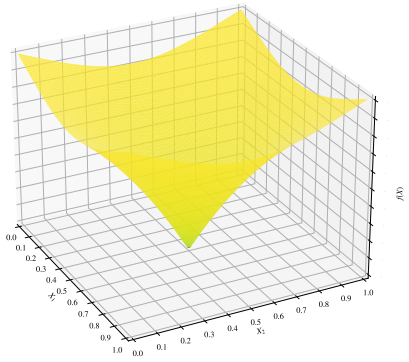
The number of local optima in an optimization problem is of great importance for an algorithm's effectiveness. If an optimization problem has only one global optimum, it is called a unimodal function. Gradient-based optimization methods score well on problems with a low number of optima. The direction with the steepest descent points in the direction of a local optimum. The global optimum is at once found when only one minimum is present in the search space. Figure 4.1b shows a successful optimization of the unimodal Ackley No. 2 function with the gradient-based optimizer Adam.

On the other hand, functions with multiple local optima can trap the heuristic into converging to a local optimum. Figure 4.1d shows the premature convergence to a local optimum when a gradient-based optimizer is subjected to the multi-modal Schwefel function. The distribution of the local optima is also relevant. Algorithms can use the periodicity of local optima to neglect them and focus on the underlying global optimum trend. As the dimensionality of the search space increases, its size grows exponentially. The number of optima with respect to the dimensionality is an important feature of a benchmark function.

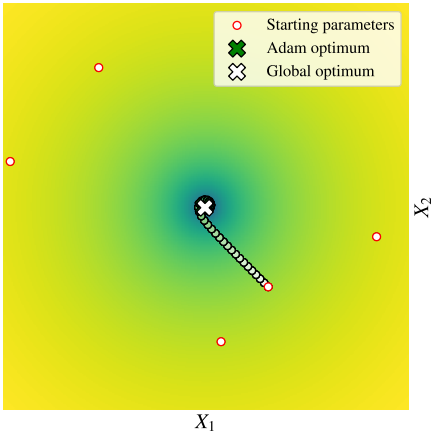
Area of attraction

The area of attraction is the area surrounding an optimum where the gradient is affected by the local minimum's depth. If a solution is located in the area of attraction, the minimum will be found by trailing the gradient in this confined space. A larger area of attraction increases the chances of solutions converging to this particular optimum. Heuristics that rely on a large area of attraction converge quickly to the minimum. Figure 4.2b shows the CMAES optimizer tackling a noisy bowl-function with a large area of attraction. The global optimum is easily found due to the information gained by the large area of attraction.

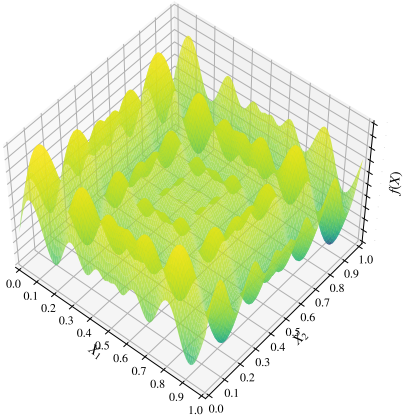
Optima with smaller areas of attraction are challenging to locate, as the surrounding search space gradient hardly gives away the optimum location. Heuristics that do not converge prematurely and explore the search space thoroughly have a higher chance to stumble across these small areas of attraction. Figure 4.2d illustrates how premature convergence of the same CMAES algorithm results in not finding the global optimum.



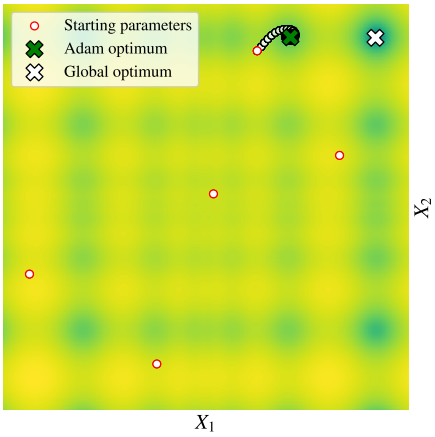
(a) Response surface of the Ackley No. 2 function.



(b) Objective function results of Adam optimizer.



(c) Response surface of the Schwefel function.



(d) Objective function results of Adam optimizer.

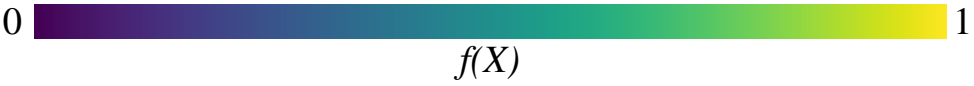
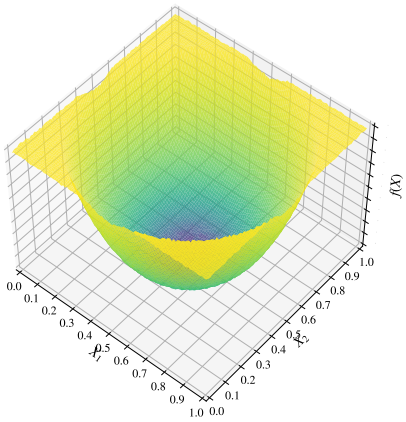
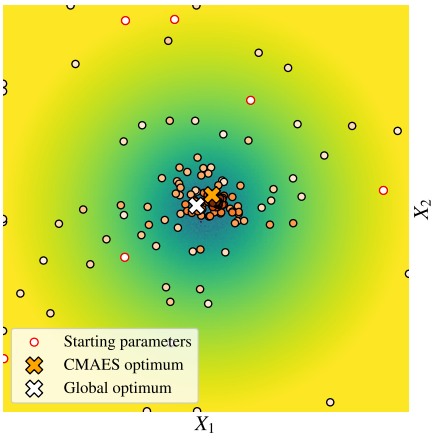


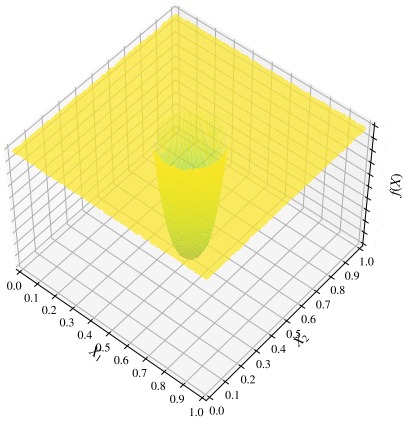
Figure 4.1: Optimization results of the Adam optimizer on the unimodal Ackley No. 2 function and the multimodal Schwefel problem. The red encircled markers indicate the 5 starting input points prior to the optimization process, while the black encircled markers with shaded green inner filling show the successive iterations of the optimization (from light green to dark green).



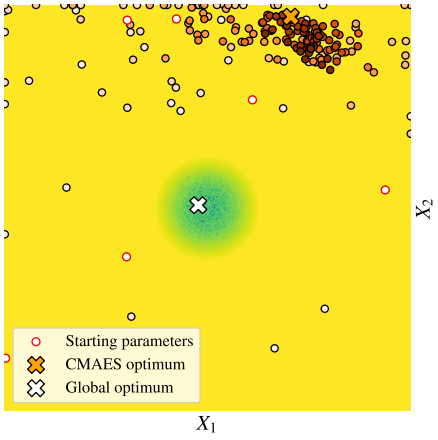
(a) Response surface of a Bowl-function with a large area of attraction.



(b) Objective function results of CMAES optimizer.



(c) Response surface of a Bowl-function with a small area of attraction.



(d) Objective function results of CMAES optimizer.

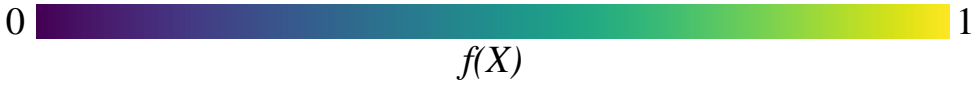
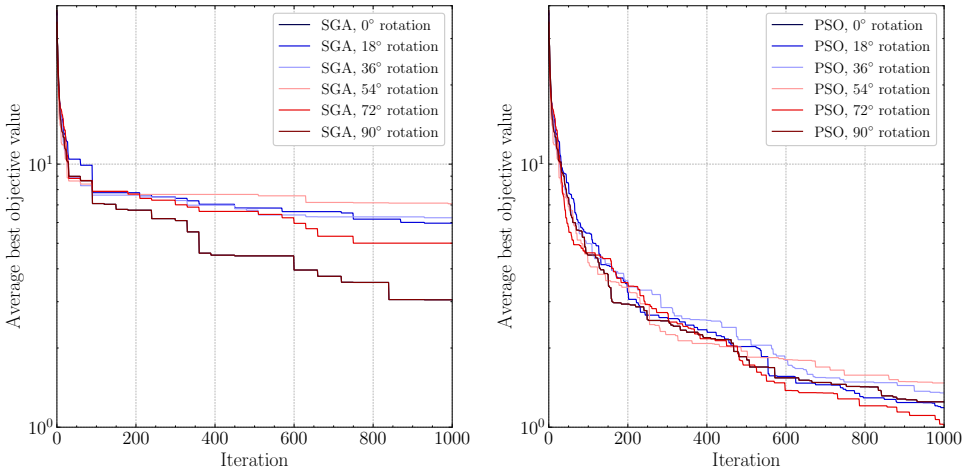


Figure 4.2: Optimization results of the CMAES optimizer on a two-dimensional noisy Bowl problem with both a large and small area of attraction. The red encircled markers indicate the 5 starting input points prior to the optimization process, while the black encircled markers with shaded orange inner filling show the successive iterations of the optimization (from light orange to dark orange).

Separability

Directional bias in optimization problems means that the optima are aligned in straight lines. If these lines correspond to the axis directions, a problem becomes separable. For separable functions, the input parameters can be optimized independently from each other. Meta-heuristics that cover mostly cross-over operations such as genetic algorithms are capable of exploiting separability. We can reduce the separability by rotating the solution space to stop the optima aligning with the axis direction. Figure 4.3 demonstrates this behaviour on the separable Rastrigin function. This function has four-fold symmetry. A simple genetic algorithm (SGA)[1] shows a decrease in performance in figure 4.3a when the solution space is rotated around its centre. Rotating the search space does not lead to a significant performance deviation for the Particle Swarm Optimizer in figure 4.3b, as this optimizer combines cross-over operations with other low-level heuristic techniques.



(a) Performance of SGA on the rotated Rastrigin function.

(b) Performance of PSO on the rotated Rastrigin function.

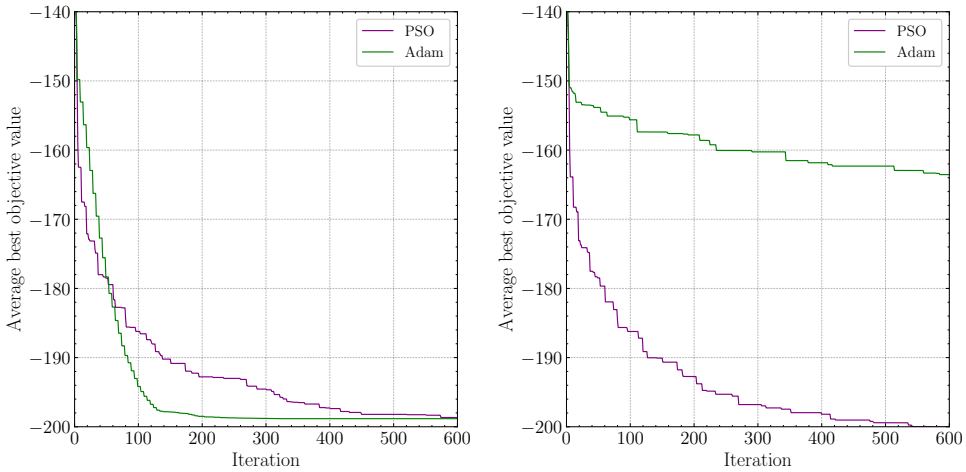
Figure 4.3: Performance of SGA and PSO on the separable Rastrigin function. The performance on $0°$ and $90°$ overlap, as both restore the axes alignment.

Stochasticity

Measurements are often subjected to physical disturbances or missing data. Real-world optimization problems are related to measurement uncertainties. This stochastic behaviour can be mimicked by adding Gaussian noise to the output signal. Stochastic optimization problems can cause problems to gradient-based optimization algorithms, as the local gradient is subjected to Gaussian noise. Other heuristics are capable of filtering the random fluctuations and focussing on the underlying objective function.

[1] The SGA algorithm is an implementation of the 'simple genetic algorithm' `pygmo.sga` with default hyper-parameters and a population size of 30.

This is demonstrated on the convex Ackley No. 2 benchmark function in figure 4.4. The gradient-based optimizer Adam shows superior performance over PSO in figure 4.4a on the noiseless function. However, when Gaussian noise is added, the performance of the gradient-based optimizer is severely lowered in figure 4.4b.



(a) Smooth Ackley No. 2.

(b) Noisy Ackley No. 2.

Figure 4.4: Performance of Adam and PSO on a smooth and noisy Ackley No. 2 function.

Set of optimization problems

We have to supply our problem set with analytical equations from a variety of the above mentioned problem-specific characteristics. The diversity of this set indicates how many different types of problems can be constructed using the analytical equation provided. The data-driven decision strategy will be trained by the behaviour of the algorithms on the generated problem set. If a new problem arises with similar problem-specifics as an entry in the training set, the data-driven decision strategy knows which heuristic exploits this feature. However, if a new problem of a completely different form arises, the data-driven heuristic decision strategy may make an unfounded decision, as data about this particular exploit is missing.

From the 'No Free Lunch' theorem, a random search would perform on average equally well as any other heuristic if all possible problems are considered. However, constructing such a generated problem set that oversees all optimization problems is impossible and can not be verified. However, we must strive to develop the most diverse problem set possible. Three sources of analytical equations have been implemented in the generated problem set. The first one refers to well-known optimization test functions that are commonly-used in benchmarking heuristics. Secondly, several parametrized family functions are adapted from the study of Rönkkönen et al. [156]. Lastly, a set of benchmark function from the CEC 2013

competition is implemented [157].

Well-known optimization benchmark functions

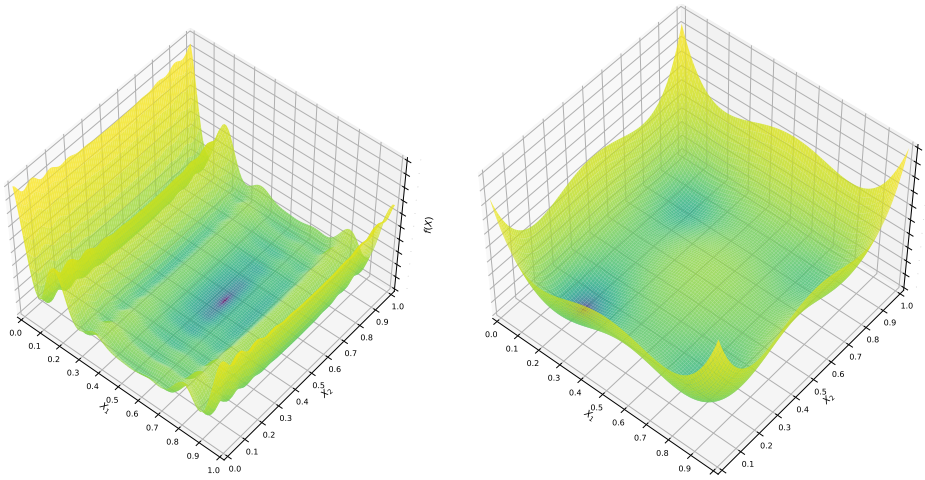
In the past decade, many optimization problems have been designed to challenge the quality of a heuristic. These optimization problems are designed for benchmarking optimization algorithms and are grouped according to their problem-specific features as seen in table 4.1. Several optimization benchmark frameworks and studies use a set from each of these categories of problems in different dimensions to outline an optimization problem's performance. The global optimum input parameters are analytically known as well as the box constraints.

	name	dimensionality
Many local minima	Ackley	any
	Levy	any
	Rastrigin	any
	Schwefel	any
Valley-shaped	Rosenbrock	any
	Bohachevsky	2
	Matyas	2
	Ackley N. 2	2
	Zakharov	any
	McCormick	2
Steep ridges/drops	Easom	2
	Schaffer F6	any
Other	Beale	2
	Branin-Hoo	2
	Styblinski-Tang	any

Table 4.1: Name, category and dimensionality of several well-known optimization test functions [158, 159].

These analytic optimization problems often isolate one specific characteristic of an optimization problem. For example, the Easom function has a deep global optimum, surrounded by a completely flat solution space. This problem is explicitly designed to test algorithms on finding a minimum while the area of attraction is low. Subsequently, the response surface of the Rastrigin function is highly multimodal, but the locations of the minima are regularly distributed. Heuristics that use the separability of an optimization problem will effortlessly find the global optimum.

A variety of these well-known functions is implemented in the problem set. Their analytical forms are found in table C.1. The two-dimensional response surface of the Levy and Styblinski-Tang test functions can be seen in figure 4.5.



(a) Levy function. (b) Styblinski-Tang function.

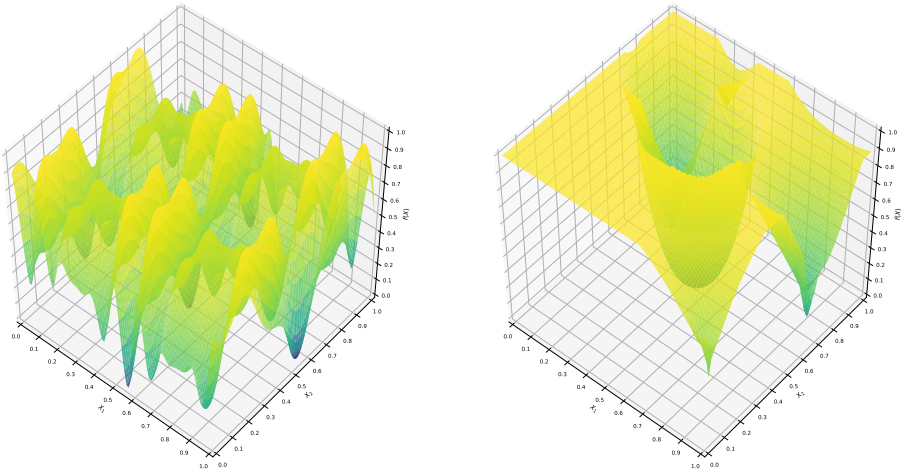
Figure 4.5: 2-dimensional response surfaces of the Levy and Styblinski-Tang well-known optimization test functions. The input parameters and response surface have been normalized.

Parametrized multimodal test functions

To enlarge the set of optimization problems, we introduce a set of parametrized test functions from the paper of Rönkkönen et al. [156]. Unlike the well-known optimization benchmark functions, the frequency and positions of local minima can be altered accordingly, as well as its area of attraction.

The study of Rönkkönen introduced a different family of functions for different problem-specific features. The cosine family function generates several local optima periodically by composing two cosine terms. The quadratic family combines several quadratic equations to create an irregular landscape of different slopes. Similarly, the plate family joins different linear equations. These landscapes test the ability of heuristics to find the global minimum with changing gradients. The hump family generates irregular landscapes with steep minima. The shape and area of attraction of these pitfalls as well as the positions are parametrized. Table C.2 in the appendix denotes the parametrized equations of the Rönkkönen tunable test functions.

The tunable test function framework offers more variability on the response surface landscape than the well-known optimization problems. Randomizing the parameters will repeatedly yield new optimization problems. The two-dimensional response surfaces for one realization of the cosine and steep family functions are visualized in figure 4.6.



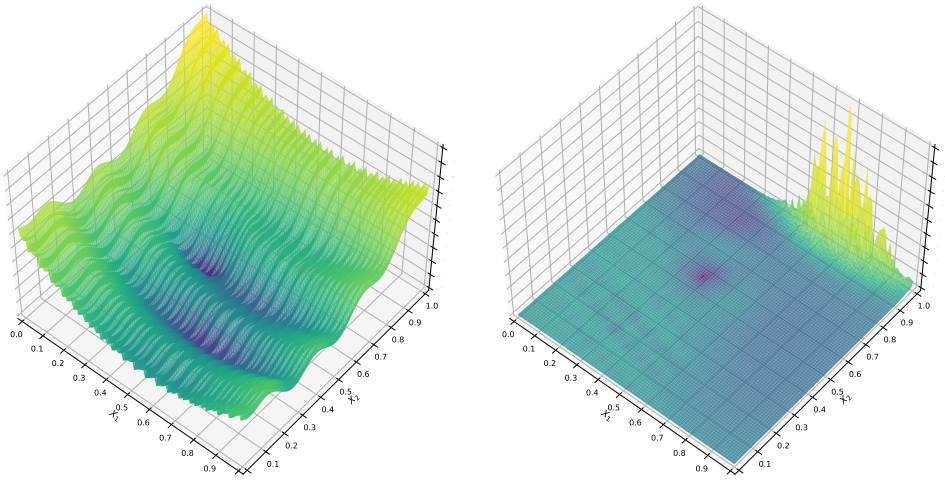
(a) Cosine family function. (b) Steep family function.

Figure 4.6: 2-dimensional response surfaces of the Rönkkönen cosine family function and the steep family function. The input parameters and response surface are normalized.

CEC 2013 competition benchmark functions

For the past years, the Congress on Evolutionary Computation (CEC) has composed a new set of optimization problems each year and hosts a competition to find the global optimum in the least amount of iterations [157]. These problems are hard to exploit and are considered challenging to modern heuristics. The CEC 2013 competition benchmark suites are designed to provide a suitable evaluation platform for testing and compare large-scale global optimization algorithms. The optimization problems are altered versions of well-known optimization problems. To resemble more real-world problems, non-linear transformations are applied to the functions to introduce some irregularity and break the symmetry of the fitness landscape.

All 28 optimization CEC 2013 competition problems are implemented in the problem set. More information on their specifics can be found in table C.4. Figure 4.7 shows the two-dimensional response surface of two of the competition functions



(a) Lunacek Bi_Rastrigin function. (b) Composition function 8.

Figure 4.7: 2-dimensional response surfaces of the Lunacek Bi_Rastrigin and composition function 8 from the CEC 2013 optimization competition problemset. The input parameters and response surface are normalized.

Post-analytical operations

Besides adding analytical equations to our generated problem set, we can also bring diversity to the problems in another way. By manipulating the mathematical equations, subtle changes can be induced to the respective response surfaces. The purpose of creating different instances of the same analytical benchmark functions is to break regularity and to isolate the true problem-specific characteristics. The three post-analytical operations performed on the analytical equations are offsetting the entire response surface with a constant vector, rotating the function by a certain angle and adding Gaussian noise.

Off-set

The global optimum of many benchmarking problems is often found in the centre of the box-constrained search space or at the origin. Heuristics based on averaging tend to have a bias towards the centre of the search space. We want to prevent algorithms from finding the optimum solely as a result of the centred position. Therefore, we add a constant off-set vector $\vec{o}$ to the input parameters $\vec{x}$. This vector has the same dimensionality as the input-vector and all of its elements are randomly selected within half of the search-space. This displaces the input parameters and thus, the global optimum. The box constraints remain constant and are not offset.

$$\vec{x}_{\text{off-set}} = \vec{x} + \vec{o} \tag{4.1}$$

Rotation

Rotating the search-space will alter the separability of the problem. Each test-function is rotated around the origin by a random angle θ between 0 and 360 degrees. For two-dimensional and three-dimensional problems, a rotation matrix $R(\theta)$ is used to perform a rotation in Euclidean space, as seen in equation 4.2 and 4.3. For problems in higher dimensions, a random orthogonal matrix is created, and a change of basis operation is performed on the input vector. Doing this operation will not only rotate the search space but will also stretch or contract it along the new axis.

$$\vec{x}_{\text{rotation}} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \cdot \vec{x} \tag{4.2}$$

$$\vec{x}_{\text{rotation}} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \vec{x} \tag{4.3}$$

Noise

Gaussian noise is added after calculating the objective function value to investigate the difference between stochastic and deterministic problems. The mean value μ of the noise is set to 0. The magnitude of the Gaussian noise is determined by the value of the standard deviation σ. This is set to 0 for a noiseless deterministic problem. No noise is added, and the problem remains deterministic. For a stochastic version of the problem, the standard deviation σ is a percentage s of the objective value, to avoid the situation where noise overshadows the fundamental equation or where noise would be insignificant. Three different levels of noise are implemented:

- low: random uniform value between 0.5% and 1%
- medium: random uniform value between 5% and 10%
- high: random uniform value between 10% and 20%

The absolute value of σ is determined by multiplying the noiseless objective value $f(\vec{x})$ with the noise percentage level s in equation 4.4:

$$f(\vec{x})_{\text{noise}} = f(\vec{x}) + \mathcal{N}(\mu, (f(\vec{x}) \cdot s)) \tag{4.4}$$

To conclude, the analytical objective function is altered according to equation 4.5:

$$f(\vec{x}) = f(R(\theta) \cdot \vec{x} + \vec{o}) + \mathcal{N}(0, (f(\vec{x}) \cdot s)) \tag{4.5}$$

A quasi-random process determines the post-analytical operational parameters $\vec{o}$, θ and s. Therefore, multiple calls to the same analytical equation with different

random seeds lead to different response surfaces. New analytical equations can be added to the program, as long as they are single-objective and box-constrained.

A set of optimization problems can be constructed by repeatedly selecting an analytical function and providing a random seed for its post-analytical operational parameters. Solving this complete set of generated problems with a set of algorithms will contribute to benchmarking the heuristics.

Two modified objective functions will act as examples for the upcoming sections to illustrate the benchmarking process. These are a two-dimensional smooth Styblinski-Tang function and a two-dimensional noisy Ackley function with the post-analytical hyper-parameters mentioned in table 4.3 and 4.2 respectively. The response-surfaces of these functions are displayed in figures 4.9 and 4.8.

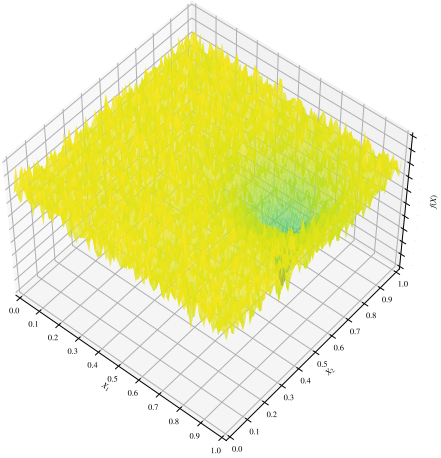


Figure 4.8: Response surface of the noisy Ackley function.

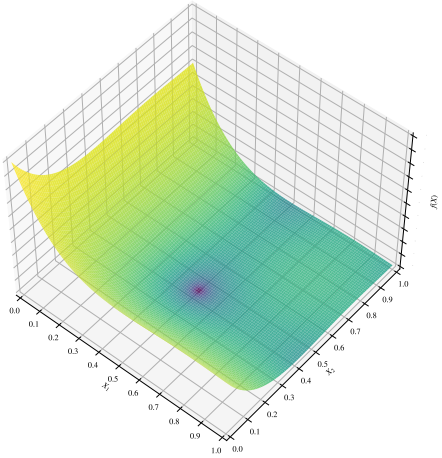


Figure 4.9: Response surface of the modified Styblinski-Tang function.

parameter	symbol	value
function	f	Ackley
dimensionality	d	2
off-set	$\vec{o}$	$[0.2401, 0.1524]$
rotation	θ	0°
noise level	s	5.61%

Table 4.2: Post-analytical hyper-parameters for the noisy Ackley optimization problem.

parameter	symbol	value
function	f	Styblinski-Tang
dimensionality	d	2
off-set	$\vec{o}$	$[0.2348, 0.2241]$
rotation	θ	0°
noise level	s	0.00%

Table 4.3: Post-analytical hyper-parameters for the smooth Styblinski-Tang optimization problem.

4.2. Algorithms

Now that we have generated a set of optimization problems, it is time to solve these problems with algorithms.

Before optimization, several initial input points are proposed at the start. The number of initial solutions is denoted as i and dependent on the dimensionality (d) of the problem via equation 4.6. The reason for using this expression is because it is desirable not to waste too many iterations on initial solutions and the number of parameter guesses for the `pygmo` implementation of CMAES must be greater than 5.

$$i = 2d + 1 \tag{4.6}$$

The initial guesses $\vec{x}_{0...i}$ and their associated objective values are assigned when the problem is generated. The initial input parameters are sampled via Latin hypercube sampling. Latin hypercube sampling is a statistical method to generate random samples in a multidimensional space. Each sample from a Latin-hypercube sampled distribution does not share elements of its axis-aligned hyperplane. This method ensures that the set of random initial solutions represents the real variability [160]. Figure 4.10 shows the difference between Latin Hypercube sampling and uniform random sampling. The points sampled by the Latin Hypercube principle are distributed more equally across the search-space.

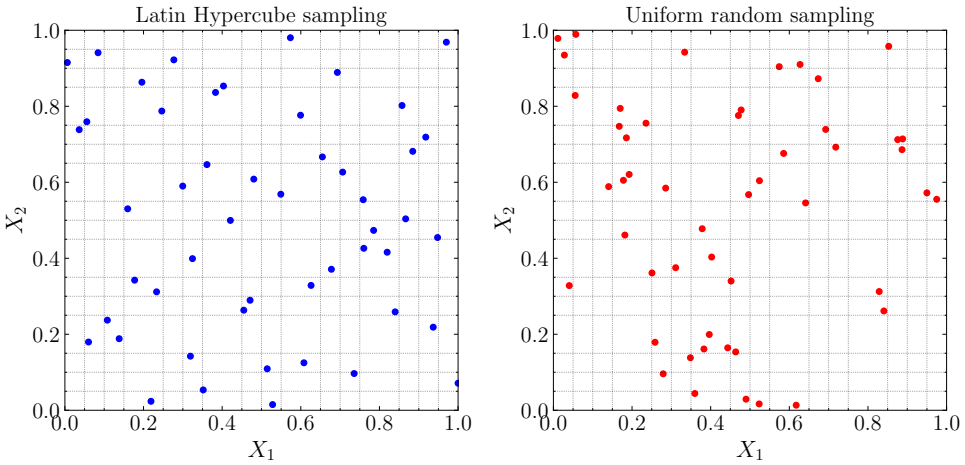


Figure 4.10: Illustration pointing out the space-filling sampling of Latin Hypercube sampling. 50 points are distributed according to Latin Hypercube (left) and uniform random sampling distribution (right).

The optimization will continue until a stopping criterion is reached. Three different stopping criteria are commonly used in benchmarking papers [161]:

- The optimization is stopped based on the convergence of the algorithm. The optimization can be halted if the new solution does not yield a better solution within some small threshold value ϵ.
- A fixed computational running time can be allocated to the algorithm. Extra care should be taken on running the different algorithms on similar programming languages and computer hardware.
- The number of function evaluations could be taken as a standard unit of time.

It was decided for this study that the stopping criterion of optimization is based on a fixed number of iterations $t_{\max}$, as this is a crucial criterion for data-scarce optimization. The convergence criterion introduces the convergence threshold parameter ϵ, potentially stopping a slow exploitation optimization process prematurely. Although the raw computational time is important to consider, the results will become platform-dependent.

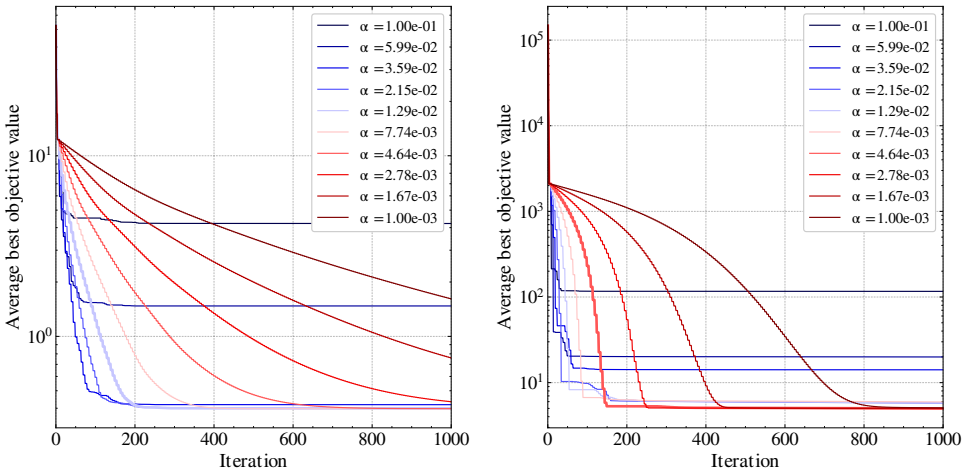
For single-solution algorithms, the best initial guess is taken as a starting point, and an update step will produce one new solution. For the population-based algorithms, each update step will modify all the initial guesses and requires several iterations. Hence, the number of update steps will be greater for single-solution algorithms than their population-based counterparts, as each update step will result in only one new solution.

Hyper-parameter optimization

Hyper-parameters can drastically alter the algorithm performance behaviour, as demonstrated in appendix A.2. Results from several hyper-parameter tuning frameworks show that using optimal hyper-parameters in contrast to hand-picked hyper-parameters dramatically improves the performance on black-box benchmark problems [137]. Figure 4.11 shows the Adam optimiser's performance on the two-dimensional smooth Branin and Rosenbrock function. By changing the learning-rate α, the performance is drastically altered. The Branin function's optimal performance is achieved with a learning-rate of $\alpha = 1.29 \cdot 10^{-2}$, whereas the best results for the Rosenbrock function are obtained with a learning-rate of $\alpha = 4.64 \cdot 10^{-3}$.

Examining the influence of hyper-parameters corresponds to executing optimization of the same algorithm with a different instantiation. As of the 'No Free Lunch' theorem, an optimal hyper-parameter setting is only beneficial for the problem for which it has been optimized. The performance gain on investing function evaluations and computational resources on hyper-parameter tuning is limited by that particular heuristic capabilities. In our view, optimizing for the algorithm instead of a fixed algorithm's hyper-parameters provides better performance because it brings more diversity to explore very different problems.

Figure 4.12 demonstrates this concept for the noisy Ackley function. Three of the selected meta-heuristics with various hyper-parameters are subjected to this optimization problem. The results show that the choice of meta-heuristic influences the solution quality much more than hyper-parameter optimization.



(a) Branin function (b) Rosenbrock function.

Figure 4.11: Performance of the Adam optimizer on the two-dimensional smooth Branin and Rosenbrock problem with different values for the learning-rate α.

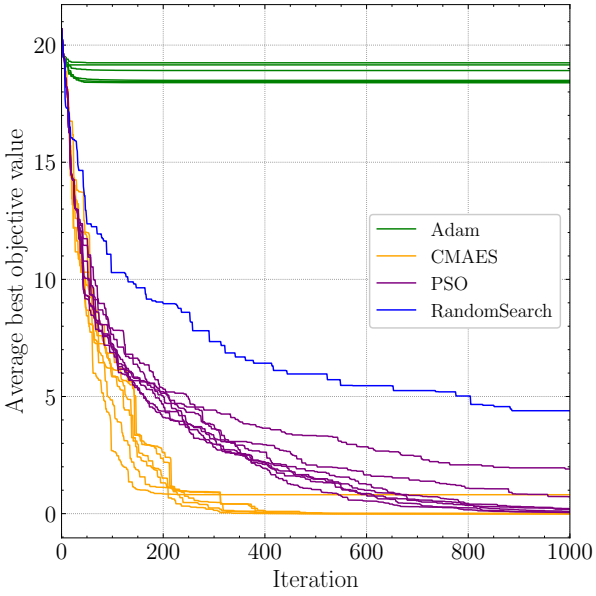


Figure 4.12: Optimization results for the noisy Ackley function for different algorithms with various hyper-parameters. The learning-rate α for Adam is logarithmically increased from 10^{-2} to 10^{-3}. The hyper-parameters ϕ_1 and ϕ_2 are altered in the same way as in figure A.3a. The population-size of CMAES is a step-wise increase from 6 to 21.

Although the hyper-parameters of algorithms drastically alter the heuristic performance behaviour, the default parameters for all the chosen heuristics are selected in this study. The reason for not optimizing the selected heuristics for their hyper-parameters is the optimization problem's data-scarce nature. If we are optimizing for the hyper-parameters, we are requesting function evaluations from the optimization problem itself. These function evaluations are added to the overall iterative cost of the algorithm. As we want to restrict the number of function evaluations, premature hyper-parameter optimization is not an option, and the hyper-parameters stay static for all generated problems. In addition, we want to assess the potential of using different algorithms without mixing another means of improvement and keep the dimensionality of the algorithm optimization problem manageable.

Combining both hyper-parameter tuning and algorithm tuning is being researched in the field of meta-learning [141, 142]. The 'learning to learn' idea is mostly directed in learning gradient-based optimizers for recurrent neural networks [143, 145, 146]. The reason is related to the fundamental obstacle of parametrizing the space of selected algorithms. If the group of usable algorithms is small, it is most likely that the set does not enclose the best performing algorithm. However, increasing the number of representing algorithms will result in an extensive search time that would scale exponentially. In this study, we tried to take a best-of-both-worlds by taking a small selection of heuristics from very distinct categories.

There has been an enormous growth in the number of meta-heuristics over the past few decades. A selection of meta-heuristics is briefly explained in the literature review, namely the population-based Covariance Matrix Adaptation Evolution Strategy (CMAES) and Particle Swarm Optimization (PSO), the gradient-based Adaptive Moment Estimate (Adam) and the novel surrogate model-based technique called Bayesian Optimization (BayesianOpt). The details of the used hyper-parameters and implemented variants can be found in appendix C.2. Table 4.4 summarizes the source-code library, the used variant, and the implemented meta-heuristics population size.

name	**library**	**variant**	**population size**
Covariance Matrix Adaptation Strategy	`pygmo.cmaes`	Classic interpretation in Hansen's review paper [123]	$4 + 3\ln(d)$
Generation Particle Swarm Optimization	`pygmo.pso_gen`	Generational, canonical variant	30
Adaptive Moment Estimate	self-coded	Standard interpretation from original paper [116]	$1 + 2d^2$
Bayesian Optimization	`GPyOpt`	RBF kernel, LCB acquisition function	1
Random Search	self-coded	Uniform random sampling	1

Table 4.4: Summary of the implemented variants of the selected algorithms, as documented in appendix C.2. For some implementations, the population size is dependent on the dimensionality d of the optimization problem.

The initial solutions have a significant impact on the outcome and performance of a heuristic. As mentioned before, the initial guesses $\vec{x}_{0...i}$ are equal between all algorithms. To truly understand an algorithm's behaviour on a specific problem, the heuristic performance should be studied over many different initial conditions. Therefore, the optimization process is repeatedly solved for different realizations to counteract the random seed bias.

The optimization problems are optimized for 10 different realization with a set amount of iterations $t_{\max} = 500$ for each of the selected meta-heuristics. The entire history of evaluations $\vec{x}$ and their respective objective values $f(\vec{x})$ are recorded, as well as the total computation time for each meta-heuristic.

4.3. Performance metrics

After the above-described algorithms have optimized a problem, the performance of the heuristics must be measured.

Constructing meaningful performance metrics is a crucial task for benchmarking various heuristics. Performance metrics show how optimizers behave when subjected to various problems and allow a meaningful comparison between similar optimizers. Carefully constructed performance metrics can have a positive effect on the credibility of said results. Nevertheless, biased metrics can mislead the reader and report unfounded conclusions [108].

Many different ways are used to show the performance of algorithms. Measuring the quality of the algorithmic output can be divided into two categories. If there is a known solution available, a fixed-target measure can be executed. With this method, the time required to find a particular target-value is evaluated. When the heuristic successfully reaches the desired target, the cost to achieve this accuracy can be used to measure the algorithm's quality on that test problem. However, if the algorithm does not reach the desired goal, it is considered unsuccessful. The other category involves optimization problems where the global optimum solution is not known beforehand. This is mostly applicable to real-world or randomly parametrized problems. The fixed-target measure can only be applied if the best-found solution from one of the benchmarking algorithms is used as fixed-target [161]. Figure 4.13 shows these performance metrics in a study where different instances of PSO and CMAES are compared [139].

Although these performance benchmarks are widely used, there are two fundamental shortcomings. First of all, the fixed-target metric only describes the results at the end of the optimization process and does not show the solution quality progression during the search process. Additionally, little account is taken of the statistical nature of initial conditions on the algorithms.

[2]Extra function evaluations are required per update step as the gradient is numerically estimated.

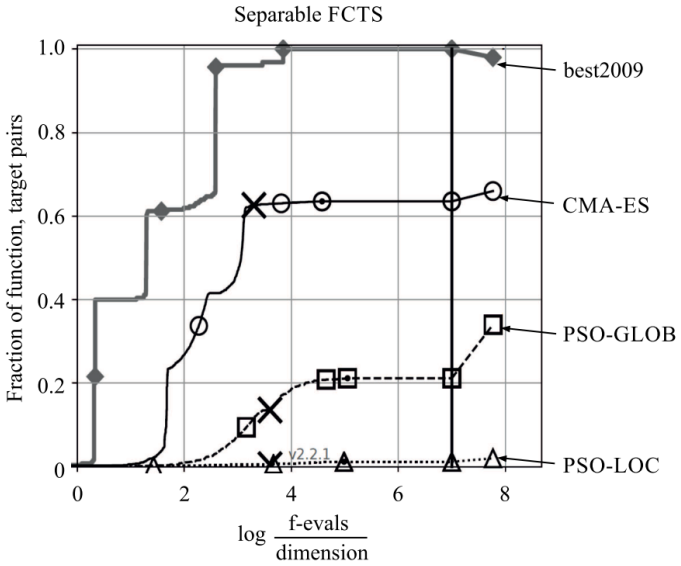


Figure 4.13: Illustration of the fraction of target pairs that are successfully reached against the number of function evaluations divided by the dimensionality. The triangle marker represents a PSO variant where only the local best vectors $\vec{p}$ of nearby particles are used. The diamond marker represents the best results of various solvers in the COCO 2009 benchmark competition. The circle and square marker represent the classical CMAES and PSO approaches respectively [139].

In this study, we propose an alternative performance metric that overcomes these shortcomings. During the optimization process, the following question could be asked about the optimization progression of an algorithm: how good is the currently best solution at a particular stage in the optimization process concerning other algorithms exhibiting the same initial conditions? In other words: how quickly does a meta-heuristic find a solution better than the initially given solutions, and how does this solution relate to the progression of other algorithms? To revoke the influence of the initial conditions, we take the average performance over the different realizations. This metric will be called the solution quality metric of an optimization process. The results from optimizing the modified Styblinski-Tang function will be used to construct the solution quality performance metric.

The mean margin of victory metric is consulted to assess the performance over a set of optimization problems [141]. This metric is explained after the solution quality metric.

Solution quality metric

To describe the solution-quality performance of the selected heuristic, we consult all the objective values $f(\vec{x})$ resulted from optimizing a particular problem. For each iteration t, we compute the value y_t with equation 4.7 by comparing the objective

value $f(\vec{x}_t)$ with the previous iteration $f(\vec{x}_{t-1})$ and appending $f(\vec{x}_t)$ to $\vec{y} = (y_0, ..., y_t)$ only if the current solution shows improvement.

$$y_t = \begin{cases} f(\vec{x}_0) & \text{if } t = 0 \\ \min(f(\vec{x}_t), f(\vec{x}_{t-1})) & \text{if } t > 0 \end{cases} \tag{4.7}$$

To compare the vector $\vec{y}$ between different realizations, we normalize this curve between the following boundaries:

- the upper-bound $y^{\max}$ will be the best solution from the initial guesses $y_{0,...,i}$ before any optimization is performed (equation 4.8). As the starting points for all the algorithms are equal, the upper bound will be the same for all $\vec{y}$.

$$y^{\max} = \min(y_{0,...,i}) \tag{4.8}$$

- the lower-bound $y^{\min}$ will be the best-found solution among all meta-heuristics after the optimization reaches its stopping criteria $t_{\max}$ (equation 4.9).

$$y^{\min} = \min(y_{t_{\max}}^{\text{CMAES}}, y_{t_{\max}}^{\text{PSO}}, y_{t_{\max}}^{\text{Adam}}, y_{t_{\max}}^{\text{BO}}, y_{t_{\max}}^{\text{RS}}) \tag{4.9}$$

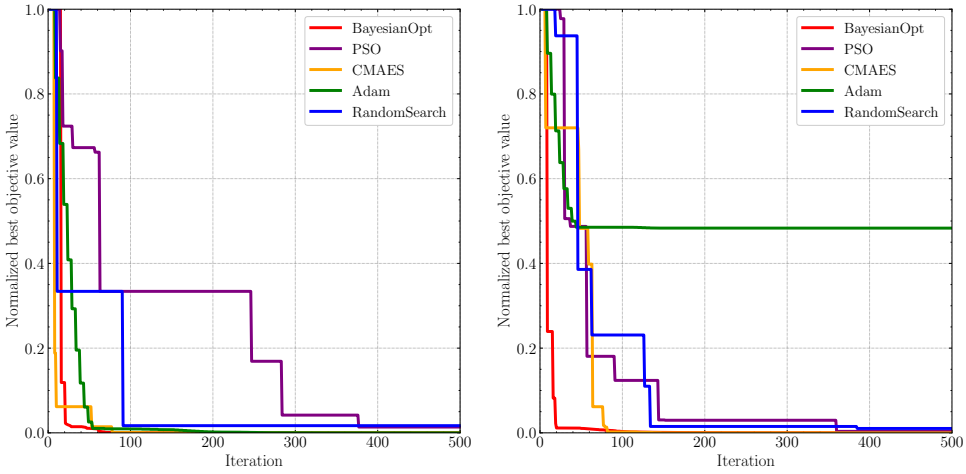
Utilizing equation 4.10 will construct the normalized best objective vector $\vec{y}^{\text{norm}}$:

$$\vec{y}^{\text{norm}} = \frac{\vec{y} - y^{\min}}{y^{\max} - y^{\min}} \tag{4.10}$$

Plotting $\vec{y}^{\text{norm}}$ against the iteration number t for every selected heuristic will result in figure 4.14:

It can be seen that on the first iteration, all heuristics express the performance value 1. Before the problem is being optimized, the initial solutions are all the same. As the algorithms are starting to explore the search-space, the performance values start to diverge. At the end of the optimization process, at least one algorithm will contain the best solution 0. For every iteration, an algorithm's solution quality can be perceived with respect to the best-found solution.

The optimization is repeated many times with different initial conditions, and the average performance is measured. Figure 4.15 denotes the average solution quality metric.



(a) Solution quality metric for one realization.

(b) Solution quality metric for another realization.

Figure 4.14: Solution quality metric for two different realizations of the modified Styblinski-Tang problem.

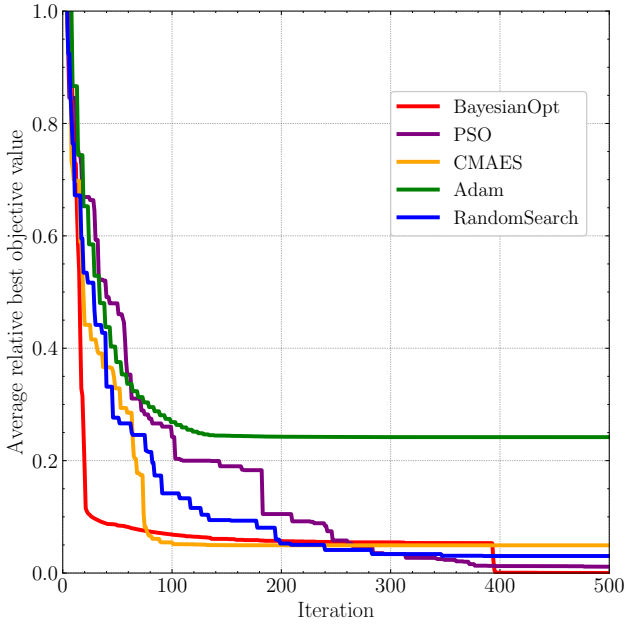


Figure 4.15: Average solution quality metric on the modified Styblinski-Tang problem, averaged over 10 realizations.

For the smooth Styblinski-Tang function, Bayesian Optimization will outperform its competitors during the first 100 iterations. However, CMAES will surpass Bayesian Optimization after around 80 iterations into the optimization process. After that, a better overall solution quality is found with Random Search between 240 and 320 iterations. The Particle Swarm optimizer is victorious within the region of 320 to 390 iterations. Lastly, Bayesian Optimization is the best-performing optimizer between 400 and 500 iterations.

The solution quality metric provides insight into which algorithm outperforms others on a particular problem. As discussed earlier, the 'No Free Lunch' theorem indicates that the best-performing algorithm differs for each problem. Besides, it can be seen that the best-performing algorithm is dependent on the number of iterations. Some algorithms will perform better on the global search space, while other algorithms will surpass their contestants in a more local search, closer to the converged solution.

Performance over a set of optimization problems

To get insight into the meta-heuristic performance over a set of optimization problems, we construct a second metric named the mean margin of victory [141]. The margin of victory $\vec{m}_i = (m_0, ..., m_t)_i$ of algorithm i for a single problem is described as the difference between the solution quality metric $\vec{y}_i^{\text{norm}}$ of the selected optimizer and the best-performing solution quality metric of the remaining optimizers. This calculation is denoted in equation 4.11.

$$\vec{m}_i = \min(\vec{y}_j^{\text{norm}}) - \vec{y}_i^{\text{norm}} \ , \ j \in A, i \neq j) \tag{4.11}$$

The parameter A denotes a list of all meta-heuristics subjected to the optimization problem. A positive margin of victory means that the meta-heuristic succeeds in finding better solutions than its competitors, averaged over many realizations. A negative value denotes that algorithm's failure, and its value gives insight into the gap between the best meta-heuristic. By taking the average margin of victory over a collection of optimization problems, the solution quality over multiple problems can be assessed.

Figure 4.16 denotes the mean margin of victory metric for the selected optimizers on a training set of 500 objective functions, optimized for 500 iterations with 10 different realizations.

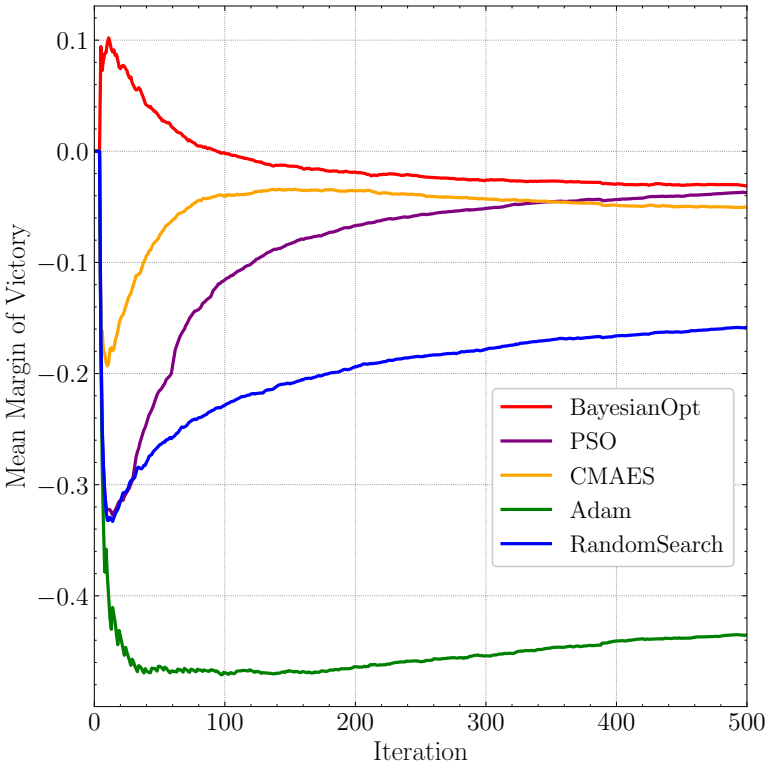


Figure 4.16: Margin of victory for the five selected algorithms for 500 different problems.

During the first 100 iterations, Bayesian Optimization shows a positive mean margin of victory. This means that on average Bayesian Optimization will be the best performing optimizer in this stage of optimization for the entire training set. As the bio-based composite case is modelled as a single-objective box-constrained optimization problem, the Bayesian Optimization algorithm is rationalized. However, even when the mean margin of victory metric averages a meta-heuristic performance over many different problem-specific features, it is still dependent on the distribution of problem-specifics in the training set. Therefore, conclusions on the mean margin of victory should be taken with care. A more elaborate discussion on the mean margin of victory results for the training set will be given in section 4.6.

4.4. Improving the heuristic decision strategy

Figure 4.15 has shown that the best-performing meta-heuristic per problem is dependent on the convergence stage during optimization. A combination of different heuristics on a problem could theoretically outperform the individual meta-

heuristics. This concept has been extensively studied in the field of hyper-heuristics [104, 105]. As discussed in the literature review, the underlying principle of the hyper-heuristic move acceptance algorithms is a black-box optimization itself [107]. Possible meta-heuristic update steps are applied to the current state of optimization, and the decision algorithm picks a move based on the objective function outcome. However, adaptive methods that change the used algorithm mid-search if certain conditions are satisfied are becoming more relevant. A study where the switching from Adam to Stochastic gradient descent (SGD) has been investigated on training deep neural networks has shown promising results [162].

Heuristic strategy

An explorative algorithm can start the optimization process, after which an exploitative algorithm takes over. The combination of heuristics and the associated amounts of iterations will be called the heuristic strategy. As one strategy will be beneficial for a single problem, it does not necessarily surpass the single algorithm strategy in another case. Therefore, the adapted heuristic strategy should be problem dependent.

Combining different algorithms in series brings up several difficulties such as switching between algorithms, changing population size and stating the number of algorithms in series.

Switching between algorithms

The data transfer between the switching of the heuristics must go well. This is reflected in the dynamic hyper-parameters of meta-heuristics and the previously acquired solutions. Some meta-heuristics exhibit dynamic hyper-parameters that control the convergence of the optimization i.e. step-size control parameter $\vec{p}_t^{\sigma}$ of CMAES or the 1st moment vector $\vec{m}_t$ of Adam. Whenever the optimization process is halted, these respective hyper-parameters are stored. If at a later stage the same heuristic is utilized again, the saved hyper-parameters are used instead.

Continuation of the stored dynamic hyper-parameters helps the optimization process to avoid an initiated explorative stage. However, suppose the optimization is at a later exploitative stage, and the heuristic decision strategy demands a change of algorithm. In that case, the hyper-parameters from an earlier explorative stage could be raised. Eventually, this could lead to a delay in convergence.

Altering the population size

Switching between meta-heuristics does also imply a possible alteration in population size. In order to retain the convergence state, a new starting population is selected. A probability density function is created from all the previously acquired solutions where the best solutions have the highest probability to be selected. From this probability density function, λ number of samples are drawn to match the new population size. In the case of Bayesian Optimization, a new surrogate model is computed with all the previous iterations.

This indicates the advantage of using Bayesian Optimization, as the heuristic switch-

ing is without loss of information. All other selected heuristics only store information from previously acquired iterations by their respective dynamic hyper-parameters. This means that only limited information is passed through to the next algorithm.

Predefined performance assessment stages

To reduce the number of decisions, we will make a heuristic strategy decision during predefined stages of the optimization process. We divide the maximum number of iterations into several equilateral parts named 'windows'. In the first window, we start iterating with one of the selected algorithms. Before we get into the second window, we stop iterating and either continue with the current meta-heuristic or switch. We will continue this process until we reach the maximum number of iterations.

A problem-specific heuristic strategy can be acquired in several ways:

- A heuristic strategy could be data-driven. By consulting the solution quality metric of a particular problem for each iteration window, we determine which meta-heuristic is the best performing heuristic. If multiple algorithms make a claim, then the algorithm with the best solutions for the majority in that segment is chosen. Figure 4.17 illustrates the concept with five decision stages for the modified Styblinski-Tang problem.
- A heuristic strategy can be obtained by randomly selecting a meta-heuristic for each iteration window.
- A heuristic strategy could be obtained by integer programming. The solution quality of the heuristic strategy could be the loss function for the meta-optimization. Over several iterations, new strategies are depicted by an algorithm. Due to the possibly huge demand of computational resources, this method will not be carried out in this study and will be further discussed as a recommendation in section 4.7.

After we have formulated a heuristic decision strategy, we can re-optimize the problem based on this strategy. Subsequently, the solution quality performance metric could be recalculated to show how the heuristic strategy compares to the single algorithm strategies in figure 4.18. From $t = 220$, the proposed heuristic strategy based on the solution quality metric outperforms the selected meta-heuristics on the modified Styblinski-Tang problem in figure 4.18a. Figure 4.19 shows the heuristic strategy based on the solution quality metric and 20 different, randomly constructed strategies.

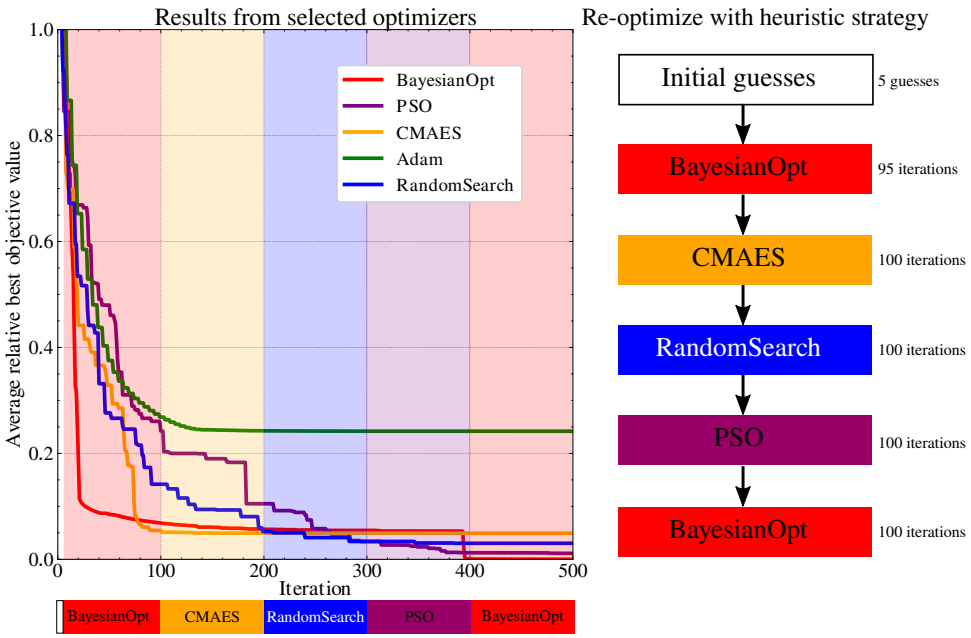
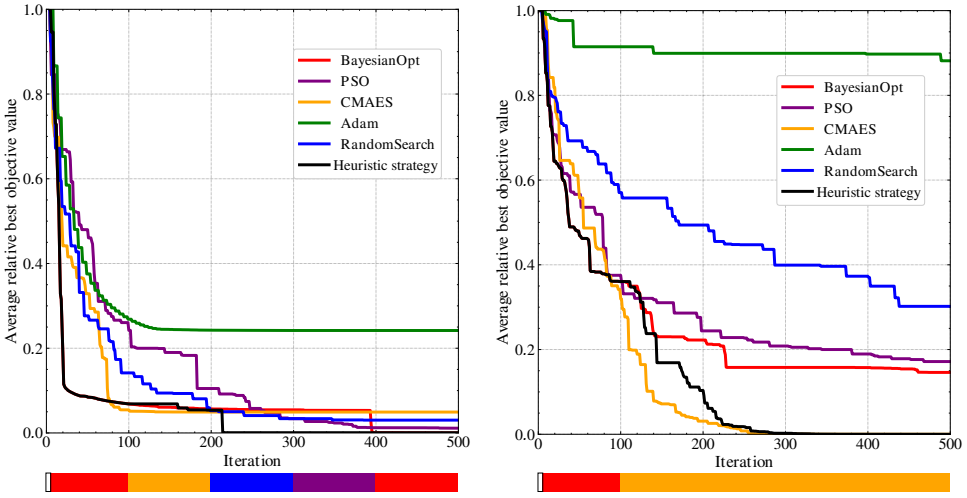


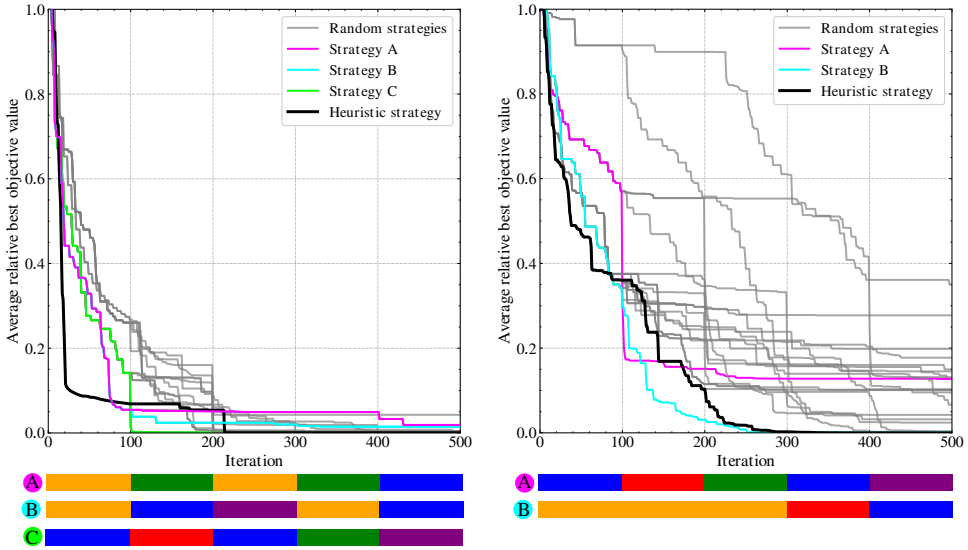
Figure 4.17: Constructing the heuristic strategy on the modified Styblinski-Tang problem for 5 predefined assessment stages.



(a) Smooth Styblinski-Tang problem.

(b) Noisy Ackley problem.

Figure 4.18: Performing the heuristic strategy on the smooth Styblinski-Tang and noisy Ackley problem for 5 predefined assessment stages. The coloured bar under the figures denote the used meta-heuristics for each iteration window.



(a) Smooth Styblinski-Tang problem.

(b) Noisy Ackley problem.

Figure 4.19: Comparison of the heuristic strategy based on the solution quality metric and 20 randomly decided heuristic strategies on the smooth Styblinski-Tang and noisy Ackley problem for 5 predefined assessment stages. The performance of the most successful random strategies (A, B and C) are displayed in fuchsia, cyan and lime green and their strategies are denoted in colour bars under the figure.

From the results of the most successful random strategies, three insights are acquired. First, implementing a random search window in the middle of the optimization process delays premature convergence. It results in better overall performance for strategy A and B on the Styblinski-Tang problem. Subsequently, combining random search and Bayesian Optimization in sequence will quickly improve both problems' solution quality. The explorative information from random search helps the Gaussian process to estimate the response-surface. Lastly, for the noisy Ackley problem, the CMAES optimizer only shows its effectiveness if applied over multiple windows, as seen with strategy B. Although the performance of the solution-quality based heuristic strategy is not necessarily the optimal strategy, it serves as an educated guess to improve the solution quality.

4.5. Data-driven heuristic decision strategy

Although the heuristic decision strategy based on the solution quality metric can find better solutions on average, comparing this method to the single algorithms is not fair. To determine the solution quality metric, the problem must first be solved several times through different heuristics. These preparatory function calls should

be included in the total number of iterations. Therefore, using a single promising heuristic for all new optimization problems may be more efficient in terms of function calls, even if this heuristic will not appear to be best on every single optimization problem. The same could be said about hyper-parameter optimization. When the optimal hyper-parameters for our specific problem are finally found, we have already been deep into the optimization process. It is not always possible to evaluate the objective function so often for data-scarce problems.

To avoid the excessive iterative process of finding an optimal heuristic strategy, we have to develop a strategy before starting the optimization. This is problematic because the heuristic strategy is problem-dependent. Without the problem-specific information of the optimization problem, coming up with a heuristic strategy is arbitrary. However, the performance information from other optimization problems similar to the current problem could be of value. By continually optimizing different generated problems discussed in section 4.1, we couple optimization landscapes to heuristic strategies. By picking the heuristic strategy of a previously optimized case, we by-pass the need for acquiring additional performance metrics of single heuristic strategies.

The data-driven heuristic decision strategy splits the optimization into two separate parts: an offline and online process.

- During the offline process, many different generated optimization problems are optimized, and suitable heuristic strategies are built. For each meta-heuristic on every generated problem, the algorithmic response is compressed into a unique heuristic identifier. Creating this unique identifier named the 'heuristic signature' is discussed in detail in section 4.5.1. The heuristic signatures, as well as the heuristic strategies, are stored in a database.
- The online process operates on unseen optimization problems. The total number of iterations is again divided into several iteration windows. First, we start iterating with a manually chosen algorithm. After the first window of iterations, we stop the optimization process and construct the same heuristic signature metric mentioned in the offline process. Subsequently, the database is consulted, and the problem is classified according to the most similar signature metrics in the database using a classification algorithm. The heuristic strategy associated with this label is then executed for the upcoming window of iterations. The classification is more thoroughly explained in section 4.5.3. This is repeated until we reach the stopping criteria.

An overview of both processes is illustrated in figure 4.20. In the upcoming sections, all the data-driven heuristic decision strategy features will be discussed in greater detail.

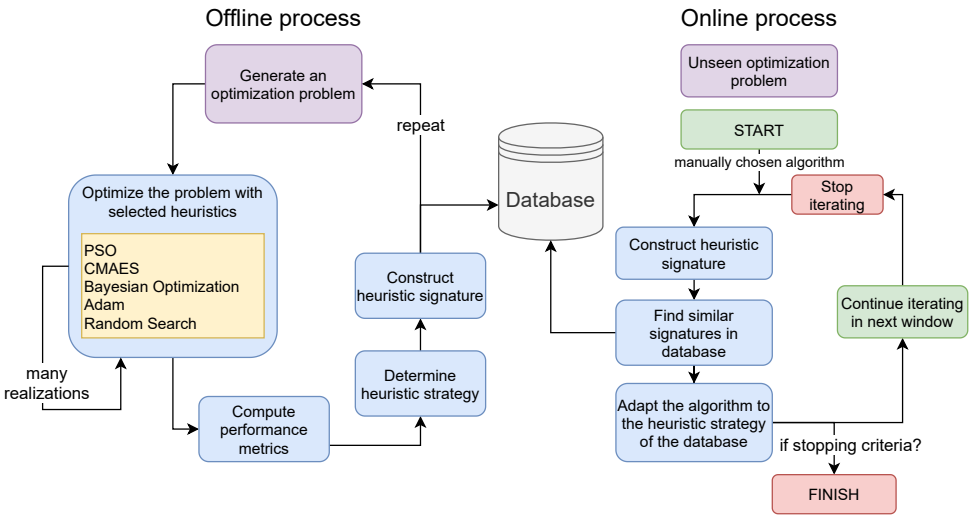


Figure 4.20: Flowchart of the data-driven heuristic decision strategy. On the left, the offline process of acquiring new database entries is illustrated. On the right, the online process is displayed.

4.5.1. Heuristic signature

It is necessary to identify a "fingerprint" of an algorithm's behaviour on an optimization problem to recognize an optimization problem from a database with previous optimization results. Section 4.1 mentioned that heuristics exploit problem-specific features. If a problem may or may not contain the correct problem-specific features for a specific algorithm, this can be seen in the algorithm's response. This response will be captured as the unique identifier metric.

Several requirements for the metric can be established:

- This "fingerprint" should show the influence of the algorithmic iterations only on the problem-specific features. Therefore, it is important that the initial conditions and any translations of the search space do not significantly affect the signature, as these operations do not change the problem-specific features.
- The heuristic signature metric must be constructible during the offline process and the online process so that the signature can be compared on equal grounds.
- The absolute values of the metric should be in the same order of magnitude so that the meta-heuristics performance over time is independent of the absolute value of the acquired problem's objective values.

In this section, we develop such a heuristic identifier metric for both the offline and the online process.

Offline process

When acquiring optimization results data in the offline process, the heuristic signature is set up as follows. First, all the objective values $f(\vec{x})$ are retrieved from all realisations. Only the objective values retrieved as a result of the meta-heuristic update steps are considered. The initial parameters can be considered a result of a random process, and therefore do not contain the unique response of the selected meta-heuristic. Because optimizing with different initial conditions is considered a stochastic process, a noisy Gaussian process regression model with an RBF kernel is used to fit the objective values. Only the average objective values from all realization are used as the training points for the Gaussian process fit to reduce the computational resources required. Figure 4.21 illustrates the process of constructing the heuristic signature for the results of the CMAES optimizer on the noisy Ackley problem. The predicted mean value of the Gaussian process is stored as the heuristic signature.

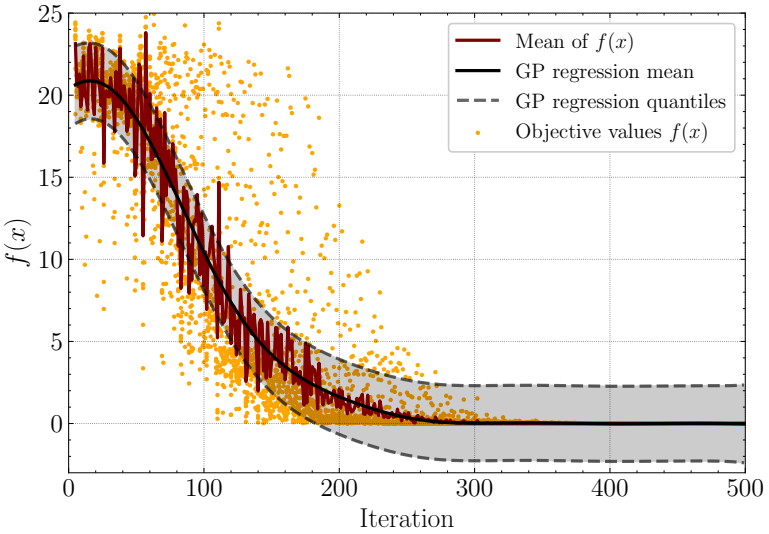
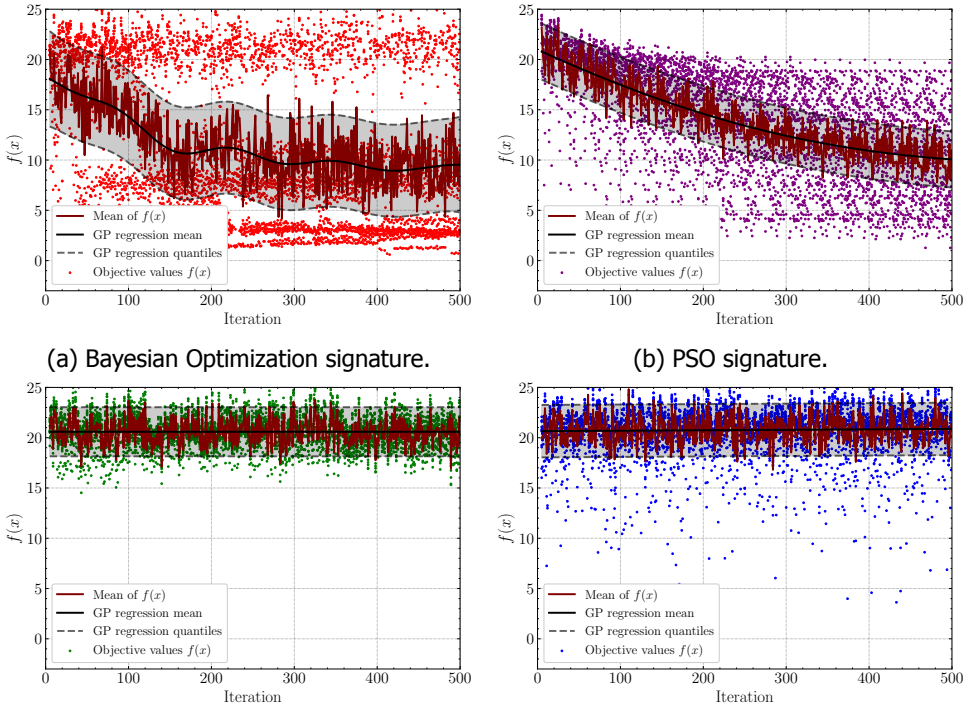


Figure 4.21: Construction of a CMA-ES heuristic signature on the noisy Ackley problem. The orange scatter represents the objective function values of many realizations. The dark red line comprehends the mean objective function values over all realizations. The mean of the Gaussian process regression is displayed in black. The 95% confidence intervals are further displayed.

For each of the selected meta-heuristics, the noisy Gaussian process fit is constructed. Figure 4.22 shows that meta-heuristic signatures for the remaining meta-heuristics.



(a) Bayesian Optimization signature.

(b) PSO signature.

(c) Adam signature.

(d) Random search signature.

Figure 4.22: Construction of the Bayesian optimization, PSO, Adam and Random search meta-heuristic signatures for the noisy Ackley problem.

Online process

A similar Gaussian process regression can be constructed during the online optimization process. However, we do not have access to multiple realizations. Therefore, the RBF kernel is optimized according to the objective values of the current realization. The predicted mean value of the Gaussian process will be compared to the entries in the database. Figure 4.23 shows how the online signatures of each of the 10 realizations compare to the offline signature of the same optimization problem.

During optimization, a new signature metric is constructed for each window of iteration. Each signature is normalized to compare the metric between objective function values within different orders of magnitude. Besides, the information from previous decisions influences the current decision. The newly constructed signature is appended to the normalized signature metrics of previous decisions as in figure 4.24. The meta-heuristics and iteration window for each part of the total signature are noted, and are compared to the same combination for every offline signature in the database. Every element of the total signature metric is used as a feature in the classification described in section 4.5.3.

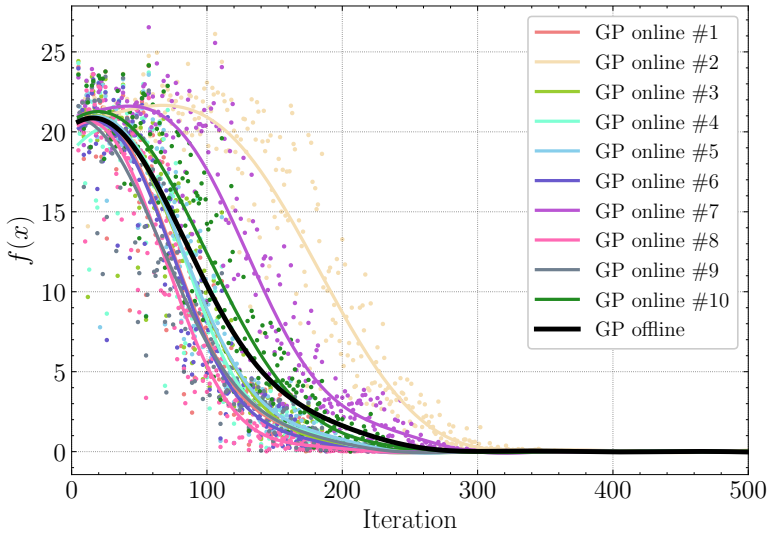


Figure 4.23: Comparison between the offline Gaussian process fit and the individual online Gaussian process fits for the noisy Ackley problem. The objective values are coloured according to their realization.

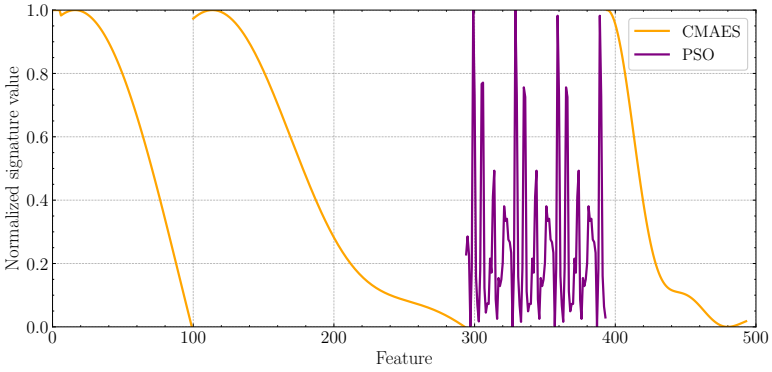


Figure 4.24: Example of the total signature metric for 5 iteration windows of 100 iterations each. For each window, the values are normalized.

4.5.2. Constructing an optimization database

After we have gathered optimization data and constructed the identifier metrics, the offline process information is stored in a `pandas` data frame. The format of these data frames is shown in table 4.5. Each function's name is the seed of the random number generator preceding with the letter '`f`'. The algorithm column denotes the algorithm for which the signature is constructed. The strategy column denotes the

best-performing algorithm for each iteration number, i.e. the strategy is a vector with its size being equal to the number of iterations. The database can be stored and imported into the online process.

name	**algorithm**	**signature**	**strategy**
f0349756184	Adam	*list*	[CMAES, CMAES, ... , Adam]
f0349756184	CMAES	*list*	[CMAES, CMAES, ... , Adam]
f0349756184	BayesianOpt	*list*	[CMAES, CMAES, ... , Adam]
f0349756184	PSO	*list*	[CMAES, CMAES, ... , Adam]
f0349756184	RandomSearch	*list*	[CMAES, CMAES, ... , Adam]
f1468498765	Adam	*list*	[BayesianOpt, PSO ... , CMAES]
f1468498765	CMAES	*list*	[BayesianOpt, PSO ... , CMAES]
f1468498765	BayesianOpt	*list*	[BayesianOpt, PSO ... , CMAES]
...	...	...	...

Table 4.5: Format of the signature database. Each row implies the optimization results of an algorithm on a particular optimization problem.

4.5.3. Classification

A classifier algorithm is used to predict the heuristic strategy during the online process. A classifier predicts the category (label) associated to a given input. New testing data can subsequently be classified according to previously trained data. The heuristic signatures are labelled according to their best-performing algorithm in the next window of iterations. To find the suggested algorithm for an unseen optimization problem, the newly constructed signature label is predicted. The elements of the unique signature metric will be used as features in the classifier.

It might be possible that some classes in the supervised learning model are over-represented and others under-represented. This phenomenon is called an imbalanced dataset. For our optimization results dataset, each of the database entries is labelled according to their best-performing heuristic in the upcoming segment of iterations. It is safe to say that one of the algorithms prevails as the best-performing heuristic across the database's problems. This will create an imbalance in the optimization results and drastically impact the prediction accuracy [163]. Also, handling multiple classes implies additional difficulty, as the boundaries between classes may overlap.

There are several methods to tackle an imbalanced dataset. First of all, the dataset could be preprocessed before building a learning model to restore the balance in the dataset. Over-sampling the minority class or under-sampling the majority class are commonly used techniques. Another technique called feature selection is to select a subset of features from the entire feature space in a way that allows optimal prediction accuracy [163]. Although preprocessing the input data could be beneficial for the balance of the dataset, it gives additional difficulties in continually keeping the generated problem set balanced. Besides, if new analytical functions and algo-

rithms are added to the framework, the balance needs to be restored. Therefore we refrain from altering the arrangement of the dataset.

Transforming the multiple-class problem to binary subsets could resolve the multi-class difficulty and handle a dataset imbalance [164]. Figure 4.25 illustrates two commonly used binarization techniques, namely one-versus-one (OVO) and one-versus-all (OVA) [163].

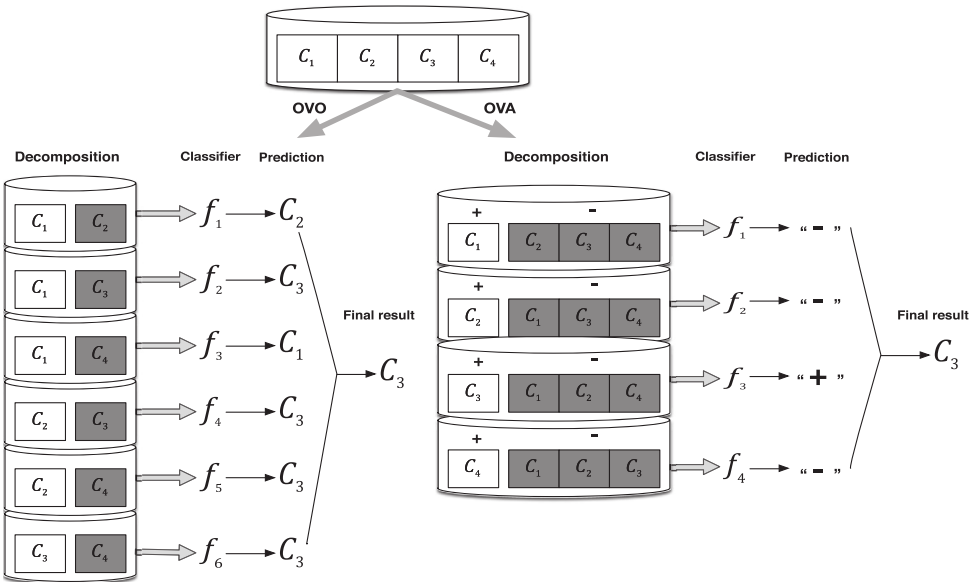


Figure 4.25: Left: One-versus-one (OVO) decomposition scheme. Right: One-versus-all (OVA) decomposition scheme. Based on the decomposition of the imbalanced dataset, the decomposed optimizers f predict the class C_3 by [163].

The OVO approach decomposes the multiple-class classification into several pairwise binary classification problems, ignoring the samples that do not belong to either of the two chosen classes [165]. Each of the classifiers f_{1-6} predict a class C_i independently. Based on the predictions' majority vote, the final result is depicted [164]. Similarly, the one-versus-all binarization decomposes the multi-class classification system into several binary problems. However, for each of the available classes, a binary classification problem is constructed against all other classes. Each classifier f_{1-4} computes the conditional probability of being labelled to the respective single class. The final result is depicted by the maximum of the conditional probabilities from all binary classifiers.

Various classification algorithms exist for solving a wide variety of classification problems. The k-nearest neighbour classifier will be discussed, including two adaptations that tackle class imbalance. Lastly, two alternative classifiers are briefly discussed, namely the C-Support Vector Classification (C-SVC) and the meta-estimator

AdaBoost [166].

k-Nearest-neighbours

The k-nearest-neighbours approach is a widely used classifier due to its simplicity and applicability. It is commonly referred to as a 'lazy learning', which means that the training data generalization is delayed until a testing query is made to the model. If new test data is to be classified, the Euclidean distance concerning the new sample's input parameters to all training data is calculated. The new instance class is equal to the majority vote of the k nearest training data, where k is an integer value specified by the user. The value of k is data-dependent: a large value of k will suppress the effects of noise, but makes the classification boundaries less distinct. The output classes are discrete variables, and all testing data is labelled beforehand. Therefore, it is a supervised learning process.

Figure 4.27 illustrates the k-nearest-neighbours procedure for a two-dimensional example 'iris dataset' retrieved from `scikit-learn` with two-dimensional features [167]. The training points are equally divided into one of three classes. Through the 15 nearest neighbours, the entire search-space can be classified according to the training samples' distance. Its position in the search-space can label new testing data.

The data-driven heuristic decision strategy uses k-nearest-neighbours classification to the following extent. First, the heuristic signature is constructed for an unknown optimization problem during the online process, as discussed in section 4.5.1. Then all heuristic signatures of that specific algorithm are retrieved from the offline database. Subsequently, the tested signature is classified. The label of the resulting class will be the meta-heuristic of choice for the next window of iterations. If the next window of iterations is fulfilled, a new heuristic signature will be constructed, and the process repeats. The features and labels of all previous windows of iterations will be included in the upcoming classifications.

Figure 4.26 illustrates this concept. The noisy Ackley function depicted as 'tested problem' is being optimized with the Adam optimizer for 100 iterations. The heuristic signature shows an increase with the number of iterations. This behaviour could mean that after a sudden improvement of solution quality, the optimization cannot find better solutions. With the k-nearest neighbour classifier, the three most similar signatures are depicted from the database. These signatures belong to different optimization problems, and their response-surfaces and solution quality metrics are displayed. Each neighbour is labelled according to the best performing algorithm in the window from 100 to 200 iterations. The majority vote of these labels classifies the tested problem. For the next window of iterations, the CMAES optimizer is used.

The OVO and OVA binarization implementations for distance-weighted k-nearest-neighbours are tested for the data-driven heuristic decision framework. The default settings from the `scikit-learn` library are used, with a k value of 5.

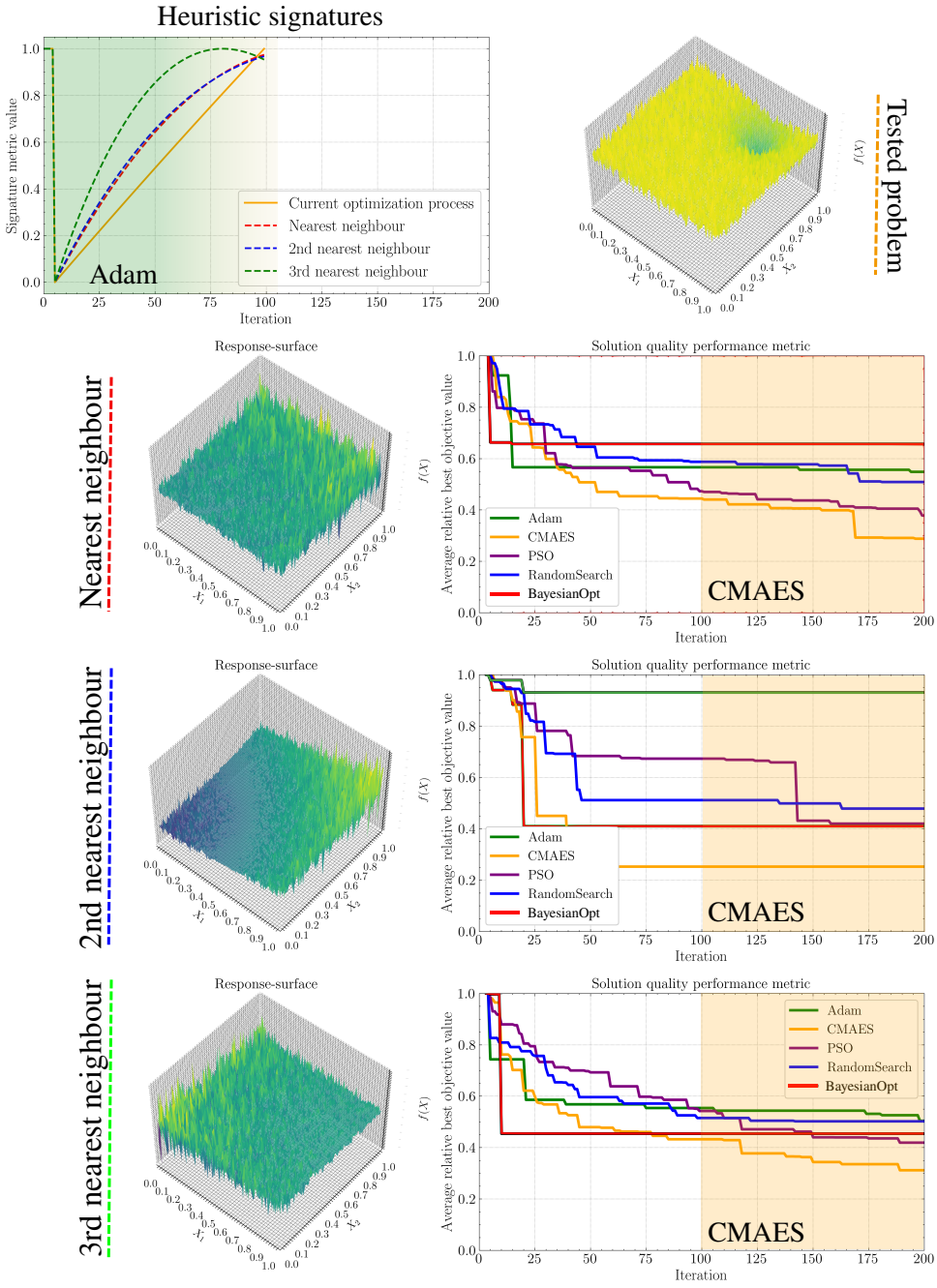


Figure 4.26: Illustration of the data-driven classification. The response-surface of the testing problem is unknown during optimization. After the first window of 100 iterations, the Adam online signature is constructed and the 3 nearest neighbours are shown. According to the solution quality performance metrics, the neighbours are classified. The next meta-heuristic is chosen by the majority vote of the nearest neighbours. In this case, CMAES will be the meta-heuristic for the next window of iterations.

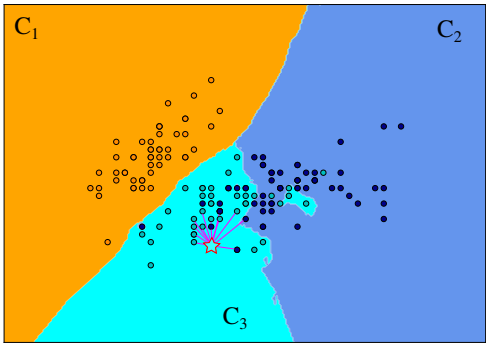


Figure 4.27: Illustration of the k-Nearest-neighbours classifier. The star-shaped sample is classified. The value of k is equal to 15 and the samples are distance weighted. The pink lines represent the distance of the neighbours [167].

C-Support Vector Classification

The C-Support Vector Classification method (C-SVC) attempts to construct a set of hyper-planes that separates individual classes. This is achieved by optimizing the hyper-parameters of a chosen kernel function. The objective of this kernel optimization is to maximize the separation margin between the classes. Although the kernel function strives for an expression that allows perfect separation of the classes, this is usually impossible. Therefore, samples that are outside a margin of the separation hyper-plane are penalized. The value of these samples' penalty is depicted with the regularization parameter C, hence the name C-Support vector classification. Once the separation has been achieved, the tested sample is easily labelled as the position related to the separation planes depict its class. Figure 4.28 shows C-SVC with both a linear and RBF kernel on the 'iris dataset'.

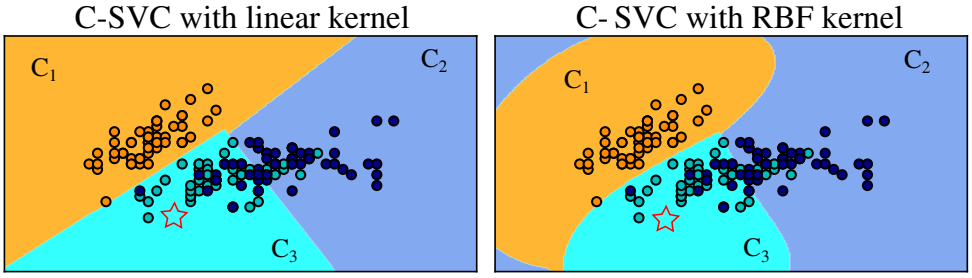


Figure 4.28: Illustration of the C-Support Vector Classifier for a linear kernel and RBF kernel. The star-shaped sample is classified [167].

The one-versus-one binarization method is used to expand C-SVC to multi-class problems. The version from the `scikit-learn` library is used, and the default value for the regularization parameter is kept at 1.0.

AdaBoost

The AdaBoost classifier is a meta-estimator that starts by fitting a classifier to the original data [166]. This is done by splitting the given input features and labels into a training and testing set. Originally, a weak classifier is used, i.e. a classifier that only shows a slight improvement over randomly selecting labels. During a sequence of learning steps, additional classifiers are fitted on modified adaptations of the input data, where weights are applied to incorrectly classified samples. In this way, subsequent iterations of the AdaBoost classifiers focus more on difficult cases as the misclassified samples' weights are 'boosted'. The majority vote of all weak classifiers will decide the outcome of the original testing case. Figure 4.29 illustrates the AdaBoost method.

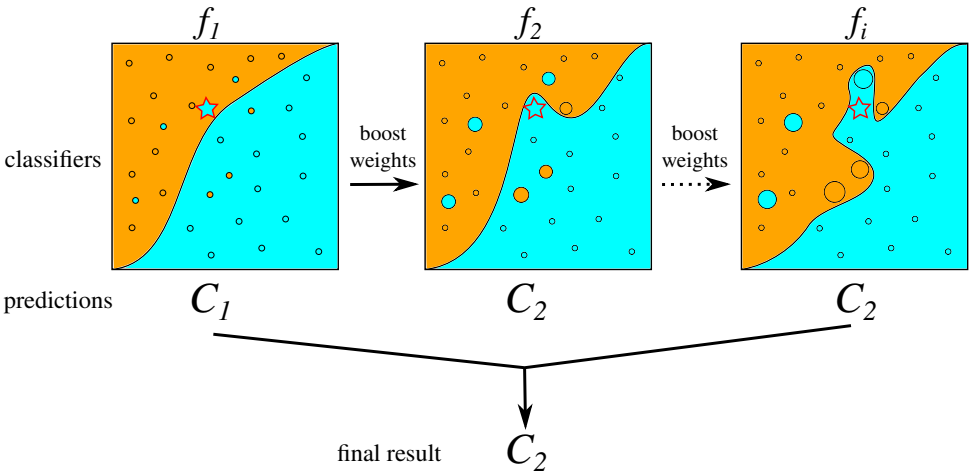


Figure 4.29: Illustration of the meta-classifier AdaBoost for a binary classification problem. The star-shaped sample is classified. Several instances of the weak classifier f_i are built with increasing weights of the mislabelled classes. The weight is depicted by the radius of the circles. The majority vote of the prediction will determine the final label.

The AdaBoost-SAMME.R adaptation from Hastie et al. is implemented [168] from the `scikit-learn` library to support multi-class classification. This alteration does not decompose the problem through binarization but adds a multi-class exponential loss function to the error calculation for each iteration.

The decision tree classifier is used as the weak learner of choice, with a maximum number of 50 boosting iterations. The learning rate depicts the amount of boosting and is kept to its default value 1.0.

Classification performance measures

The classification performance for the online process is assessed by optimizing the unseen problems with the selected heuristic and comparing the actual heuristic

strategies with the predictions. The effectiveness of multi-class classification is commonly evaluated by constructing a confusion matrix [169]. A confusion matrix is a table where each row of the square matrix represents the actual target classes' labels, while the columns denote the class predictions. The diagonal elements contain the true positive (tp) and true negative predictions (tn), whereas the upper and lower triangular parts of the matrix denote the false positives (fp) and false negatives (fn).

The confusion matrix can be used to calculate the multi-class accuracy with equation 4.13 as well as the F-score. The F-score shows the balance between precision (what proportion of positive predictions are actually truly positive) and recall (what proportion of actual true positive is identified correctly) and can be calculated with equation 4.12.

$$\mathrm{F-score} = \frac{\mathrm{tp}}{\mathrm{tp} + \frac{1}{2}(\mathrm{fp} + \mathrm{fn})} \tag{4.12}$$

$$\mathrm{Accuracy} = \frac{\mathrm{tp} + \mathrm{tn}}{\mathrm{tp} + \mathrm{fn} + \mathrm{fp} + \mathrm{tn}} \tag{4.13}$$

4.6. Results & Discussion

The data-driven heuristic decision strategy framework handles three different hyper-parameters:

- the choice of the classifier. This study investigates k-nearest neighbour one-versus-one, k-nearest neighbour one-versus-all, C-Support Vector Classification and AdaBoost.
- the number of predefined decision stages.
- the number of iterations in each window.

Each generated problem is optimized for 500 iterations. Three different instances with varying window sizes are investigated:

- large: 4 decision windows with 125 iterations each.
- medium: 5 decision windows with 100 iterations each.
- small: 6 decision windows with 75 iterations each. The last window consists of the remaining 125 iterations.

While evaluating the effectiveness of the data-driven heuristic decision strategy, it is essential to look at the performance of the solution quality and the accuracy of the decisions independently. For example, if the data-driven optimizer obtains an improved solution quality, it is not necessarily true that the classification is also accurate. The optimizer may make decisions that yield an improved strategy, but these decisions could well be built on misclassification. On the contrary, an accurate classifier will approach the solution quality of the heuristic strategy. However,

a combination of randomly selected meta-heuristics could well outperform the proposed heuristic strategy. Efforts in improving the heuristic strategy will result in a more successful optimizer in the offline process. However, if the classification based on the heuristic signature can not accurately couple the problem-specifics with previously optimized results, this enhanced performance is not obtained in the online process.

The mean margin of victory is denoted by the single heuristic optimizers and the data-driven optimizer to assess the average solution quality. The accuracy and F-score deducted from the confusion matrix will give insight into the used classifier's effectiveness.

The generated problem set results are divided into three parts:

- applying the offline heuristic strategy to the training set.
- applying the online data-driven heuristic strategy to the training set.
- applying the online data-driven heuristic strategy to the testing set.

Offline training set

Training data will first have to be collected in the offline stage. 500 two-dimensional optimization problems are generated and optimized with 10 different initial conditions by the selected meta-heuristics. The heuristic strategy is based on 5 decision windows with 100 iterations each. A more detailed explanation of the training set and the decisions and a distribution of the selected algorithms for each window can be found in appendix C.4. The margins of victory for these algorithms and the suggested heuristic strategy are displayed in figure 4.30.

The behaviour of the different optimizers' performance can be explained by the composition of the generated problem set. The gradient-based optimizer Adam is not successful on noisy or highly multimodal functions. A large part of the implemented functions is multimodal. Due to the post-analytical operations' implementation, there is a 50% chance that Gaussian noise will be added to the function. The number of optimization problems that are both unimodal and smooth is 10% in the training set. Besides, an optimal step-size hyper-parameter α is very decisive for Adam's performance. Because the default hyper-parameters of each meta-heuristic are used, this has a significant effect on the performance. Only two-dimensional objective functions are implemented in the training set. Adam and other gradient-based algorithms are more effective on highly dimensional surfaces, such as training neural networks. Lastly, derivative information is not given. Additional function evaluations are necessary to approximate the gradient numerically, which is not the standard way of using the Adam optimizer.

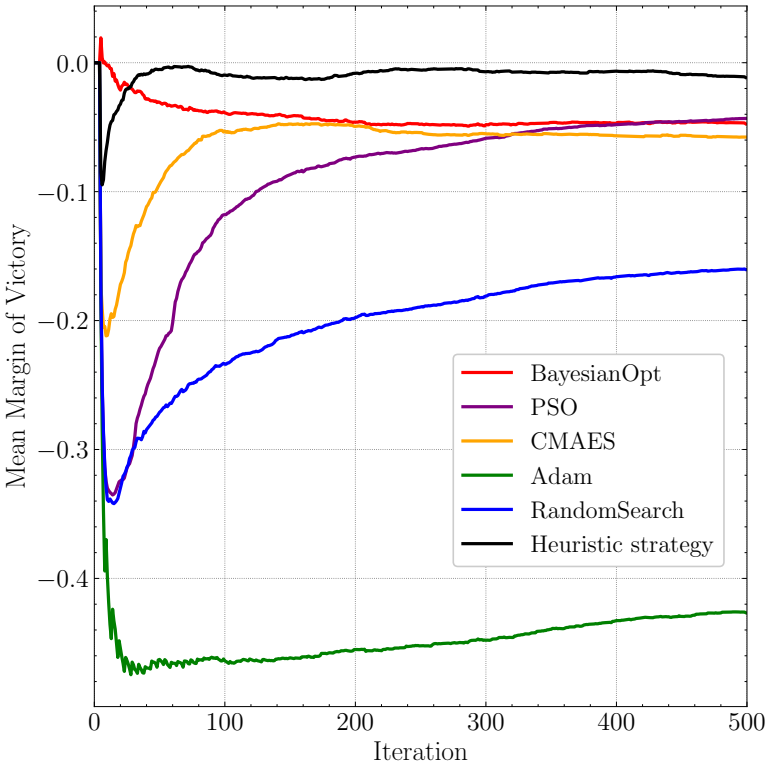


Figure 4.30: Margin of victory of selected meta-heuristics and the heuristic strategy for 500 optimization problems in the training set.

The CMAES and PSO optimizers show their strengths in different stages in the optimization process. Due to the large initial covariance hyper-parameter $\mathbf{C}_0$ of CMAES, significant exploration progress is made at the beginning of the optimization process. We see that PSO has difficulty leading up to this explorative phase. The difference in population size can explain this: the PSO optimizers have five times fewer update steps. Also, the individual particles' driving force for PSO is the best position of one of the particles $\vec{g}$. If no apparent converging factor is known at the start of the optimization process, the particles will only rely on their local best position $\vec{p}$ and velocity $\vec{v}$. As the optimization process continues, we see that the performance of PSO is catching up. We see that CMAES converges faster due to the emergence of the step-size control parameters. In contrast, the PSO optimizer particles are circling the global optimum and effectively exploiting the found optimum. Although the regular variant of PSO performs poorly in noisy objective functions, the used generational variant is beneficial in these circumstances.

From the start of the optimization process, Bayesian Optimization becomes the dominant optimizer. The information from all function evaluations is optimally used

to exploit a promising region by constructing a surrogate model. This is regulated mainly by the greedy nature of the Lower Confidence Bound acquisition function. This justifies the use of Bayesian Optimization for the bio-based composite case, as this optimizer is highly effective in the first couple of iterations. However, where other algorithms further along the optimization process take up a more exploitative nature, the Lower Confidence Bound acquisition function chooses to investigate the search space. As a result, we see the mean margin of victory decrease and eventually levels with CMAES and PSO.

The other meta-heuristics easily surpass the naive random search benchmark except for the Adam optimizer. As the optimization progresses, the mean margin of victory for random search increases slowly, guarding the 'No Free Lunch' theorem. On average, better solutions are found by lucky guesses of the input parameters.

Applying the offline heuristic strategies based on each respective optimization problem's performance metrics results in an optimizer that outperforms all other single heuristic optimizers after 30 iterations. This is reasonable, as the heuristic strategies are extracted from the optimization results and therefore adapted to the best-performing meta-heuristic on a per problem basis. It should be noted that the proposed heuristic strategies are not optimized, as this would require numerous optimization runs with different combinations of meta-heuristics in series. However, using a simple solution quality metric to guide the optimizer still results in an improved optimizer.

To indicate that solution quality is not necessarily improved by switching algorithms halfway through, we test an optimizer that randomly chooses a meta-heuristic for each iterations window. Figure 4.31 indicates the performance of this random strategy. It can be seen that after every 100 iterations, the mean margin of victory improves. However, ultimately the random switching of algorithm ensures an average performance surpassed by CMAES, PSO and Bayesian Optimization.

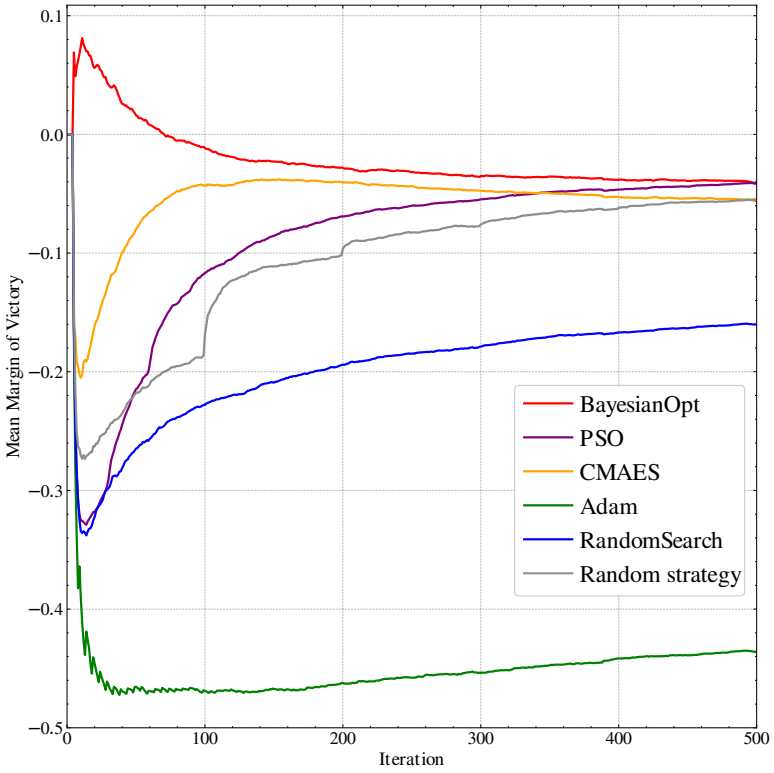


Figure 4.31: Margin of victory of selected meta-heuristics and a random strategy for 500 optimization problems in the training set.

Online training set

Now the training set is subjected to the online data-driven heuristic decision strategy, denoted as 'DataOpt' in the figures. The heuristic strategies are not known prior to optimization. The heuristic decision has to be made by using the information in the signature database. Figure 4.32 shows the mean margin of victory results for the single heuristic optimizers and the data-driven heuristic decision strategy with various classifiers.

The mean margin of victories for both AdaBoost and k-nearest neighbour approaches surpass all other meta-heuristics after 200 iterations. The performance of the C-Support Vector Classifier is less effective than the other classifiers.

Table 4.6 shows the accuracy and F-scores of the classifiers' decisions for each iteration window. The confusion matrices and more details about the decisions can be found in appendix C.4.

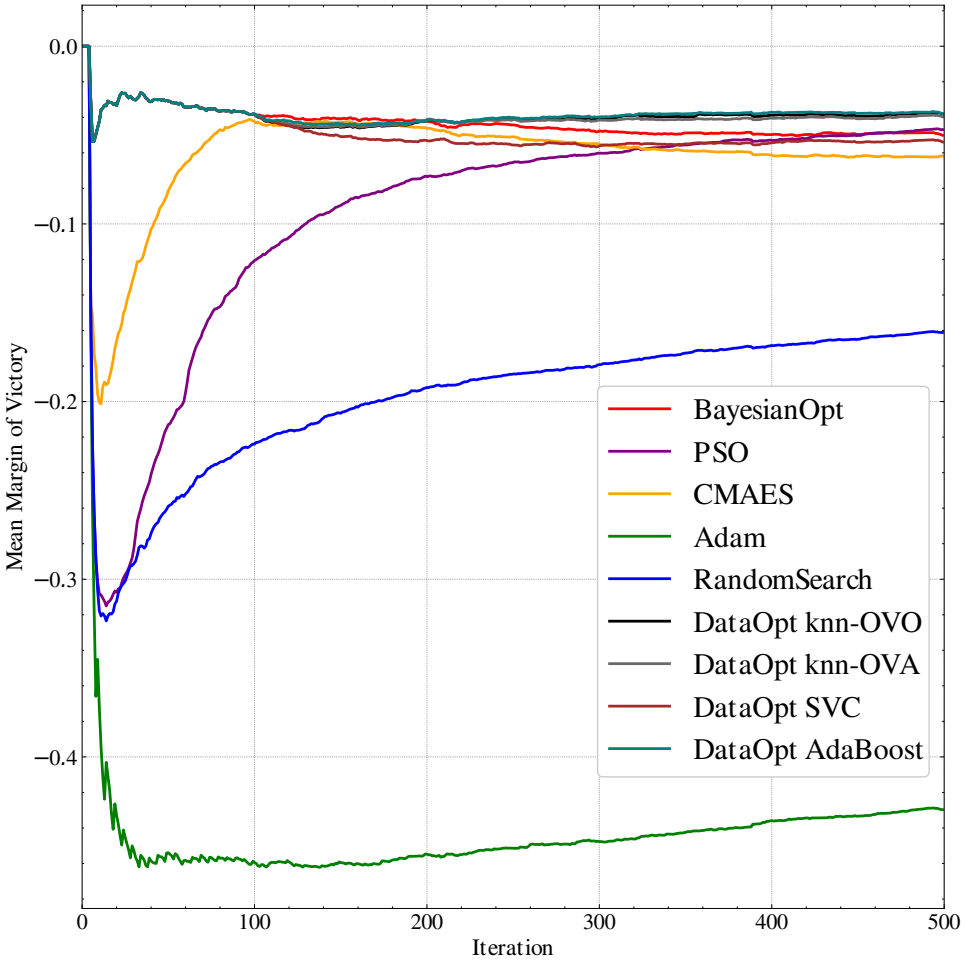


Figure 4.32: Margin of victory of selected meta-heuristics and the data-driven heuristic decision strategy with 4 different classifiers for 500 optimization problems from the training set.

	window	1	2	3	4	5	overall
k-NN: OVO	Accuracy	0.3843	0.7986	0.7825	0.7740	0.7647	**0.7433**
	F-score	0.6479	0.5620	0.5389	0.5281	0.5161	**0.4832**
k-NN: OVA	Accuracy	0.3843	0.7945	0.7757	0.7663	0.7572	**0.7335**
	F-score	0.6479	0.5570	0.5300	0.5195	0.5085	**0.4737**
C-SVC	Accuracy	0.3843	0.6963	0.6441	0.6276	0.6192	**0.5444**
	F-score	0.6479	0.4551	0.4026	0.3875	0.3791	**0.3119**
AdaBoost	Accuracy	0.3843	0.7862	0.7326	0.7252	0.7115	**0.6731**
	F-score	0.6479	0.5261	0.4687	0.4579	0.4423	**0.3909**

Table 4.6: Accuracy and F-score for the online training set with different classifiers. The overall scores are determined by taking all the decisions excluding the first window.

The accuracy and F-scores are similar in the first window. As for every problem, the first window is entirely dedicated to Bayesian Optimization. After the first window, the classifier's accuracy sets at a maximum of 0.7986 for the k-nearest neighbours one-versus-one approach.

All classifiers show a decrease in accuracy for every additional iteration window. Every decision window gives the data-driven optimizer a possibility to change meta-heuristic. The information from the previous window dilutes the signature metric at the current window by repetitive switching algorithms. This is also caused by changing population size and switching hyper-parameters. The diluted signatures differ from the pure signatures in the offline database. As we introduce more possibilities to switch algorithm, the signature gets more diluted.

The C-Support Vector Classifier demonstrates both the worst accuracy and mean margin of victory among the other classifiers. Although the AdaBoost classifier shows the second to worst overall accuracy score, it shows the best mean margin of victory performance. Both the k-nearest neighbour approaches show great accuracy scores and outperform all the single heuristic optimizers. However, the F scores of all the classifiers do not surpass a value of 0.50. It could be argued that most of the classification is still biased due to the imbalance of the dataset.

Online testing set

The testing set consists of 500 unseen generated problems. For the data-driven heuristic decision strategy, the heuristic signature database from the training set is used. Figure 4.33 shows the margin-of-victory results of the single heuristic optimizers and the data-driven optimization with the different classifiers for 5 windows of 100 iterations.

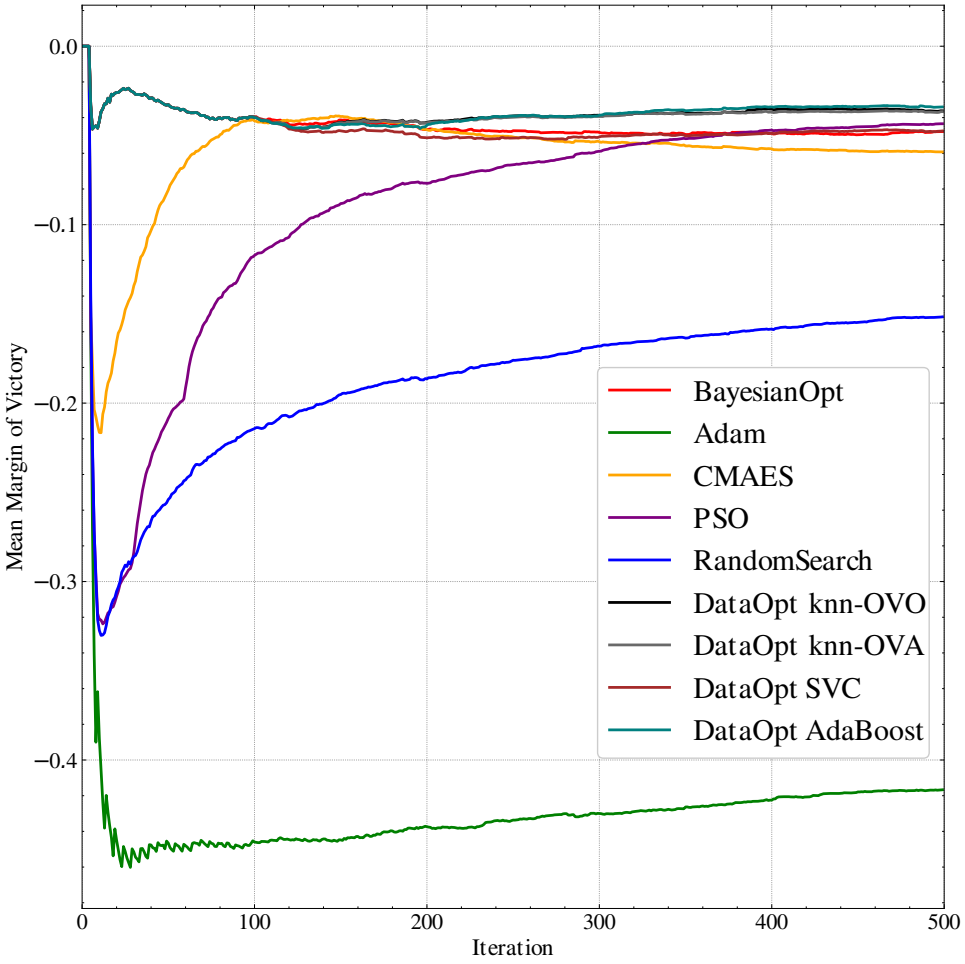


Figure 4.33: Margin of victory of selected meta-heuristics and the data-driven heuristic decision strategy with 4 different classifiers for 500 unseen optimization problems.

The performances of the various classifiers are not significantly different from the testing set to the training set. Due to the binarization methods used, classification is based on the results of several decomposed classification problems. Therefore, the advantage of optimizing the exact training set vanishes, as the decision is not based on one instance of the training data. Table 4.7 shows the accuracy and F-scores for the testing set. The accuracies and F-scores are similar within an error margin of the training set.

	window	1	2	3	4	5	overall
kNN: OVO	Accuracy	0.4234	0.7992	0.7806	0.7682	0.7603	**0.7429**
	F-score	0.6118	0.5631	0.5322	0.5140	0.5050	**0.4783**
kNN: OVA	Accuracy	0.4234	0.7936	0.7745	0.7613	0.7529	**0.7326**
	F-score	0.6118	0.5561	0.5255	0.5079	0.4971	**0.4685**
C-SVC	Accuracy	0.4234	0.6892	0.6446	0.6338	0.6322	**0.5719**
	F-score	0.6118	0.4433	0.4009	0.3917	0.3904	**0.3351**
AdaBoost	Accuracy	0.4234	0.7787	0.7253	0.7242	0.7107	**0.6782**
	F-score	0.6118	0.5148	0.4565	0.4546	0.4388	**0.3955**

Table 4.7: Accuracy and F-score for the online testing set with different classifiers. The overall scores are determined by taking all the decisions excluding the first window.

Next, the data-driven heuristic decision strategy is tested for a small, medium and large window size. The k-nearest neighbour OVA binarization is used. Figure 4.34 shows the mean margin of victory results and table 4.8 the accuracy and F-scores.

	window	1	2	3	4	5	6	overall
4 windows	Accuracy	0.3809	0.7906	0.7698	0.7534			**0.7217**
	F-score	0.6358	0.5505	0.5204	0.4993			**0.4539**
5 windows	Accuracy	0.3746	0.8000	0.7792	0.7633	0.7514		**0.7230**
	F-score	0.6640	0.5696	0.5373	0.5150	0.4991		**0.4579**
6 windows	Accuracy	0.4003	0.8147	0.7874	0.7703	0.7596	0.7505	**0.7235**
	F-score	0.7042	0.5414	0.5470	0.5249	0.5091	0.4966	**0.4551**

Table 4.8: Accuracy and F-score for the online testing set with different window sizes. The overall scores are determined by taking all the decision excluding the first window.

As shown, the size of the predefined assessment stages does not have significant influence on the mean margin of victory metric. Although the overall accuracy and F score are within an error margin, the accuracy of the smaller windows is higher.

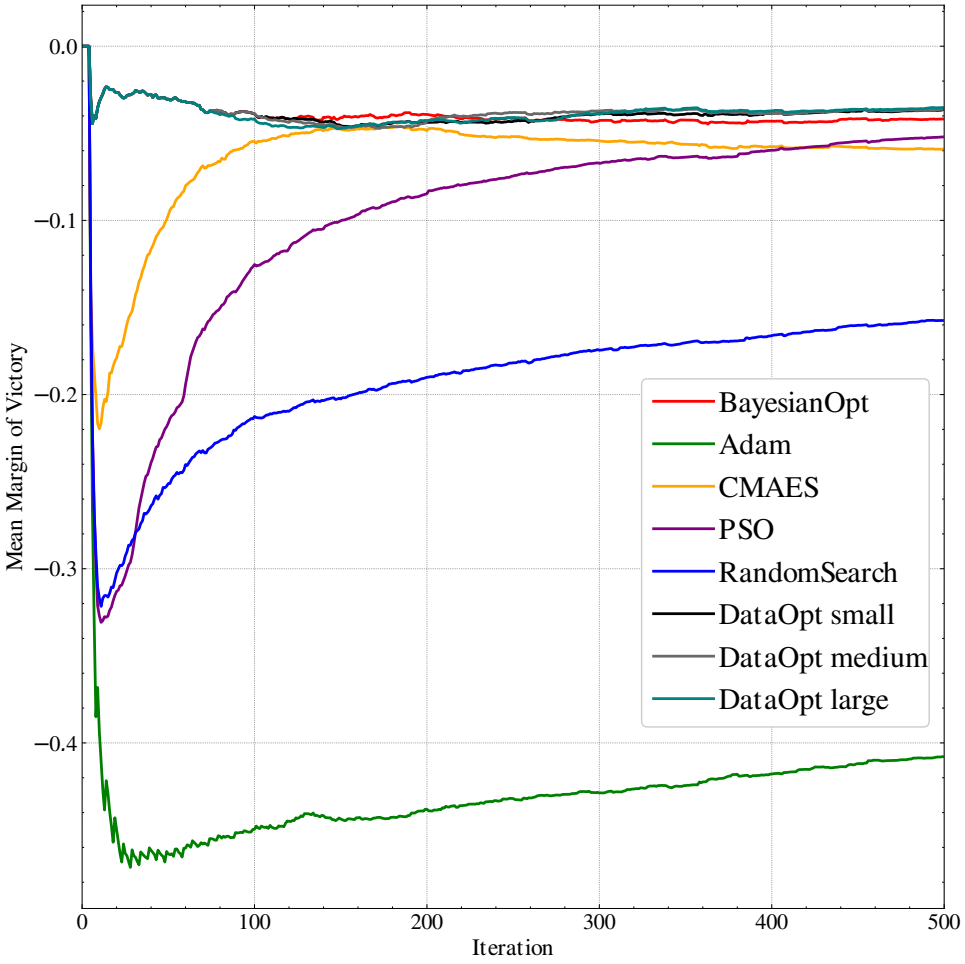


Figure 4.34: Margin of victory of selected meta-heuristics and the data-driven heuristic decision strategy with 3 different window sizes for 500 unseen optimization problems.

Computation time

Table 4.9 compares the average computation time between the selected meta-heuristics and various adaptation of the data-driven heuristic decision strategy. It can be seen that the standard deviation of the average times is approximately 25% of the average values. The parallelization of the optimization process can explain this wide distribution. The results are obtained by running the program on different machines via a computer cluster network. This resulted in a large deviation of computation time.

Bayesian Optimization's kernel optimization is an expensive mathematical opera-

tion and scales with the number of training points n. The time-complexity for this algorithm is non-polynomial $\mathcal{O}(n^3)$. The remaining meta-heuristics require only their current population to iterate. Hence they show a constant time-complexity $\mathcal{O}(c)$.

As the PSO and CMAES optimizer handle relatively simple mathematical operations, the average computation time is significantly lower than its competitors. Due to the absence of derivative information, the Adam optimizer numerically approximates the gradient. These mathematical approximations increase the average computation time of this optimizer. The implementation of a random search shows a surprisingly high average computation time. It is expected that some expressions in the source code are extremely inefficient.

optimizer	**average computation time (s)**
Bayesian Optimization	3410.62 ± 823.68
PSO	1.25 ± 0.47
CMAES	8.08 ± 2.59
Adam	21.13 ± 6.03
Random search	100.82 ± 22.18
DataOpt knn-OVA	1224.70 ± 613.15
DataOpt knn-OVO	1251.98 ± 605.48
DataOpt SVC	968.34 ± 520.57
DataOpt AdaBoost	1323.14 ± 559.42
DataOpt small	1326.93 ± 586.09
DataOpt medium	1265.67 ± 618.79
DataOpt large	1328.79 ± 619.79

Table 4.9: Comparison of average total computation time and standard deviation in seconds for 500 iterations for the selected meta-heuristics (top) and the various implementations of classifiers (middle) and window sizes (bottom) of the data-driven heuristic decision strategy.

Secondly, the average computation time between the selected meta-heuristics and various adaptation of the data-driven heuristic decision strategy is compared. Since the computational resources difference between the expensive Bayesian Optimization and the other algorithms is very significant, the average computation time is primarily determined by the meta-heuristic choice and not so much by the data-driven framework's operations. This is supported by the even larger standard deviation of approximately 50%. We can see that the C-Support Vector Classifier has the lowest computation time and AdaBoost requires the most computational resources. Figure C.10c shows that on average Bayesian Optimization is selected less frequent than the other classifiers.

The data-driven strategy with small and large windows are significantly slower than

the standard adaptation. If more predefined assessment stages are assigned to the optimization process, a signature metric must be constructed more often. As the Gaussian process kernel's optimization is computationally expensive, the average computation time is higher. In contrast, if the number of windows is reduced, the first window of iterations where Bayesian Optimization is performed will be bigger, resulting in longer runtime. Overall, the various adaptions of the data-driven heuristic strategy are 2.5 times faster in runtime compared to the best-performing meta-heuristic Bayesian Optimization.

Comparison with the 'learning to optimize' study

In this section, the data-driven heuristic decision strategy is benchmarked against the reinforcement learned optimizer in the 'learning to optimize' paper from Li and Malek [141].

The autonomous optimizer is designed from a reinforcement learning perspective. The learned optimizer is given a choice of actions in each time step, which changes the state of the environment and receives feedback based on the consequences (reward) of the depicted action. The objective is to learn an optimization update policy given the current state's feedback such that the expected reward is maximized. The actions in the reinforcement setting are changes to the update policy of the autonomous optimizer. The update policy is some function of the objective values and gradients evaluated at the current and past locations. By taking the action of subjecting a particular update policy to a training problem (environment), a reward is calculated by evaluating the speed of convergence and the objective value at the current location. Changes to the update step are promoted if the reward is positive and neglected if the reward is negative. Over many iterations, an update policy is learned to tackle new optimization problems [141].

In the paper, the autonomous optimization algorithm is subjected to three experiments: a logistic regression problem, a robust linear regression model and a two-layer neural net classifier. The data-driven heuristic decision strategy is benchmarked against the logistic and linear regression model to compare its performance. The specifications of these optimization problems can be found in appendix C.3.

- Both optimizers are trained on 90 different logistic regression loss-functions. Subsequently, the performance is measured on a test set of 100 objective functions.
- For the robust linear regression problem, a training set of 120 objective functions is used, and its performance is tested on 100 randomly generated objective functions.

An identical set of selected heuristics is used to match the same experimental conditions of the autonomous algorithm. The set of algorithms[3] consists of Gradient

[3]Gradient Descent and Momentum Gradient Descent are self-coded, whereas the Conjugate Gradient algorithm and L-BFGS are adapted from `scipy.optimize.minimize`

Descent, Momentum Gradient Descent, the Conjugate Gradient algorithm and L-BFGS. The autonomous optimizer is optimized for 100 iterations. The selected meta-heuristics require first derivative information for their update steps. As the gradient is numerically approximated in this framework, these algorithms demand 9 function evaluations per update step. Therefore, the total number of iterations for the data-driven heuristic decision strategy is scaled from 100 to 900 to compare the optimizers equally.

The starting algorithm for the data-driven strategy will be Gradient Descent. After that, a decision is made 7 times, each after a window of 100 iterations. The learning-rates (α) of Gradient Descent and Momentum Gradient Descent are optimized by a simple parameter sweep for both regression problems and are displayed in table 4.10.

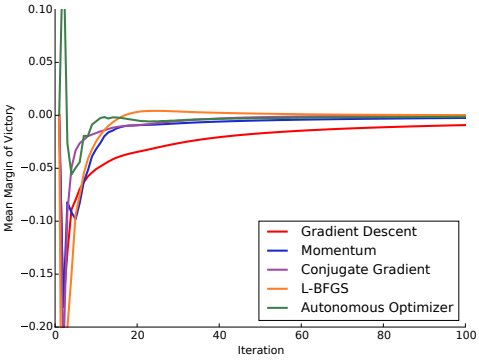
learning-rate α	**Gradient Descent**	**Momentum Gradient Descent**
Logistic regression	$1 \cdot 10^{-2}$	$5 \cdot 10^{-2}$
Robust linear regression	$1 \cdot 10^{-2}$	$5 \cdot 10^{-2}$

Table 4.10: Values of optimized learning rate α for Gradient Descent and Momentum Gradient Descent for the two benchmarking problems.

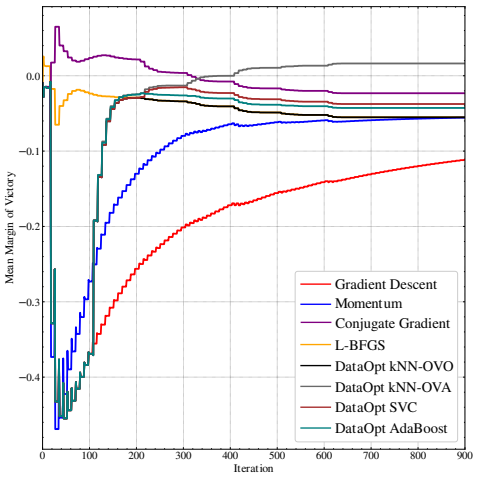
Figure 4.35a shows the results Logistic regression of the single heuristic optimizers and the autonomous optimizers of the 'Learning to optimize' paper and figure 4.35b the outcome of the data-driven heuristic decision strategy.

It can be seen that for the results in the 'Learning to optimize' paper, the performance difference is substantial for the first few iterations. Such a trend is excluded from the data-driven heuristic decision strategy ('DataOpt' in the figures). This behaviour could be the fact that in this study, the initial solutions are equal for all algorithms. In this way, there is no margin of victory difference for the individual algorithms. Information about the fairness of initial conditions is not present in the paper of Li and Malik. Both experiments report a performance drop of the data-driven algorithm in the first set of iterations. For the data-driven heuristic decision strategy, a considerable performance increase can be seen after the first data-driven decision at $t = 100$. The C-Support Vector Classifier improves its mean margin of victory after every new assessment stage, eventually reaching a significant difference between the Conjugate Gradient optimizer.

Figure 4.36a shows the robust linear regression results of the single heuristic optimizers and the autonomous optimizers of the 'Learning to optimize' paper and figure 4.36b the outcome of the data-driven heuristic decision strategy.

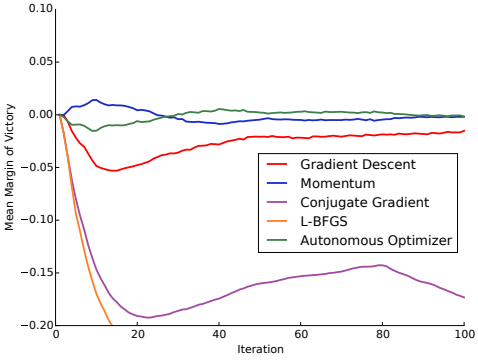


(a) Logistic regression results with the autonomous optimizer [141].

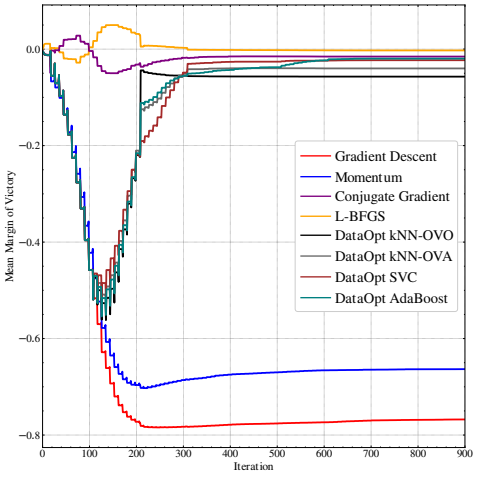


(b) Logistic regression results with the data-driven heuristic decision strategy.

Figure 4.35: Mean margin of victory comparison of the data-driven optimizers for the logistic regression problem.



(a) Robust linear regression results with the autonomous optimizer [141].



(b) Robust linear regression results with the data-driven heuristic decision strategy.

Figure 4.36: Comparison of the data-driven optimizers for the robust linear regression problem.

The results from the robust linear regression are very different from each other. First of all, the L-BFGS and Conjugate Gradient optimizer performance suffer from

the multimodal landscape in Li and Malek paper's experiment. However, both optimizers are the best-performing algorithms in the experiment in this study. There could be discrepancies between the two experiments, which influences the performance of these two optimizers. Secondly, the data-driven heuristic decision strategy lacks performance and finds, on average, a lower solution quality than Momentum Gradient Descent, L-BFGS and Conjugate Gradient.

4.7. Recommendations

This study's results have shown that a significant improvement of solution quality can be acquired for data-scarce optimization when incorporating a data-driven strategy. The continuation of this project can either lead to improving the current state of the framework or creating a new framework that introduces a reinforcement learning approach.

Improving the current state of the framework

Improvements in the current state of the framework can be sought in expanding the generated problem set, including hyper-parameter optimization, improving the heuristic strategy, changing the heuristic signature and assessing the classification.

Expand generated problem set

The generated problem set was applicable as a proof-of-concept, but only accompanies a restricted spectrum of optimization problems. Problems of different dimensionality can be implemented, which will open up new problem-specific features that benefit algorithms that incorporate dimensionality reduction. These problems can already be generated and optimized by the current framework. New analytical equations can easily be added to the framework, as long as they are single-objective and box-constrained. Data sets of real-world problems can also be added to include non-artificially generated problems.

Besides, the framework is restricted to single-objective box constrained problems only. Therefore, the performance of the selected meta-heuristics is only relevant for these types of problems. This can be seen in the performance of the Adam algorithm. Despite being state-of-the-art on optimizing neural networks, the well-known optimizer has limited effectiveness on the generated problem set in this study. In order to strive for justice for the Adam optimizer, it is essential to expand the set to other domains. The framework requires to be more flexible to handle these optimization forms.

Include variation in hyper-parameters

The set of selected meta-heuristics was restricted in consideration with the computational resources required for the offline process. This restriction of algorithms has also resulted in no hyper-parameter optimization before optimization. It was decided to look for diversity in the set of meta-heuristics, in exchange for the absence of different instances of the same optimizer. For some meta-heuristics,

hyper-parameter optimization is crucial to achieving the desired performance. In a follow-up study, the set of meta-heuristics can be expanded with multiple instances of the same optimizers with different hyper-parameter settings. This idea can be extended by changing the decision strategy to a discrete choice of one default meta-heuristic, followed by a continuous choice of hyper-parameters.

The increase of computational resources to study the alteration of the various hyper-parameters per algorithm must be taken into account. Only the most influential hyper-parameters could be parametrized with dimensionality reduction according to the findings in appendix A.2.

Improvements on the heuristic strategy

After the offline optimization for the selected heuristic, the heuristic strategy is defined by consulting the solution quality performance metric. However, information about the convergence of the algorithms during optimization is not taken into account. A second performance metric could measure when the selected meta-heuristics are converged to recommend switching to another algorithm. Switching an otherwise suspended algorithm allows the optimization process to explore other regions in the search space. Considering both the solution quality and the convergence of the algorithms' performance may improve the data-driven heuristic strategy's design.

The number of decisions and the length of the iteration window was kept fixed for the heuristic strategy. However, each meta-heuristic converges at a different rate at various stages in the optimization process. For example, the Adam optimizer will result in immediate improvement in its first iterations. However, as optimization continues, the optimizer will settle in a local optimum and is retained from improving the solution quality. On the other hand, the PSO optimizer exhibits slow convergence in its initial phase, but when a promising minimum has been located the exploitative behaviour is significantly effective. Introducing a variable window size and an assessment on every update step allows the optimizer to adapt the heuristic strategy to the algorithm's convergence needs. This will drastically increase the number of possibilities, but opens up more variability of the combination of meta-heuristics.

The initial conditions of an algorithm play a major role in its performance. The heuristic strategy is based on the average performance of the selected meta-heuristics. However, the variability of the performance should also be taken into account. As discussed earlier, a change of algorithm requires a transfer of problem-specific information and hyper-parameters. This transition is far from perfect and will affect the convergence of the optimization process. As a result, repeatedly switching affects the promised solution quality gain. Therefore, a trade-off between expected performance gain and the cost of switching should be considered before changing the optimizer.

In the current state's heuristic strategy framework, no feedback on the heuristic strategy's performance is assessed. Instead of using performance metrics to suggest a suitable heuristic strategy, the heuristic strategy could also be optimized.

Even the most successful random strategies are able to outperform the solution quality metric strategy. A loss function describing the solution quality difference between the best performing algorithm and the currently used heuristic strategy could be optimized by integer programming. This meta-optimization for the heuristic strategy is a data-scarce, noisy optimization process itself. Therefore, Bayesian Optimization could be a useful optimizer for this meta-process.

Assessing the heuristic signature and classification

The heuristic signature serves the purpose to linking the problem-specifics of the optimization to the algorithms' behaviour. As such metric did not yet exist, it was chosen to construct this metric based on the resulting objective functions and by Gaussian process regression. The classification during the online process builds on the signature features and does not have access to the underlying optimization data. Therefore, the classification relies on the effectiveness of this metric. An alteration of the heuristic signature requires the construction of an entirely new signature database. To limit this project's computational resources, it was decided to investigate different classifiers and keep the signature metric the same. Further study could be addressed by trying alternative identifier metrics.

An improvement to our strategy could be to use the standard deviation of the objective function results as the noise term for a heteroscedastic Gaussian process regression. The heteroscedasticity of the resulting objective values can only be assessed if data from multiple realizations is present. Hence, this would mean that this modification can only be implemented for the offline signature construction. However, the heteroscedasticity could contain valuable information to improve the Gaussian process regression fit.

Similar to meta-heuristics, the used classification algorithms are also heavily influenced by their hyper-parameters. The number of neighbours k can be adjusted to suppress noisy observations or make the classification boundaries more distinct, the regularization term for C C-Support Vector Classification is usually lowered from its default value when more noisy labels are present, and the learning rate for AdaBoost could be adjusted to increase the boosting of misclassified labels. Additionally, new classifiers and meta-classifiers from the `scikit-learn` library are easily implemented in the framework. A classifier analysis study can give insights into the effectiveness of the classification selection.

Improving both the effectiveness of the signature metric and the classifier's accuracy will bring the performance of testing the data-driven heuristic decision strategy more towards the performance of the offline training set's heuristic strategy.

Introducing a reinforcement learning approach

The changes described above are primarily aimed at increasing the applicable problems and possible update steps. By scaling up the number of possibilities, the applicable search space of decisions exponentially increases as the curse of dimensionality states. Besides, an additional option, such as adding an optimizer, cannot be applied until the entire training set has been analyzed with this implementation.

Finally, the data-driven heuristic decision strategy's performance can only be analyzed after the test set's margin of victory has been computed. In a more practical case, we want to have feedback on the data-driven decisions during optimization. Similar to Li and Malek's paper [141], a reinforcement learning approach could be solving the current shortcomings of the data-driven model. The idea of designing learned optimizers brings us back to the field of meta-learning. However, instead of learning the update policy, the meta-heuristic strategy could be learned.

A reinforcement learning model can be described as an agent performing actions in an unknown environment. Based on the state of the environment, a reaction is given. By repetitively gaining feedback on these actions, the agent learns a profile of actions that will maximize the reward of its actions.

Reinforcement learning model for one optimization problem.

Regarding the data-driven heuristic decision strategy for one particular problem, the environment is the unknown response-surface of the optimization problem. The action that the agent performs on the optimization problem will be the choice of meta-heuristic and hyper-parameters. The environment responds with a series of objective function values, which can be converted to a reward such as the margin of victory metric. By penalizing the weights of certain heuristic decisions with a low reward and favouring those with a high reward, the model learns which decisions will result in the best optimization results. Figure 4.37 illustrates this concept.

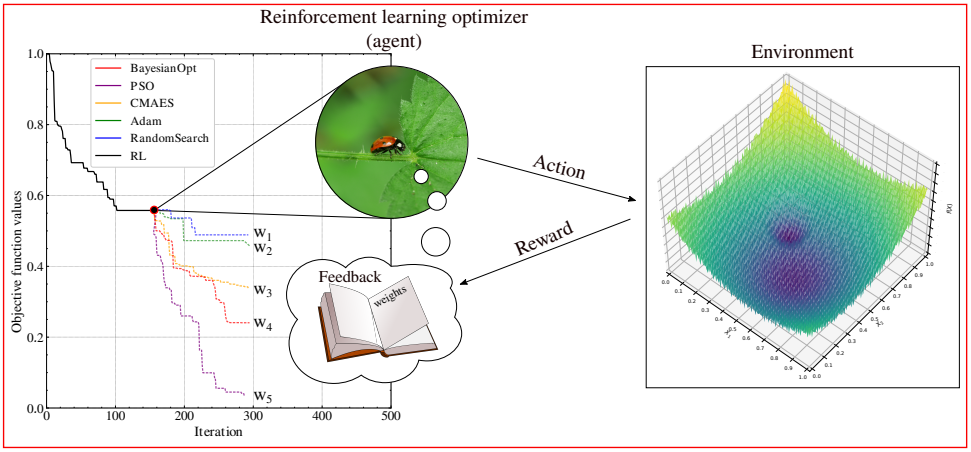


Figure 4.37: Conceptual illustration of reinforcement learned optimizer. At each update step, the ladybug learned optimizer can suggest a new meta-heuristic based on the weights w_i. The optimizer depicts its meta-heuristic choice on every update step according to the feedback of its actions.

Reinforcement learn to reinforcement learn.

To scale the reinforcement learning approach to a set of optimization problems, one can introduce a meta-learning approach. The meta-learning method consists of a

lower-level system that learns on one particular optimization problem, illustrated in figure 4.37, and a higher-level system that oversees how rapidly the lower-level system is learned. This can be described as reinforcement learning for a reinforcement learning model and is described in the paper of Wang et al. [147].

The meta-learning model is trained on a set of optimization problems. When the model encounters a new testing problem, rewards from the actions of the optimizer will trigger the decision weights of the reinforcement learned optimizer. On a global scale, the learning behaviour of the reinforcement learning model itself can be evaluated. The higher-level system tries to optimize the reinforcement model to adapt to new optimization problems quickly.

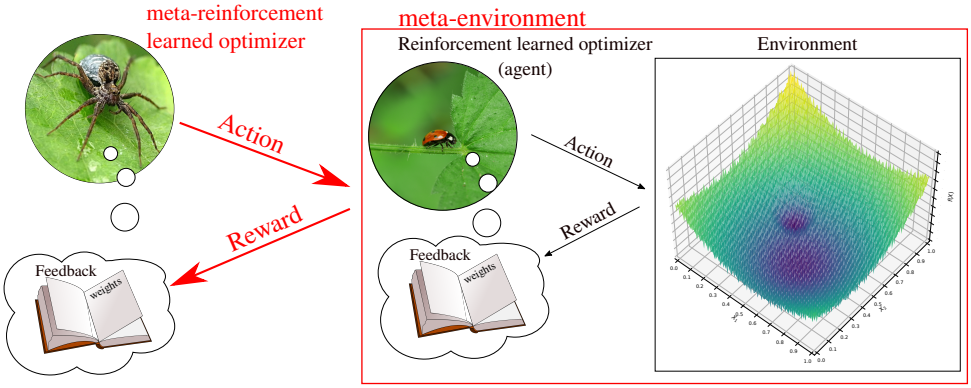


Figure 4.38: Conceptual illustration of meta-reinforced learning optimizer. The spider meta-reinforcement learned optimizer controls the learning process by handling the learning improvement of the single-problem ladybug reinforcement learned optimizer.

5

Conclusion

In conclusion, this study looked at combining algorithmic optimization with novel research in bio-based composites. The experimental research process of the bulk moulding compound has been described in terms of an optimization problem, and a Bayesian Optimization model has been constructed to generate new recipes. Employing a weighted single-objective penalty score, we combine the optimization of multiple output parameters to a single objective.

Given the limited capabilities throughout the COVID-19 pandemic to conduct experiments, we could only provide a proof-of-concept optimization model which generates recipes that can be produced and which have been shown to have good mechanical properties through a three-point bending test. For the development of the model, when implementing fibre modification or additives it is recommended to consider the increase in the search space and the speed at which bio-based composite could be produced. Finally, it is recommended to include the composites' measurement uncertainty in the optimization model.

Although collecting experimental data was challenging, a proof-of-concept model with a user-friendly interface was presented. More analysis on the effectiveness of the optimization model has to be done when sufficient data is captured.

Despite promising reviews and confident recommendations, the choice of meta-heuristic for optimization problems in general should be based on a per problem basis. By collecting optimization results from a select group of distinct meta-heuristics, we verified that a meta-heuristic choice depends on the number of iterations and the problem-specific features.

By concatenating different algorithms based on the solution quality metric, we create a heuristic decision optimizer that performs better on average than the individual optimizers. This so-called heuristic strategy is problem-dependent, as the No Free Lunch theorem withholds one algorithm to be successful on all data-scarce

problems.

To apply this method to unseen data-scarce optimization problems, we split the process into an offline stage, where a set of generated optimization problems is optimized and analyzed and an online stage, where the choice of meta-heuristic is assessed during the optimization process based on the information from the offline process.

This optimization data is captured in a unique heuristic identifier, named the heuristic signature, and captures an algorithm's behaviour on various problem-specific features. A classification algorithm depicts the choice of meta-heuristic during the online process.

The data-driven heuristic decision strategy was trained and tested on 500 objective functions. Applying a suitable heuristic strategy in the offline phase results in an optimizer that shows dominating performance, in contrast to making unfounded random decisions. The data-driven heuristic decision strategy shows excellent optimizer performance during the online phase, outperforming the individual optimizers. However, the classifier's F score for all implemented classification algorithms is at most 50%. It could be argued that most of the classification is due to the accuracy paradox of imbalanced datasets, despite the used binarization techniques. In terms of computational resources, the various adaptions of the data-driven heuristic strategy are 2.5 times faster in runtime compared to the best-performing meta-heuristic Bayesian Optimization.

Both the optimizer and classifier performance are comparable for the training and testing set. Due to the binarization methods used, classification is based on the results of several decomposed classification problems. Therefore, the advantage of optimizing the same training set vanishes, as the decision is not based on one instance of the training data.

Lastly, the data-driven heuristic decision strategy is benchmarked against the autonomous optimizer in the 'learning to optimize' paper from Li and Malek. The data-driven heuristic decision strategy shows excellent performance compared to the autonomous optimizer on the logistic regression testing set. Despite having the worst performance on the general generated problem set, the C-Support Vector Classifier is very effective. The results of the robust linear regression problem show discrepancies between the single meta-heuristic optimizers. Besides, the data-driven heuristic decision strategy is not outperforming the individual optimizers.

State-of-the-art computational optimization has found its way into solving complex engineering problems. With this study, it has been shown that even with the limited information of data-scarce and black-box situations, data-driven optimization is an effective means of improving the current standard. Future studies are recommended to improve the proposed data-driven heuristic strategy framework or implement a meta-reinforced learning approach.

Acknowledgements

As much as I wanted to predict this masters thesis's problem-specifics, the COVID-19 pandemic has had his signature written all over. I would like to thank the people of NPSP, especially Zoya and Willem, for believing in this project despite the challenging circumstances. During my time in Amsterdam, I've learned a lot and developed a great interest in bio-based composites' fascinating world.

I would also like to thank Miguel Bessa for his contributions to this project as the main supervisor. During the many meetings, we had long-lasting discussions that significantly contributed to the quality of this research. His involvement and support have led to pushing this thesis to a higher level. I have great admiration for his critical thinking and inspiring ideas. Miguel, many thanks for your endless support on this project!

Next, I would like to thanks the remaining committee members, Marcel and Kees, for their time in assessing this masters thesis. Marcel's computational materials science courses were part of the inspiration for a computational specialization during my master's degree.

I thank the entire Bessa research group for listening to my presentations and giving useful feedback. In particular, Taylan for his help on coding related questions.

Lastly, despite the strange times of social distancing, it is important to keep your friends close. I want to thank Sophia and the (ex-)inhabitants of Huize Wolf for the past year's joyful moment! Ahoe!

References

[1] M. A. Bessa, R. Bostanabad, Z. Liu, A. Hu, D. W. Apley, C. Brinson, W. Chen, and W. K. Liu, *A framework for data-driven analysis of materials under uncertainty: Countering the curse of dimensionality,* Computer Methods in Applied Mechanics and Engineering **320**, 633 (2017).

[2] M. A. Bessa, P. Glowacki, and M. Houlder, *Bayesian Machine Learning in Metamaterial Design: Fragile Becomes Supercompressible,* Advanced Materials **31**, 1 (2019).

[3] W. Böttger, M. Lepelaar, and R. Groot, *NPSP Composieten,* http://www.npsp.nl/ (2009), Accessed: 2020-09-21.

[4] D. H. Wolpert and W. G. Macready, *No free lunch theorems for optimization,* IEEE Transactions on Evolutionary Computation **1**, 67 (1997).

[5] I. M. Daniel and O. Ishai, *Materials & Design*, Vol. 17 (Elsevier BV, 1996) pp. 1–42.

[6] S. A. Miller, M. D. Lepech, and S. L. Billington, *Application of multi-criteria material selection techniques to constituent refinement in bio-based composites,* Materials and Design **52**, 1043 (2013).

[7] J. Beigbeder, L. Soccalingame, D. Perrin, J. C. Bénézet, and A. Bergeret, *How to manage bio-composites wastes end of life? A life cycle assessment approach (LCA) focused on polypropylene (PP)/wood flour and polylactic acid (PLA)/flax fibres biocomposites,* Waste Management **83**, 184 (2019).

[8] R. S. Trask, H. R. Williams, and I. P. Bond, *Self-healing polymer composites: mimicking nature to enhance performance,* Bioinspiration & Biomimetics **2**, P1 (2007).

[9] H. Ku, H. Wang, N. Pattarachaiyakoop, and M. Trada, *A review on the tensile properties of natural fiber reinforced polymer composites,* Composites Part B: Engineering **42**, 856 (2011).

[10] L. Osorio, E. Trujillo, A. W. Van Vuure, and I. Verpoest, *Morphological aspects and mechanical properties of single bamboo fibers and flexural characterization of bamboo/epoxy composites,* Journal of Reinforced Plastics and Composites **30**, 396 (2011).

[11] R. Dunne, D. Desai, R. Sadiku, and J. Jayaramudu, *A review of natural fibres, their sustainability and automotive applications,* Journal of Reinforced Plastics and Composites **35**, 1041 (2016).

[12] C. Unterweger, O. Brüggemann, and C. Fürst, *Synthetic fibers and thermoplastic short-fiber-reinforced polymers: Properties and characterization,* Polymer Composites **35**, 227 (2014).

[13] A. Vinod, M. R. Sanjay, S. Suchart, and P. Jyotishkumar, *Renewable and sustainable bio-based materials: An assessment on biofibers, biofilms, biopolymers and biocomposites,* Journal of Cleaner Production **258**, 120978 (2020).

[14] P. K. Mallick, *Fiber-Reinforced Composites: Materials, Manufacturing, and Design, Third Edition* (CRC Press, 2007).

[15] D. Puglia, J. Biagiotti, and J. M. Kenny, *A Review on Natural Fibre-Based Composites—Part II,* Journal of Natural Fibers **1**, 23 (2005).

[16] L. Kerni, S. Singh, A. Patnaik, and N. Kumar, *A review on natural fiber reinforced composites,* Materials Today: Proceedings International Conference on Aspects of Materials Science and Engineering, **28**, 1616 (2020).

[17] P. Wambua, J. Ivens, and I. Verpoest, *Natural fibres: can they replace glass in fibre reinforced plastics?* Composites Science and Technology Eco-Composites, **63**, 1259 (2003).

[18] S. C. Jana and A. Prieto, *On the development of natural fiber composites of high-temperature thermoplastic polymers,* Journal of Applied Polymer Science **86**, 2159 (2002).

[19] V. Kumar, L. Tyagi, and S. Sinha, *Wood flour–reinforced plastic composites: a review,* Reviews in Chemical Engineering **27** (2011), 10.1515/revce.2011.006.

[20] J. Gassan, *A study of fibre and interface parameters affecting the fatigue behaviour of natural fibre composites,* Composites Part A: Applied Science and Manufacturing **33**, 369 (2002).

[21] D. G. Hepworth, J. F. V. Vincent, G. Jeronimidis, and D. M. Bruce, *The penetration of epoxy resin into plant fibre cell walls increases the stiffness of plant fibre composites,* Composites Part A: Applied Science and Manufacturing **31**, 599 (2000).

[22] K. Van de Velde and P. Kiekens, *Development of a Flax/Polypropylene Composite with Optimal Mechanical Characteristics by Fiber and Matrix Modification,* Journal of Thermoplastic Composite Materials **15**, 281 (2002).

[23] K. Van de Velde and P. Kiekens, *Thermoplastic polymers: overview of several properties and their consequences in flax fibre reinforced composites,* Polymer Testing **20**, 885 (2001).

[24] M. A. López-Manchado, J. Biagiotti, and J. M. Kenny, *Comparative Study of the Effects of Different Fibers on the Processing and Properties of Polypropylene Matrix Composites,* Journal of Thermoplastic Composite Materials **15**, 337 (2002).

[25] J. D. Badia, T. Kittikorn, E. Strömberg, L. Santonja-Blasco, A. Martínez-Felipe, A. Ribes-Greus, M. Ek, and S. Karlsson, *Water absorption and hydrothermal performance of PHBV/sisal biocomposites,* Polymer Degradation and Stability **108**, 166 (2014).

[26] K. Murali Mohan Rao, K. Mohana Rao, and A. V. Ratna Prasad, *Fabrication and testing of natural fibre composites: Vakka, sisal, bamboo and banana,* Materials & Design **31**, 508 (2010).

[27] A. V. Ratna Prasad and K. Mohana Rao, *Mechanical properties of natural fibre reinforced polyester composites: Jowar, sisal and bamboo,* Materials & Design **32**, 4658 (2011).

[28] N. Saba, M. T. Paridah, and M. Jawaid, *Mechanical properties of kenaf fibre reinforced polymer composite: A review,* (2015).

[29] M. A. A. Ghani, Z. Salleh, K. M. Hyie, M. N. Berhan, Y. M. D. Taib, and M. A. I. Bakri, *Mechanical Properties of Kenaf/Fiberglass Polyester Hybrid Composite,* Procedia Engineering International Symposium on Robotics and Intelligent Sensors 2012 (IRIS 2012), **41**, 1654 (2012).

[30] C. W. Chin and B. F. Yousif, *Potential of kenaf fibres as reinforcement for tribological applications,* Wear **267**, 1550 (2009).

[31] A. Gupta, A. Kumar, A. Patnaik, and S. Biswas, *Effect of Different Parameters on Mechanical and Erosion Wear Behavior of Bamboo Fiber Reinforced Epoxy Composites,* (2011).

[32] U. Nirmal, J. Hashim, and K. O. Low, *Adhesive wear and frictional performance of bamboo fibres reinforced epoxy composite,* Tribology International **47**, 122 (2012).

[33] M. M. Kabir, H. Wang, K. T. Lau, and F. Cardona, *Chemical treatments on plant-based natural fibre reinforced polymer composites: An overview | Elsevier Enhanced Reader,* (2012).

[34] Y. Seki, *Innovative multifunctional siloxane treatment of jute fiber surface and its effect on the mechanical properties of jute/thermoset composites,* Materials Science and Engineering: A **508**, 247 (2009).

[35] X. Li, L. G. Tabil, and S. Panigrahi, *Chemical treatments of natural fiber for use in natural fiber-reinforced composites: A review,* Journal of Polymers and the Environment **15**, 25 (2007).

[36] S. H. Aziz and M. P. Ansell, *The effect of alkalization and fibre alignment on the mechanical and thermal properties of kenaf and hemp bast fibre composites: Part 1 – polyester resin matrix,* Composites Science and Technology **64**, 1219 (2004).

[37] M. N. Ichazo, C. Albano, J. González, R. Perera, and M. V. Candal, *Polypropylene/wood flour composites: treatments and properties,* Composite Structures Third International Conference on Composite Science and Technology, **54**, 207 (2001).

[38] D. Ray, B. K. Sarkar, A. K. Rana, and N. R. Bose, *Effect of alkali treated jute fibres on composite properties,* Bulletin of Materials Science **24**, 129 (2001).

[39] J. I. Preet Singh, V. Dhawan, S. Singh, and K. Jangid, *Study of Effect of Surface Treatment on Mechanical Properties of Natural Fiber Reinforced Composites,* Materials Today: Proceedings **4**, 2793 (2017).

[40] K. Sever, M. Sarikanat, Y. Seki, G. Erkan, and Ü. H. Erdogan, *The Mechanical Properties of γ-Methacryloxypropyltrimethoxy silane-treated Jute/Polyester Composites,* Journal of Composite Materials **44**, 1913 (2010).

[41] A. K. Bledzki, A. A. Mamun, M. Lucka-Gabor, and V. S. Gutowski, *The effects of acetylation on properties of flax fibre and its polypropylene composites,* Express Polymer Letters **2**, 413 (2008).

[42] B. Wang, S. Panigrahi, L. Tabil, and W. Crerar, *Pre-treatment of Flax Fibers for use in Rotationally Molded Biocomposites,* Journal of Reinforced Plastics and Composites **26**, 447 (2007).

[43] S. Mishra, A. K. Mohanty, L. T. Drzal, M. Misra, S. Parija, S. K. Nayak, and S. S. Tripathy, *Studies on mechanical performance of biofibre/glass reinforced polyester hybrid composites,* Composites Science and Technology **63**, 1377 (2003).

[44] K. Joseph, S. Thomas, and C. Pavithran, *Effect of chemical treatment on the tensile properties of short sisal fibre-reinforced polyethylene composites,* Polymer **37**, 5139 (1996).

[45] C. Vallo, J. M. Kenny, A. Vazquez, and V. P. Cyras, *Effect of Chemical Treatment on the Mechanical Properties of Starch-Based Blends Reinforced with Sisal Fibre,* Journal of Composite Materials **38**, 1387 (2004).

[46] P. K. Kushwaha and R. Kumar, *Influence of chemical treatments on the mechanical and water absorption properties of bamboo fiber composites,* Journal of Reinforced Plastics and Composites **30**, 73 (2011).

[47] M. Jawaid and H. P. S. Abdul Khalil, *Cellulosic/synthetic fibre reinforced polymer hybrid composites: A review,* Carbohydrate Polymers **86**, 1 (2011).

[48] P. Reis, J. Ferreira, F. Antunes, and J. Costa, *Flexural behaviour of hybrid laminated composites,* Composites Part A: Applied Science and Manufacturing **38**, 1612 (2007).

[49] H. Jiang, D. P. Kamdem, B. Bezubic, and P. Ruede, *Mechanical properties of poly(vinyl chloride)/wood flour/glass fiber hybrid composites,* Journal of Vinyl and Additive Technology **9**, 138 (2003).

[50] H. M. Akil, I. M. D. Rosa, C. Santulli, and F. Sarasini, *Flexural behaviour of pultruded jute/glass and kenaf/glass hybrid composites monitored using acoustic emission,* Materials Science and Engineering: A **527**, 2942 (2010).

[51] S. B. Koradiya, J. P. Patel, and P. H. Parsania, *The Preparation and PhysicoChemical Study of Glass, Jute and Hybrid Glass-Jute Bisphenol-C-Based Epoxy Resin Composites,* Polymer-Plastics Technology and Engineering **49**, 1445 (2010).

[52] O. L. S. Alsina, L. H. d. Carvalho, F. G. R. Filho, and J. R. M. d'Almeida, *Immersion Temperature Effects on the Water Absorption Behavior of Hybrid Lignocellulosic Fiber Reinforced-Polyester Matrix Composites,* Polymer-Plastics Technology and Engineering **46**, 515 (2007).

[53] A. Arbelaiz, B. Fernández, J. Ramos, A. Retegi, R. Llano-Ponte, and I. Mondragon, *Mechanical properties of short flax fibre bundle/polypropylene composites: Influence of matrix/fibre modification, fibre content, water uptake and recycling,* Composites Science and Technology **65**, 1582 (2005).

[54] C. Wu, K. Yang, Y. Gu, J. Xu, R. O. Ritchie, and J. Guan, *Mechanical properties and impact performance of silk-epoxy resin composites modulated by flax fibres,* Composites Part A: Applied Science and Manufacturing **117**, 357 (2019).

[55] S. Amico, C. Angrizani, and M. Drummond, *Influence of the Stacking Sequence on the Mechanical Properties of Glass/Sisal Hybrid Composites,* Journal of Reinforced Plastics and Composites **29**, 179 (2010).

[56] M. Ashok Kumar, G. Ramachandra Reddy, Y. Siva Bharathi, S. Venkata Naidu, and V. Naga Prasad Naidu, *Frictional Coefficient, Hardness, Impact Strength, and Chemical Resistance of Reinforced Sisal-Glass Fiber Epoxy Hybrid Composites,* Journal of Composite Materials **44**, 3195 (2010).

[57] M. Idicula, K. Joseph, and S. Thomas, *Mechanical Performance of Short Banana/Sisal Hybrid Fiber Reinforced Polyester Composites,* Journal of Reinforced Plastics and Composites **29**, 12 (2010).

[58] M. Jacob, B. Francis, S. Thomas, and K. T. Varughese, *Dynamical mechanical analysis of sisal/oil palm hybrid fiber-reinforced natural rubber composites,* Polymer Composites **27**, 671 (2006).

[59] M. M. Davoodi, S. M. Sapuan, D. Ahmad, A. Ali, A. Khalina, and M. Jonoobi, *Mechanical properties of hybrid kenaf/glass reinforced epoxy composite for passenger car bumper beam,* Materials & Design **31**, 4927 (2010).

[60] P. K. Kushwaha and R. Kumar, *Effect of Silanes on Mechanical Properties of Bamboo Fiber-epoxy Composites,* Journal of Reinforced Plastics and Composites **29**, 718 (2010).

[61] S. Mandal, S. Alam, I. Varma, and S. Maiti, *Studies on Bamboo/Glass Fiber Reinforced USP and VE Resin,* Journal of Reinforced Plastics and Composites **29**, 43 (2010).

[62] S. K. Samal, S. Mohanty, and S. K. Nayak, *Polypropylene—Bamboo/Glass Fiber Hybrid Composites: Fabrication and Analysis of Mechanical, Morphological, Thermal, and Dynamic Mechanical Behavior,* Journal of Reinforced Plastics and Composites **28**, 2729 (2009).

[63] P. V. Joseph, K. Joseph, and S. Thomas, *Effect of processing variables on the mechanical properties of sisal-fiber-reinforced polypropylene composites,* Composites Science and Technology **59**, 1625 (1999).

[64] J. George, S. S. Bhagawan, and S. Thomas, *Effects of environment on the properties of low-density polyethylene composites reinforced with pineapple-leaf fibre,* Composites Science and Technology **58**, 1471 (1998).

[65] M. M. Thwe and K. Liao, *Durability of bamboo-glass fiber reinforced polymer matrix hybrid composites,* Composites Science and Technology **63**, 375 (2003).

[66] J. Gassan and V. S. Gutowski, *Effects of corona discharge and UV treatment on the properties of jute-fibre epoxy composites,* Composites Science and Technology **60**, 2857 (2000).

[67] K. L. Pickering, M. G. A. Efendy, and T. M. Le, *A review of recent developments in natural fibre composites and their mechanical performance,* Composites Part A: Applied Science and Manufacturing **83**, 98 (2016).

[68] E. Zini and M. Scandola, *Green composites: An overview,* Polymer Composites **32**, 1905 (2011).

[69] M. Barletta, E. Pizzi, M. Puopolo, and S. Vesco, *Design and manufacture of degradable polymers: Biocomposites of micro-lamellar talc and poly(lactic acid),* Materials Chemistry and Physics **196**, 62 (2017).

[70] S. Chaitanya, I. Singh, and J. I. Song, *Recyclability analysis of PLA/Sisal fiber biocomposites,* Composites Part B: Engineering **173**, 106895 (2019).

[71] T. Mukherjee and N. Kao, *PLA based biopolymer reinforced with natural fibre: A review,* Journal of Polymers and the Environment **19**, 714 (2011).

[72] G. Mehta, A. K. Mohanty, M. Misra, and L. T. Drzal, *Bio-based resin as a toughening agent for biocomposites,* Green Chemistry **6**, 254 (2004).

[73] Y. W. Leong, M. B. A. Bakar, Z. A. M. Ishak, A. Ariffin, and B. Pukanszky, *Comparison of the mechanical properties and interfacial interactions between talc, kaolin, and calcium carbonate filled polypropylene composites,* Journal of Applied Polymer Science **91**, 3315 (2004).

[74] J. Kiehl, J. Huser, S. Bistac, and C. Delaite, *Influence of fillers content on the viscosity of unsaturated polyester resin/calcium carbonate blends:,* Journal of Composite Materials (2012), 10.1177/0021998311427780.

[75] S. Wong, R. Shanks, and A. Hodzic, *Properties of Poly(3-hydroxybutyric acid) Composites with Flax Fibres Modified by Plasticiser Absorption,* Macromolecular Materials and Engineering **287**, 647 (2002).

[76] A. Qaiss, R. Bouhfid, and H. Essabir, *Characterization and Use of Coir, Almond, Apricot, Argan, Shells, and Wood as Reinforcement in the Polymeric Matrix in Order to Valorize These Products,* (Springer International Publishing, Cham, 2015) pp. 305–339.

[77] N. A. N. Azman, M. R. Islam, M. Parimalam, N. M. Rashidi, and M. Mupit, *Mechanical, structural, thermal and morphological properties of epoxy composites filled with chicken eggshell and inorganic CaCO3 particles,* Polymer Bulletin **77**, 805 (2020).

[78] R. Kumar, J. S. Dhaliwal, G. S. Kapur, and Shashikant, *Mechanical properties of modified biofiller-polypropylene composites,* Polymer Composites **35**, 708 (2014).

[79] H.-Y. Li, Y.-Q. Tan, L. Zhang, Y.-X. Zhang, Y.-H. Song, Y. Ye, and M.-S. Xia, *Bio-filler from waste shellfish shell: Preparation, characterization, and its effect on the mechanical properties on polypropylene composites,* Journal of Hazardous Materials **217-218**, 256 (2012).

[80] L. Luan, W. Wu, M. H. Wagner, and M. Mueller, *Seaweed as novel biofiller in polypropylene composites,* Journal of Applied Polymer Science **118**, 997 (2010).

[81] S. Kuciel, K. Mazur, and P. Jakubowska, *Novel Biorenewable Composites Based on Poly (3-hydroxybutyrate-co-3-hydroxyvalerate) with Natural Fillers,* Journal of Polymers and the Environment **27**, 803 (2019).

[82] F. Dominici, D. G. García, V. Fombuena, F. Luzi, D. Puglia, L. Torre, and R. Balart, *Bio-polyethylene-based composites reinforced with alkali and palmitoyl chloride-treated coffee silverskin,* Molecules **24** (2019), 10.3390/molecules24173113.

[83] D. S. Bajwa, S. Adhikari, J. Shojaeiarani, S. G. Bajwa, P. Pandey, and S. R. Shanmugam, *Characterization of bio-carbon and ligno-cellulosic fiber reinforced bio-composites with compatibilizer,* Construction and Building Materials **204**, 193 (2019).

[84] S. Boufi, *18 - Biocomposites from olive-stone flour: A step forward in the valorization of the solid waste from the olive-oil industry,* in *Lignocellulosic Fibre and Biomass-Based Composite Materials*, Woodhead Publishing Series in Composites Science and Engineering, edited by M. Jawaid, P. Md Tahir, and N. Saba (Woodhead Publishing, 2017) pp. 387–408.

[85] A. Gharbi, R. B. Hassen, and S. Boufi, *Composite materials from unsaturated polyester resin and olive nuts residue: The effect of silane treatment,* Industrial Crops and Products **62**, 491 (2014).

[86] S. Agustin-Salazar, P. Cerruti, L. Á. Medina-Juárez, G. Scarinzi, M. Malinconico, H. Soto-Valdez, and N. Gamez-Meza, *Lignin and holocellulose from pecan nutshell as reinforcing fillers in poly (lactic acid) biocomposites,* International Journal of Biological Macromolecules **115**, 727 (2018).

[87] M. N. Prabhakar, A. U. R. Shah, K. C. Rao, and J.-I. Song, *Mechanical and thermal properties of epoxy composites reinforced with waste peanut shell powder as a bio-filler,* Fibers and Polymers **16**, 1119 (2015).

[88] D. Sánchez-Acosta, A. Rodriguez-Uribe, C. R. Álvarez-Chávez, A. K. Mohanty, M. Misra, J. López-Cervantes, and T. J. Madera-Santana, *Physicochemical characterization and evaluation of pecan nutshell as biofiller in a matrix of poly(lactic acid),* Journal of Polymers and the Environment **27**, 521 (2019).

[89] N. F. Zaaba and H. Ismail, *Thermoplastic/natural filler composites: A short review,* Journal of Physical Science **30**, 81 (2019).

[90] J. Sliseris, L. Yan, and B. Kasal, *Numerical modelling of flax short fibre reinforced and flax fibre fabric reinforced polymer composites,* Composites Part B: Engineering **89**, 143 (2016).

[91] L. Puech, K. R. Ramakrishnan, N. Le Moigne, S. Corn, P. R. Slangen, A. L. Duc, H. Boudhani, and A. Bergeret, *Investigating the impact behaviour of short hemp fibres reinforced polypropylene biocomposites through high speed imaging and finite element modelling,* Composites Part A: Applied Science and Manufacturing **109**, 428 (2018).

[92] D. S. Chethan and G. S. Venkatesh, *Experimental and Numerical Modeling of Hemp-Polyester Composites,* Wood is Good: Current Trends and Future Prospects in Wood Utilization , 333 (2017).

[93] C. A. Wallace, G. C. Saha, M. T. Afzal, and A. Lloyd, *Experimental and computational modeling of effective flexural/tensile properties of microwave pyrolysis biochar reinforced GFRP biocomposites,* Composites Part B: Engineering **175**, 107180 (2019).

[94] J.-B. Grill, M. Valko, and R. Munos, *Black-box optimization of noisy functions with unknown smoothness,* Advances in Neural Information Processing Systems , 667 (2015).

[95] F. Rothlauf, *Design of modern heuristics: principles and application* (Springer-Verlag, 2011).

[96] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms* (MIT Press, 2009).

[97] G. J. Woeginger, *Exact algorithms for NP-hard problems: A survey,* Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **2570**, 185 (2003).

[98] J. Scholz, *Genetic Algorithms and the Traveling Salesman Problem a historical Review,* arXiv:1901.05737 [cs, stat] (2018), 10.13140/RG.2.2.22632.78088/1, arXiv: 1901.05737.

[99] V. V. Vazirani, *Approximation Algorithms* (Springer Berlin Heidelberg, 2003).

[100] C. Blum and A. Roli, *Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison,* ACM Computing Surveys **35**, 268 (2003).

[101] M. Birattari, L. Paquete, T. Stützle, and K. Varrentrapp, *Classification of Metaheuristics and Design of Experiments for the Analysis of Components,* Tech. Rep. (2001).

[102] S. Salcedo-Sanz, *Modern meta-heuristics based on nonlinear physics processes : A review of models and design procedures,* Physics Reports **655**, 1 (2016).

[103] W. B. Powell, *A unified framework for stochastic optimization,* European Journal of Operational Research **275**, 795 (2019).

[104] E. Özcan, B. Bilgin, E. E. Korkmaz, and I. Cad, *A Comprehensive Analysis of Hyper-heuristics,* Intelligent Data Analysis **12**, 3 (2008).

[105] J. H. Drake, A. Kheiri, E. Özcan, and E. K. Burke, *Recent advances in selection hyper-heuristics,* European Journal of Operational Research **285**, 405 (2020).

[106] G. Ochoa, M. Hyde, T. Curtois, J. A. Vazquez-rodriguez, J. Walker, M. Gendreau, G. Kendall, B. Mccollum, A. J. Parkes, S. Petrovic, and E. K. Burke, *HyFlex : A Benchmark Framework for Cross-Domain Heuristic Search,* European Conference on Evolutionary Computation in Combinatorial Optimization , 136 (2012).

[107] W. G. Jackson, E. Özcan, and R. I. John, *Move acceptance in local search metaheuristics for cross-domain search,* Expert Systems With Applications **109**, 131 (2018).

[108] K. Sörensen, *Metaheuristics—the metaphor exposed,* International Transactions in Operational Research **22**, 3 (2015).

[109] Z. W. Geem, J. H. Kim, and G. Loganathan, *A New Heuristic Optimization Algorithm: Harmony Search,* SIMULATION **76**, 60 (2001).

[110] S. Mirjalili, S. M. Mirjalili, and A. Lewis, *Grey Wolf Optimizer,* Advances in Engineering Software **69**, 46 (2014).

[111] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, *Taking the human out of the loop: A review of Bayesian optimization,* Proceedings of the IEEE **104**, 148 (2016).

[112] E. Brochu, V. M. Cora, and N. de Freitas, *A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning,* (2010), arXiv:1012.2599 .

[113] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, *Algorithms for hyper-parameter optimization,* Advances in neural information processing systems **24**, 2546 (2011).

[114] R. Eberhart and J. Kennedy, *New optimizer using particle swarm theory,* Proceedings of the International Symposium on Micro Machine and Human Science , 39 (1995).

[115] N. Hansen and A. Ostermeier, *Proceedings of IEEE International Conference on Evolutionary Computation* (IEEE, 1996) pp. 312–317.

[116] D. P. Kingma and J. L. Ba, *Adam: A method for stochastic optimization,* 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings , 1 (2015), arXiv:1412.6980 .

[117] J. Mockus, V. Tiesis, and A. Zilinskas, *The application of Bayesian methods for seeking the extremum,* Towards Global Optimization **2**, 117 (1978).

[118] Y. Shi and R. Eberhart, *A modified particle swarm optimizer,* in *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360)* (1998) pp. 69–73.

[119] M. Zambrano-Bigiarini, M. Clerc, and R. Rojas, *Standard Particle Swarm Optimisation 2011 at CEC-2013: A baseline for future PSO improvements,* in *2013 IEEE Congress on Evolutionary Computation* (2013) pp. 2337–2344, iSSN: 1941-0026.

[120] M. R. Bonyadi and Z. Michalewicz, *SPSO2011 - Analysis of stability, local convergence, and rotation sensitivity,* GECCO 2014 - Proceedings of the 2014 Genetic and Evolutionary Computation Conference , 9 (2014).

[121] N. Padhye, K. Deb, and P. Mittal, *Boundary Handling Approaches in Particle Swarm Optimization,* in *Advances in Intelligent Systems and Computing* (Springer India, 2012) pp. 287–298.

[122] M. El-Abd and M. S. Kamel, *Black-box optimization benchmarking for noiseless function testbed using particle swarm optimization,* in *Proceedings of the 11th annual conference companion on Genetic and evolutionary computation conference - GECCO '09* (ACM Press, 2009).

[123] N. Hansen, *The CMA evolution strategy: A comparing review,* Studies in Fuzziness and Soft Computing **192**, 75 (2006).

[124] N. Qian, *On the momentum term in gradient descent learning algorithms,* Neural Networks **12**, 145 (1999).

[125] J. Duchi, E. Hazan, and Y. Singer, *Adaptive Subgradient Methods for Online Learning and Stochastic Optimization,* Journal of Machine Learning Research **12**, 2121 (2011).

[126] G. Hinton and T. Tieleman, *Lecture 6.5 - RMSProp: Divide the gradient by a running average of its recent magnitude,* (2012).

[127] L. Shi, Y. Zhang, W. Wang, J. Cheng, and H. Lu, *Rethinking The Pid Optimizer For Stochastic Optimization Of Deep Networks,* in *2020 IEEE International Conference on Multimedia and Expo (ICME)* (2020) pp. 1–6, iSSN: 1945-788X.

[128] F. R. S. Bayes, *An essay towards solving a problem in the doctrine of chances,* Biometrika **45**, 296 (1958).

[129] C. E. Rasmussen and C. K. I. Williams, *The MIT Press, Cambridge, MA, USA*, Vol. 38 (2006) pp. 715–719.

[130] J. Snoek, H. Larochelle, and R. P. Adams, *Practical Bayesian optimization of machine learning algorithms,* in *Advances in Neural Information Processing Systems* (2012) arXiv:1206.2944 .

[131] The GPyOpt authors, *GPyOpt: A Bayesian Optimization framework in Python,* `http://github.com/SheffieldML/GPyOpt` (2016).

[132] R. Moriconi, K. S. Kumar, and M. P. Deisenroth, *High-dimensional Bayesian optimization with projections using quantile Gaussian processes,* Optimization Letters **14**, 51 (2020).

[133] R. Leardi, *Genetic algorithms,* Comprehensive Chemometrics **1**, 631 (1986).

[134] F. Hutter, H. H. Hoos, and K. Leyton-Brown, *Sequential model-based optimization for general algorithm configuration, International conference on learning and intelligent optimization*, Springer , 507 (2011).

[135] Y. Bengio, *Gradient-based optimization of hyperparameters,* Neural computation **12**, 1889 (2000).

[136] J. Bergstra and Y. Bengio, *Random search for hyper-parameter optimization,* The Journal of Machine Learning Research **13**, 281 (2012).

[137] B. Doerr and C. Doerr, *Theory of parameter control for discrete black-box optimization: Provable performance gains through dynamic parameter choices, Theory of Evolutionary Computation*, , 271 (2020), 1804.05650 .

[138] J. Vesterstrom and R. Thomsen, *A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems, Proceedings of the 2004 congress on evolutionary computation (IEEE Cat. No. 04TH8753)*, **2**, 1980 (2004).

[139] P. Szynkiewicz, *Comparative Study of PSO and CMA-ES Algorithms on Black-box Optimization Benchmarks,* Journal of Telecommunications and Information Technology (2018), 10.26636/jtit.2018.127418.

[140] S. P. Lim and H. Haron, *Performance comparison of genetic algorithm, differential evolution and particle swarm optimization towards benchmark functions, 2013 IEEE Conference on Open Systems (ICOS)*, , 41 (2013).

[141] K. Li and J. Malik, *Learning to Optimize,* (2016), 1606.01885 .

[142] M. Andrychowicz, M. Denil, S. G. Colmenarejo, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. De Freitas, *Learning to learn by gradient descent by gradient descent,* Advances in Neural Information Processing Systems , 3988 (2016), arXiv:1606.04474 .

[143] Y. Chen, M. W. Hoffman, S. G. Colmenarejo, M. Denil, T. P. Lillicrap, M. Botvinick, and N. Freitas, *Learning to Learn without Gradient Descent by Gradient Descent,* in *International Conference on Machine Learning* (PMLR, 2017) pp. 748–756.

[144] C. Finn, P. Abbeel, and S. Levine, *Model-agnostic meta-learning for fast adaptation of deep networks,* in *Proceedings of the 34th International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 70, edited by D. Precup and Y. W. Teh (PMLR, International Convention Centre, Sydney, Australia, 2017) pp. 1126–1135.

[145] O. Wichrowska, N. Maheswaranathan, M. W. Hoffman, S. G. Colmenarejo, M. Denil, N. Freitas, and J. Sohl-Dickstein, *Learned Optimizers that Scale and Generalize,* in *International Conference on Machine Learning* (PMLR, 2017) pp. 3751–3760.

[146] Y. Cao, T. Chen, Z. Wang, and Y. Shen, *Learning to Optimize in Swarms,* arXiv:1911.03787 [cs, q-bio, stat] (2019), arXiv: 1911.03787.

[147] J. X. Wang, Z. Kurth-Nelson, D. Tirumala, H. Soyer, J. Z. Leibo, R. Munos, C. Blundell, D. Kumaran, and M. Botvinick, *Learning to reinforcement learn,* arXiv:1611.05763 [cs, stat] (2017), arXiv: 1611.05763.

[148] International Organization for Standardization, *Plastics-Determination of Flexural Properties (ISO 178:2010),* (2010).

[149] KNN Cellulose BV, *Recell,* `https://www.recell.eu/` (2020), Accessed: 2020-12-24.

[150] C. Chevalier and D. Ginsbourger, *Fast computation of the multi-points expected improvement with applications in batch selection,* International Conference on Learning and Intelligent Optimization (2013), 10.1007/978-3-642-44973-4$_7$.

[151] K. S. Siddharthan, M. Sasikumar, and A. Elayaperumal, *Mechanical and thermal properties of glass/polyester composite with glycerol as additive,* International Journal of Engineering Trends and Technology **7**, 61 (2014).

[152] L. Bottou, *Large-scale machine learning with stochastic gradient descent, Proceedings of COMPSTAT'2010*, , 177 (2010).

[153] S. Rajeev and C. Krishnamoorthy, *Discrete optimization of structures using genetic algorithms,* Journal of structural engineering **118**, 1233 (1992).

[154] A. P. Piotrowski, *Regarding the rankings of optimization heuristics based on artificially-constructed benchmark functions,* Information Sciences **297**, 191 (2015).

[155] I. Loshchilov, *CMA-ES with restarts for solving CEC 2013 benchmark problems,* in *2013 IEEE Congress on Evolutionary Computation* (IEEE, 2013).

[156] J. Rönkkönen, X. Li, V. Kyrki, and J. Lampinen, *A generator for multimodal test functions with multiple global optima,* in *Lecture Notes in Computer Science* (Springer Berlin Heidelberg, 2008) pp. 239–248.

[157] X. Li, K. Tang, M. N. Omidvar, Z. Yang, K. Qin, and H. China, *Functions for the CEC 2013 special session, Benchmark and competition on large-scale global optimization,* Congress on Evolutionary Computation **7**, 8 (2013).

[158] S. Surjanovic and D. Bingham, *Virtual Library of Simulation Experiments: Test Functions and Datasets,* https://www.sfu.ca/~ssurjano/optimization.html (2013), Accessed: 2020-09-21.

[159] M. A. Ardeh, *BenchmarkFcns Toolbox,* http://benchmarkfcns.xyz/ (2020), Accessed: 2021-02-03.

[160] M. D. McKay, R. J. Beckman, and W. J. Conover, *A comparison of three methods for selecting values of input variables in the analysis of output from a computer code,* Technometrics **42**, 55 (2000).

[161] V. Beiranvand, W. Hare, and Y. Lucet, *Best practices for comparing optimization algorithms,* Optimization and Engineering **18**, 815 (2017), arXiv:1709.08242 .

[162] N. S. Keskar and R. Socher, *Improving Generalization Performance by Switching from Adam to SGD,* arXiv preprint arXiv:1712.07628 (2017), 1712.07628 .

[163] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, and G. Bing, *Learning from class-imbalanced data: Review of methods and applications,* Expert Systems with Applications **73**, 220 (2017).

[164] A. Fernández, V. López, M. Galar, M. J. del Jesus, and F. Herrera, *Analysing the classification of imbalanced data-sets with multiple classes: Binarization techniques and ad-hoc approaches,* Knowledge-Based Systems **42**, 97 (2013).

[165] T. Hastie and R. Tibshirani, *Classification by pairwise coupling,* Annals of Statistics **26**, 451 (1998).

[166] Y. Freund and R. E. Schapire, *A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting,* Journal of Computer and System Sciences **55**, 119 (1997).

[167] G. Kumar and T. Head, *Scikit-optimize,* Tim Head and contributors (2017).

[168] T. Hastie, S. Rosset, J. Zhu, and H. Zou, *Multi-class AdaBoost,* Statistics and Its Interface **2**, 349 (2009).

[169] M. Sokolova and G. Lapalme, *A systematic analysis of performance measures for classification tasks,* Information Processing & Management **45**, 427 (2009).

[170] G. B. Dantzig, *Programming of Interdependent Activities: II Mathematical Model,* Econometrica **17**, 200 (1949).

[171] S. Boyd and L. Vandenberghe, *Undergraduate Convexity* (Cambridge University Press, 2013) pp. 223–251.

[172] Y. Nesterov and A. Nemirovskii, *Interior-Point Polynomial Algorithms in Convex Programming* (Society for Industrial and Applied Mathematics, 1994).

[173] F. Biscani and D. Izzo, *A parallel global multiobjective framework for optimization: pagmo,* Journal of Open Source Software **5**, 2338 (2020).

[174] Y. Shi and R. Eberhart, *Empirical study of particle swarm optimization,* in *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, Vol. 3 (IEEE, 1999) pp. 1945–1950.

[175] Z. Li-Ping, Y. Huan-Jun, and H. Shang-Xu, *Optimal choice of parameters for particle swarm optimization,* Journal of Zhejiang University-Science A **6**, 528 (2005).

[176] S. Chen, J. Montgomery, and A. Bolufé-Röhler, *Measuring the curse of dimensionality and its effects on particle swarm optimization and differential evolution,* Applied Intelligence **42**, 514 (2015).

[177] N. Hansen and S. Kern, *Evaluating the CMA Evolution Strategy on Multimodal Test Functions,* in *Parallel Problem Solving from Nature - PPSN VIII*, Lecture Notes in Computer Science, edited by X. Yao, E. K. Burke, J. A. Lozano, J. Smith, J. J. Merelo-Guervós, J. A. Bullinaria, J. E. Rowe, P. Tino, A. Kabán, and H.-P. Schwefel (Springer, Berlin, Heidelberg, 2004) pp. 282–291.

[178] H.-G. Beyer and B. Sendhoff, *Simplify your covariance matrix adaptation evolution strategy,* IEEE Transactions on Evolutionary Computation **21**, 746 (2017).

[179] M. C. J. Kennedy, *The particle swarm - explosion, stability, and convergence in a multidimensional complex space,* IEEE (2002), 10.1109/4235.985692.

A

Support information for literature review

A.1. P-type optimization

If an algorithm has a time-complexity at most polynomial that manages to solve an optimization problem exactly, we speak of P-type optimization. P-type algorithms operate on problems with specific characteristics. Both linear programming and convex problems will be briefly discussed in the upcoming sections.

Linear programming

The first group of problems are linear and quadratic optimization problems. Linear optimization, also known as linear programming, is a category of P-type problems where the objective function and constraints are linearly dependent on the input parameters. In practice, linear programming is widely used to solve planning and transportation issues. The linear problem is defined in equation A.1:

$$f(\vec{x}) = c_1 x_1 + c_2 x_2 + \ldots + c_d x_d = \sum_{j=1}^{d} c_j x_j \tag{A.1}$$

The simplex algorithm solves linear optimization problems for continuous input parameters. This algorithm establishes a simplex on the boundaries of the search space. Because of the linear constraints, one of the corners of the simplex must be the optimal solution. Since the number of the simplex vertices is finite, the optimum can be found by checking all vertices of the simplex [170].

Although the simplex method finds an exact solution, it has a worst-case time com-

plexity that is near-exponential. The ellipsoid algorithm has a worst-case time complexity that is better than the simplex method. However, the average time complexity is worse than the simplex method. Today, linear programming is widely understood, and numerous solvers can be chosen that can yield polynomial time complexity [95].

Convex problems

The second category of P-type problems is convex problems. A problem is convex when for every two points in the solution space, the straight line between those points always has a larger objective value than the respective function values. Convex optimization has its applications in various disciplines, such as signal processing and electronic circuit design. For two-dimensional problems, this can be expressed with the following statement:

$$f(ax_1 + (1-a)x_2) \leq af(x_1) + (1-a)f(x_2) \tag{A.2}$$

For $a \in [0,1]$ and $x_1, x_2 \in \mathbb{R}$. In a convex set, a local optimum is also directly the global minimum. If we can formulate a problem as a convex problem, we can solve it exactly within polynomial time with the interior-point method [171].

The interior-point method is suitable for tackling convex problems exactly within polynomial time [172]. This algorithm first converts a convex non-linear function and its constraints into a barrier function. A barrier function is a function that approaches infinity as the function value comes closer to the barrier. For simplicity, consider the minimization of the function $f(\vec{x})$ with an inequality constraint $c_i(\vec{x}) > 0$. The logarithmic barrier function $B(\vec{x}, \mu)$ will be:

$$B(\vec{x}, \mu) = f(\vec{x}) - \mu \sum_{i=1}^{m} \log(c_i(\vec{x})) \qquad \mu > 0 \tag{A.3}$$

The second part of this equation indicates the behaviour of the barrier of the solution space. This is scaled with a small positive scalar μ, the barrier parameter. As μ converges to zero, we will approach to the exact solution of $f(\vec{x})$ [171, 172].

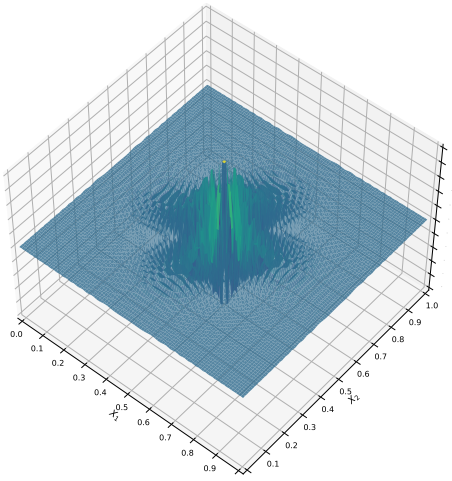
A.2. Performance of meta-heuristics on benchmark problems

The following sections demonstrate the performance of the selected meta-heuristics on several benchmark problems. The influence of the hyper-parameters and the population size are investigated.

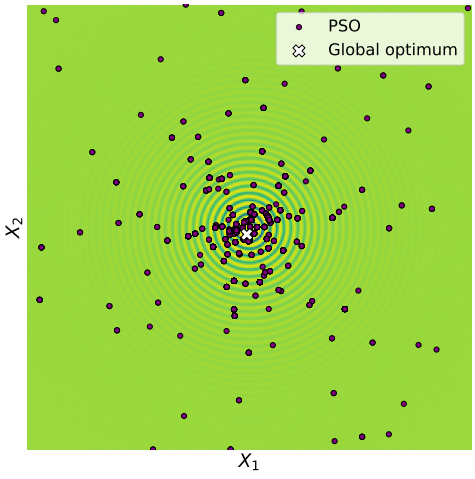
A.2.1. Performance of PSO

In the original paper of Eberhart [114], the particle swarm optimization heuristic has been benchmarked against the single-objective non-linear Schaffer F6 benchmark function. This function contains a lot of local minima in a confined space and is generally difficult to optimize. Figure A.1a shows a visual representation of the 2D Schaffer F6 function. Nowadays, more extensive benchmarking frameworks are being used to characterize this heuristic, but the Schaffer F6 function is still relevant for today's performance tests [122].

Figure A.1b shows the solutions, indicated with purple dots, generated with the Particle Swarm Optimization implementation of the Python `pygmo` library [173]. The hyper-parameters ϕ_1 and ϕ_2 are both set to 2.05. A swarm-population of 30 is used. The white cross represents the position of the global optimum. The experiment is repeated 20 times with different initial populations to counteract the influence of the initial population on the heuristic performance. The average performance with respect to the number of iterations is presented in figure A.2.



(a) Normalized 2-dimensional Schaffer F6 function



(b) Response surface of Schaffer F6 function. The white cross indicates the global optimum.

Figure A.1: Visualization of the Schaffer F6 function and the results of a Particle Swarm Optimization with random initial conditions.

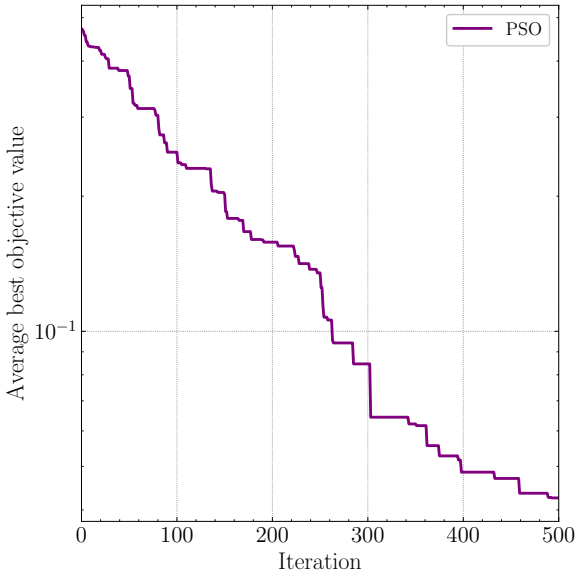
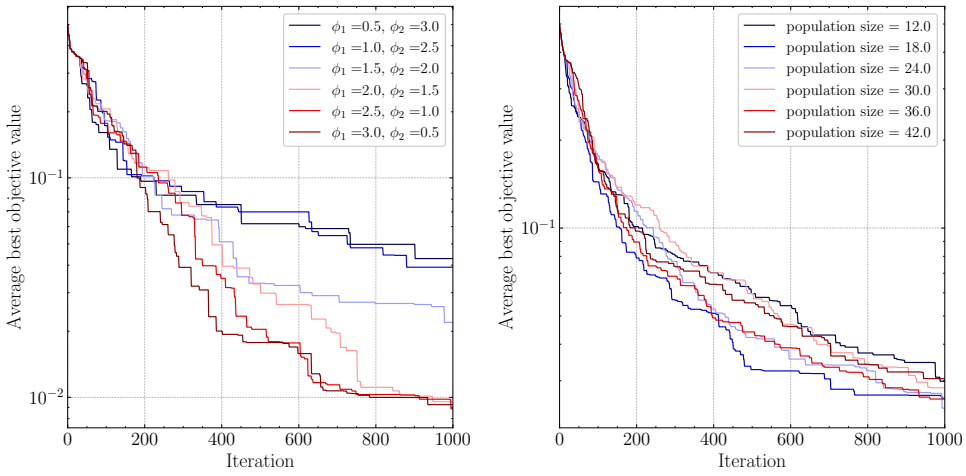


Figure A.2: The best objective value found with respect to the number of iterations, averaged over 20 runs with a different initial population each.

According to the results of Eberhart, Particle Swarm Optimization successfully found the global optimum for each run.

Due to the heavy influence of the global best particle position ($\vec{g}_t$), the solutions of Particle Swarm Optimization quickly converge towards a promising region within the search space. However, PSO drastically slows down its convergence when exploiting this region. The influence of the local best position ($\vec{p}_t$) of each particle contributes to the characteristic spiral motion. Computational experiments have shown that the influence of the hyper-parameters ϕ_1 and ϕ_2 can drastically alter the performance of PSO. In contrast, research has concluded that the size of the population has significantly less impact on the performance of the algorithm [174]. Figure A.3a shows the optimization performance on the Schaffer F6 function with different values for ϕ_1 and ϕ_2 and figure A.3b with altering population size. Changing the weights on the cognition and social term drastically changes the optimizer's performance.



(a) Changing ϕ_1 and ϕ_2.

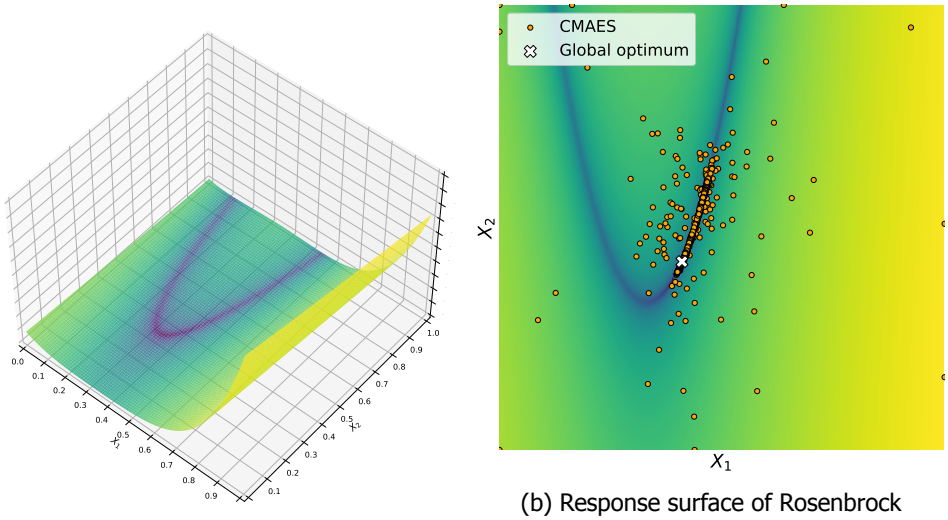
(b) Changing population size.

Figure A.3: Performance of the PSO optimizer on the Schaffer F6 problem with different values for the hyper-parameters ϕ_1 and ϕ_2 and population size. Changing the weights on the cognition and social term drastically changes the optimizers performance.

Another study on the optimal choice of hyper-parameters of PSO concluded that for lower dimensional problems ($d < 10$), a swarm size of $\lambda = 30$ is appropriate. For higher dimensional problems ($d > 30$), increasing the population size above $\lambda = 50$ will not lead to better performance [175, 176].

A.2.2. Performance of CMAES

In the original paper of Hansen [115], CMAES is compared to various simple evolutionary strategy algorithms on the single-objective unimodal Rosenbrock function. This well-known test function is still a relevant benchmark for various algorithms. Figure A.4a shows a visual representation of the 2D Rosenbrock function. An optimization run of a CMAES implementation of the Python `pygmo` library [173] on the 2D Rosenbrock problem is visualized in figure A.4b. The hyper-parameters are set to their default values as described earlier, and the population size is 6. Analogous to the experiment with particle swarm optimization, the optimization is repeated 20 times, and the average performance is plotted in figure A.5.



(a) Normalized 2-dimensional Rosenbrock function

(b) Response surface of Rosenbrock function. The white cross indicates the global optimum.

Figure A.4: Visualization of the Rosenbrock function and the results of CMAES Optimization with random initial conditions.

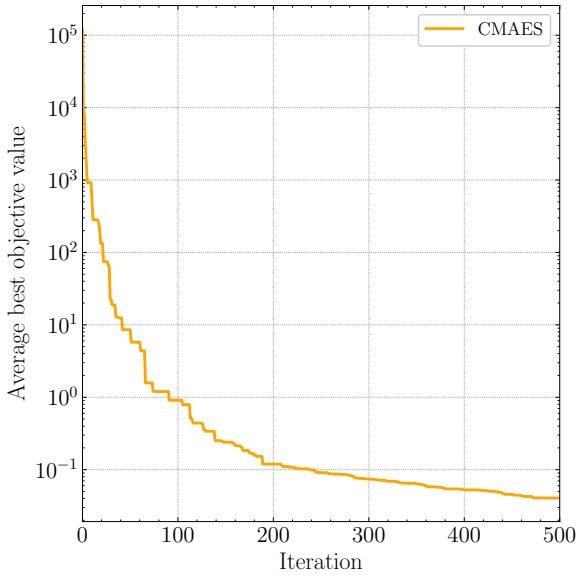


Figure A.5: Performance of the CMAES algorithm on the Rosenbrock problem. The axes represent the best objective value found with respect to the number of iterations, averaged over 20 runs with a different initial population for each run.

In contrast to particle swarm optimization, the size of the population drastically influences the algorithm's performance. Smaller populations lead to faster convergence. However, an increase in the population helps avoid the algorithm getting stuck at local minima [123]. Figure A.6 shows the optimization results of the Rosenbrock problem with different population sizes.

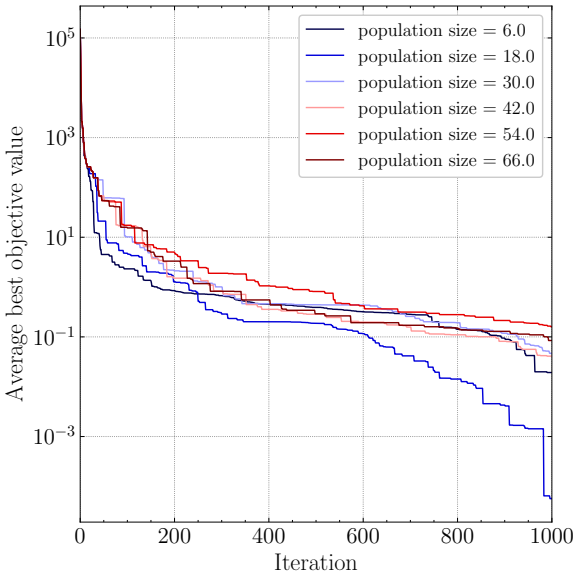
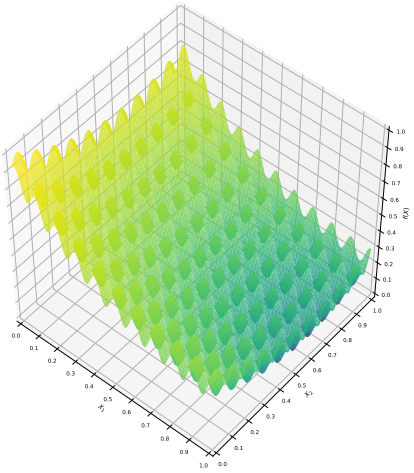


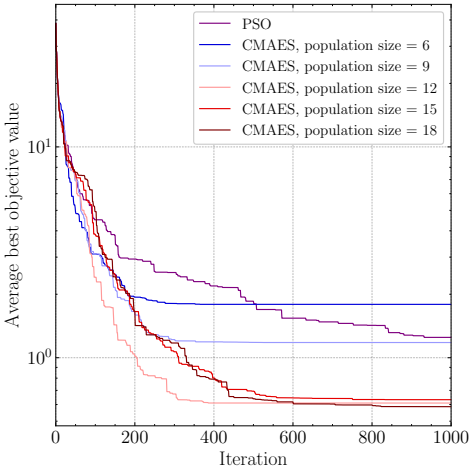
Figure A.6: Performance of the CMAES optimizer on the Rosenbrock problem with different population sizes.

If the population size is significantly large, CMAES can overcome these local minima if they can be interpreted as smaller oscillating regions of an underlying global unimodal function. The 'Rank-μ-update' and the evolutionary path keep the mean and covariance matrix from varying too much each generation so that the underlying global optimum overshadows the smaller local fluctuations. This is demonstrated in the case of the multimodal Rastrigin function in figure A.7a. The local minima are regularly distributed over the search-space. The Particle Swarm Optimizer is easily outperformed by increasing the population size, as seen in figure A.7b.

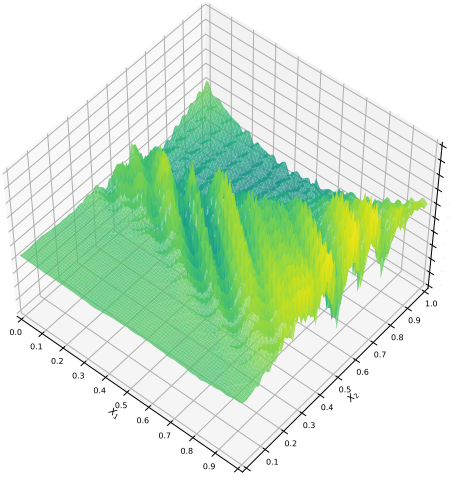
On the other hand, CMAES is easily outperformed if the objective function lacks global unimodal behaviour [177]. This is demonstrated in the case of the Composition function 4 in figure A.7c and A.7d. Despite increasing the population size, the performance of CMAES is not significantly increased in contrast to the PSO optimizer.



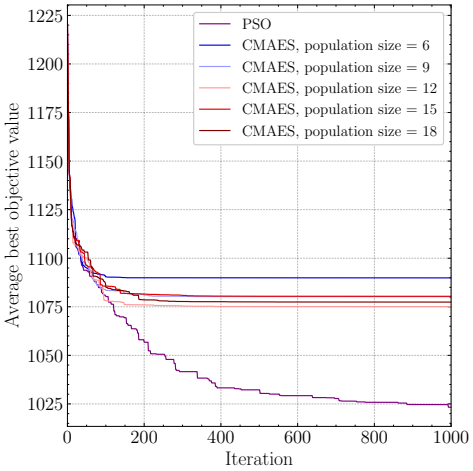
(a) Rastrigin function response-surface.



(b) Rastrigin performance of CMAES and PSO.



(c) Composition function 4 response-surface.



(d) Composition function 4 performance of CMAES and PSO.

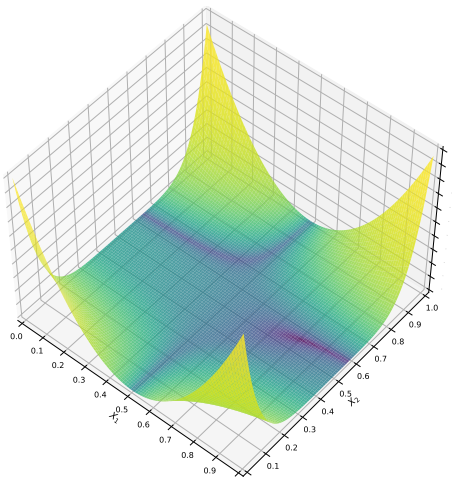
Figure A.7: Response-surface and performance of the CMAES and PSO optimizer on the Rastrigin and Composition function 4. Due to the lack of global unimodal behaviour, CMAES is easily outperformed by PSO on the composition function.

The high variability in performance with respect to the population size has lead to the development of CMAES variants such as 'BiPop-CMAES' where optimization is done multiple times in succession with increasing population size [178].

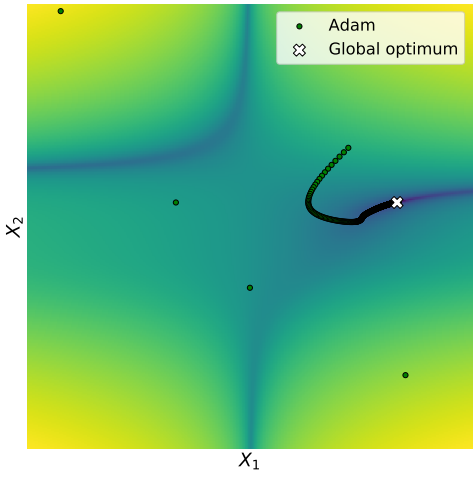
A.2.3. Performance of Adam

The original paper of Kingma [116] demonstrates that Adam can be used to optimize different deep learning models, including logistic regression and multi-layer neural networks. To keep the performance demonstration over the selected heuristics in the same scope, we are looking at Adam's behaviour at single-objective functions where the analytical form is known. For example, the Beale function is a multimodal smooth objective function used for demonstrating the performance of various gradient-based algorithms [102, 127].

Figure A.8a shows a visual representation of the 2D Beale function. A learning rate α of 10^{-2} is used and β_1 and β_2 are set to 0.9 and 0.999 respectively. The optimization of the 2D Rosenbrock problem is represented in figure A.8b. The optimization is repeated 20 times and the average performance is plotted in figure A.9.



(a) Normalized 2-dimensional Beale function



(b) Response surface of Beale function. The white cross indicates the global optimum.

Figure A.8: Visualization of the Beale function and the results of Adam Optimization with random initial conditions.

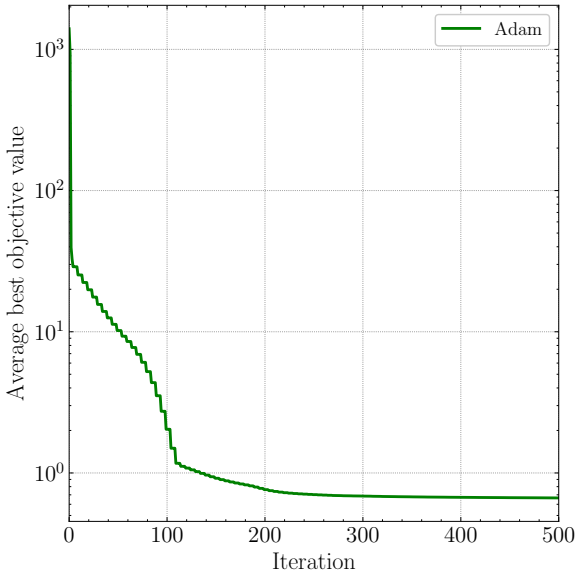
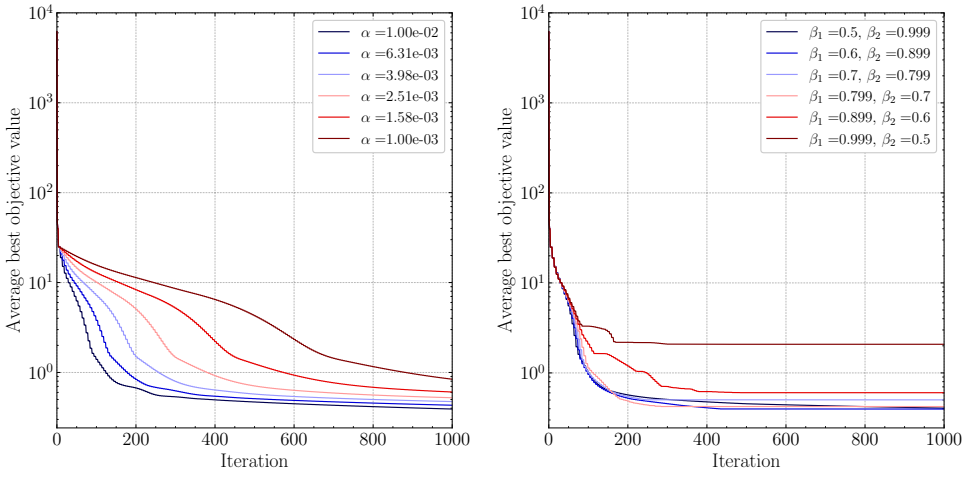


Figure A.9: Performance of the Adam algorithm on the Beale problem. The axes represent the best objective value found with respect to the number of iterations, averaged over 20 runs, each with a different initial population.



(a) Changing learning rate α.

(b) Changing hyper-parameters β_1 and β_2.

Figure A.10: Performance of the Adam optimizer on the Beale problem with different values for the learning rate α and hyper-parameters β_1 and β_2. Changing the learning rate α drastically changes the optimizers performance.

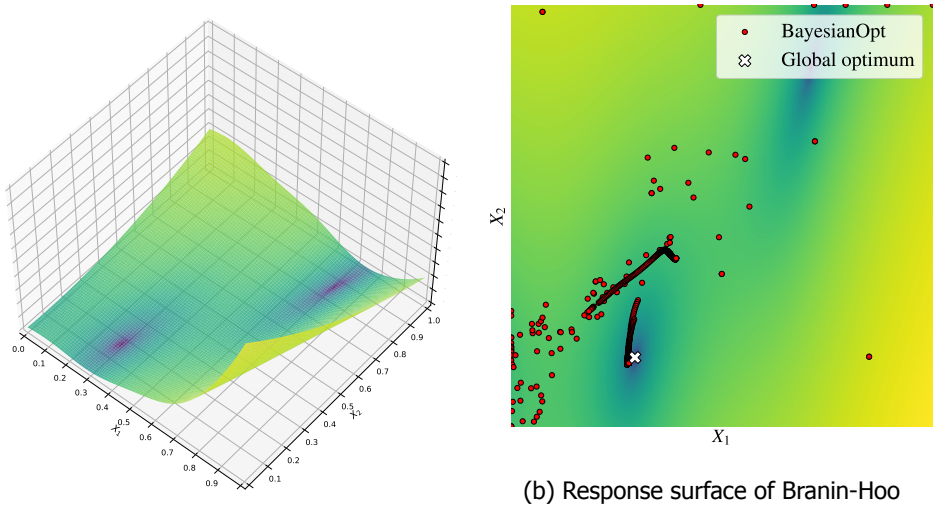
Adam converges on average after 300 iterations. Figure A.10 shows the influence of the learning rate α and the hyper-parameters β_1 and β_2. Changing the learning rate α drastically changes the optimizers performance.

A.2.4. Performance of Bayesian Optimization

The standard Branin-Hoo function is commonly used to analyze Bayesian Optimization's performance on single-objective functions, [130]. This test function has three local optima and is shown in figure A.11a. With an implementation of Bayesian Optimization of the Python library `GPyOpt` [131], the optimum in figure A.11b is sought. The "Lower Confidence Bound" acquisition function, an RBF kernel and a Gaussian process surrogate model were used to generate these results. The optimizations of 20 different initial conditions were combined to yield the average performance with respect to the number of iterations in figure A.12.

The choice for the kernel functions drastically influences the Gaussian Process surrogate model and the Bayesian Optimization outcome. Figure A.13 shows optimization results on the Branin-Hoo function with three different kernel functions. The same could be said about the acquisition function. Figure A.14 demonstrates the behaviour of Bayesian Optimization for a 'maximum probability of improvement', 'lower confidence bound' and 'expected improvement' acquisition function.

For the two-dimensional Branin-Hoo function, altering the acquisition function does not lead to a significant improvement deviation. However, the use of the linear kernel results in a significant performance drop.



(a) Normalized 2-dimensional Branin-Hoo function

(b) Response surface of Branin-Hoo function. The white cross indicates the global optimum.

Figure A.11: Visualization of the Branin-Hoo function and the results of Bayesian Optimization with random initial conditions.

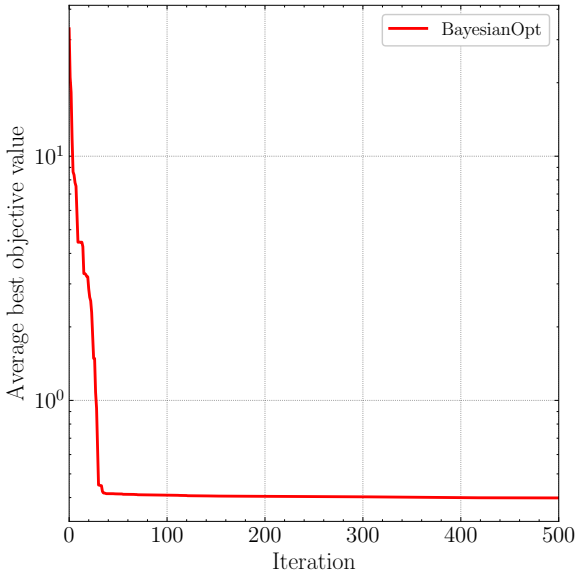


Figure A.12: Performance of the Bayesian Optimization algorithm on the Branin-Hoo problem. The axes represent the normalized best objective value found with respect to the number of iterations, averaged over 20 runs each with a different initial population.

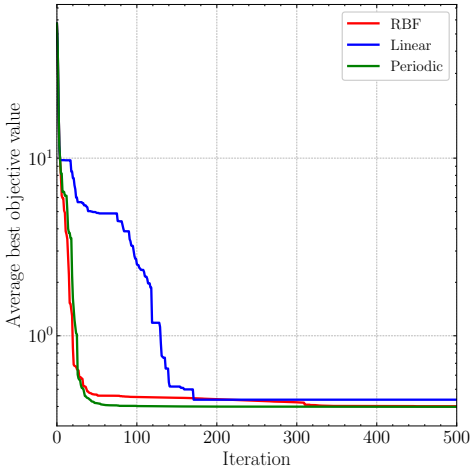


Figure A.13: Bayesian Optimization performances on the Branin-Hoo function with RBF, Linear and Periodic kernel functions.

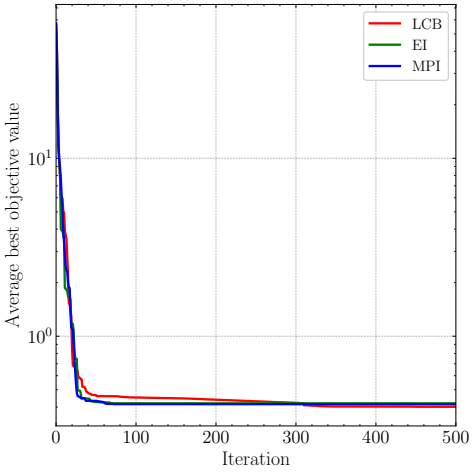


Figure A.14: Bayesian Optimization performances on the Branin-Hoo function with MPI, LCB and EI acquisition functions.

B

Support information for BMC optimization model

Appendix B.1 describes the Bayesian Optimization Python program to suggest new bulk moulding compound recipes for bio-based composites. This program is designed in collaboration with NPSP B.V. Subsequently, several new recipes are proposed within the search space boundaries. The Python script builds a Gaussian process surrogate model given the data provided.

Appendix B.2 show the three-point bending flexural testing data or the proposed recipes.

B.1. Documentation for BMC-optimizer

The program is open-source and available on GitHub. The following guidelines will help you get started with using the optimization model.

B.1.1. Downloading the program

Linux command line

1. Clone the repository.
2. Put a database excel file in the `/files/` folder.
3. Navigate to the `/Linux/` folder within the repository.
4. Run the model by executing `./model`.

Windows

1. Download the repository as ZIP-file.
2. Unzip the file somewhere locally.

3. Put a database excel file in the `/files/` folder.
4. Run the model by double-clicking `model_windows.exe`

Edit the source-code

You need to have a Python interpreter installed like Anaconda for this method.

1. Clone the repository.
2. Make sure you are using Python 3.6+.
3. Navigate to the `/Python/` folder.
4. Install the required packages: `pip install -r bmc_requirements.txt`. Alternatively, you can install the required packages yourself: `scikit-optimize`, `pandas`, `datetime`, `xlrd`.
5. Put a database Excel file in the `/files/` folder.
6. Open, edit and run the model `/Python/model.py` in any Python interpreter.

B.1.2. Contents of the repository

Upon downloading the repository, the following files are present:

- `bmc_requirements.txt`: Pip requirements file. No need to change this.
- `config.txt`: Configuration file with model parameters. Parameters are loaded up at the start of the program.
- `help.txt`: File for which the contents are displayed if you are executing 'help' within the python program.
- `input_fake.xlsx`: Artificially generated data used as a placeholder for the bio-based composite database.
- `objective.txt`: File that states the different objectives, their weights and if they are to be minimized or maximized
- `model.py`: The python model to execute.
- `calculator_for_model.xlsx`: Microsoft Excel tool that converts mass-based recipes to the 3 parameter-based format and vice-versa.

B.1.3. Database file

The user has to provide its own database file. The database file should consist of predefined input parameter columns. Each row contains a new composite. The required input columns can be seen in table B.1. The database file `input_fake.xlsx` is artificially generated data to serve as a placeholder and template. This data should not be used for experiments.

name	type fiber	type filler	fiber ratio	filler ratio	dry ratio
FlaxOli50	Flax	Olive stone	0.0995	0.1542	0.6532
ReedPeach50	Reed	Peach stone	0.0732	0.1375	0.5592
...	...	...	...	...	...

Table B.1: Format of input columns.

Any other columns are read as output columns. as can be seen in table B.2. The '**testable?**' column is optional; if a plate has failed and is not up for testing, the output columns can be left blank and the '**testable?**' cell is filled with 'no'. Subsequently, a huge penalty is given to the plate's total score, making it unlikely that the model will explore in that region. The value of the huge penalty can be set with the `badplatepenalty` parameter.

testable?	density	impact	stiffness	flex. strength	E-modulus
yes	1.7187	2.1	9.7	28.8	4770
yes	1.4654	2.3	9.5	32.1	5647
...	...	...	...	...	...

Table B.2: Format of output columns.

B.1.4. Configuration file

The `config.txt` file contains the parameters that are used in the model. For each line, the parameter is specified, followed by a space and end the value of that parameter. Lines proceeding a `#` are interpreted as comments and will not be imported by the program.

```
# search space boundaries
fiber_lb 0.05
fiber_ub 0.25
filler_lb 0.0
filler_ub 1.0
dry_lb 0.4
dry_ub 0.75

# Optimization model parameters
acq_func EI
strategy cl_min

# Recipe parameters
max_recipes 6
total_mass 2800.0
```

```
# Score parameters
badplatepenalty 2.0

author Martin van der Schelling
```


You can alter the configuration file and save it locally to quickly load up custom parameters. All available parameters can be seen in section B.1.7.

B.1.5. Objective file

This model uses a single-objective optimization method. However, by combining multiple output parameters in a single objective, several bio-based composite's mechanical properties can be considered in the search.

For every output element, a normalized value relative to the fibre-natural filler combination's best and worst properties is taken. The values are multiplied by a weight ω and added together to form a bio-based composite penalty `score`. The objective is to find the recipe with the smallest penalty.

Every line in the `objective.txt` file starts with the output parameter (same as one of the columns in the database file). Lines preceding with a `#` are ignored.

The second element defines the polarization vector a and is either `min` or `max`.

- `min`: the requested parameter has to be minimized.
- `max`: the requested parameter has to be maximized.

Next, a weight is added. If no weight is given, the normalized value is multiplied by 1.0. The final objective file looks like the following:

```
density min 0.3
stiffness max
impact max 0.6
flex. strength max 0.3
E-modulus min
```

Currently, it is not possible to edit the objective values within the program. They have to be altered and saved in the `objective.txt` file before running the model. If output data is missing for specific recipes, that parameter's influence will not be influencing the final score of the recipe.

B.1.6. Available commands

Instructions can be executed by typing certain keywords in the command line interface. The commands are split into 4 categories:

- `show` for showing parameters and data on the screen.
- `set` for setting parameters to a certain value.
- `ask` for asking the surrogate model for recipes.

- `print` for writing the configuration parameters to a file or exporting the suggested recipes.

SHOW

`show`	show the config parameters
`show config`	show the config parameters
`show data`	show the entire database. If no data is important, it will ask to `set data`.
`show <parameter>`	show the requested parameter

SET

`set config`	import the config parameters from the config file again
`set now`	set the variable `now` to the current time
`set data`	import an Excel database and set to variable `data`
`set materials`	specify the materials you want to investigate (fibre, natural filler)
`set output`	calculate the weighted single-objective penalty score
`set batch`	specify the amount of BMC doughs you want to make
`set <parameter>`	set the requested parameter to the requested value

ASK

`ask model`	ask the optimization model for new recipes

PRINT

`print config`	save the altered config parameters to the config.txt file
`print model`	print the suggested recipes to a `.csv` file
`print scores`	print the penalty scores for each output column per row to a `.csv` file

Other commands

`help/?`	show the available commands
`exit`	exit program

B.1.7. Available parameters

The parameters in table B.3 can be altered and viewed by executing the `set` and `show` commands:

B

name	type	description
acq_func	string	Type of acquisition function used.
author	string	Name that will be printed on the recipes
badplatepenalty	float	Penalty that is given to plates that cannot be tested (>= 1.0)
batch	integer	Number of recipes to be generated
config	list	Copy of the imported config.txt file
data	DataFrame	Imported BMC database
databasename	string	Filename of the imported database
dry_lb	float	Lower bound of the dry materials parameter
dry_ub	float	Upper bound of the dry materials parameter
fiber_lb	float	Lower bound of the fibre parameter
fiber_ub	float	Upper bound of the fibre parameter
fiber_t	string	Name of the fibre to investigate
filler_lb	float	Lower bound of the natural filler parameter
filler_ub	float	Upper bound of the natural filler parameter
filler_t	string	Name of the natural filler to investigate
max_recipes	integer	Maximum number of recipes to request
now	datetime	Current time at start of program
output	Series	Calculated total penalty score of each selected entry in the database
scores	DataFrame	Calculated penalty score for each individual output of each selected entry in the database
strategy	string	Parallel Bayesian Optimization strategy
total_mass	float	Total mass of each BMC dough

Table B.3: Overview of customizable parameters in the BMC optimization model

B.2. Three-point bending flexural test data

Five specimens are cut by a water-jet cutter for each bio-based composite recipe and subjected to the three-point bending flexural test. The 'ISO 178 (2010) Plastics - Determination of Flexural Properties (Method B)' standard is used [148]. The test is performed on an Instron 5969 50kN Dual Column universal testing system. The tests are performed at room temperature and with a relative humidity of 64%. For each testing result, the mean values of the flexural modulus and strength are used as an output variable in the optimization model.

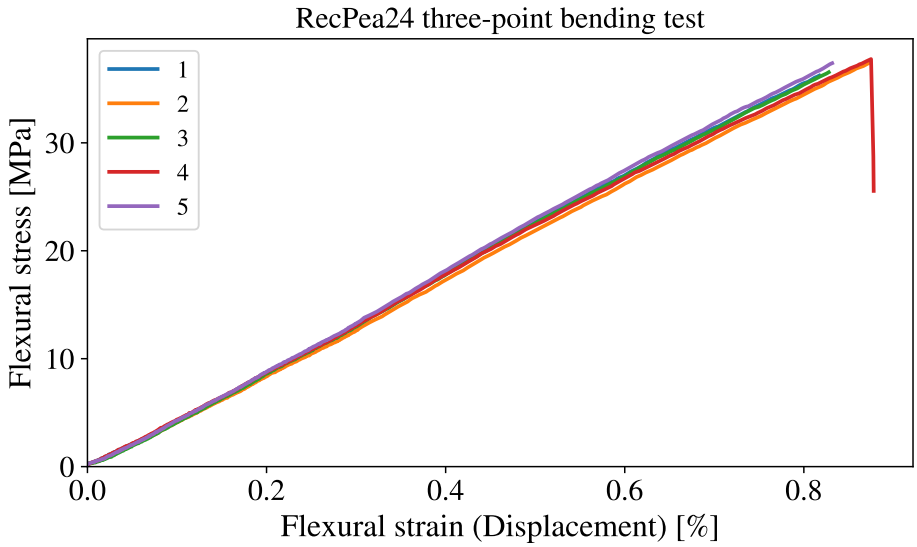


Figure B.1: Stress-strain curve of the Recell-Peach composite with $x_{\text{fibre}} = 0.2411$, $x_{\text{filler}} = 0.1258$ and $x_{\text{dry}} = 0.6923$

Sample	Flexural Strength (MPa)	Flexural Strain (%)	Flexural Modulus (MPa)	Force at maximum Flexure load (N)	Displacement at Break (mm)
1	41.5	0.95	4410	169.15	1.58
2	39.2	0.92	4260	170.71	1.29
3	40	0.91	4460	166.21	1.51
4	37.8	0.88	4370	164.74	1.25
5	42.7	0.97	4510	178.51	1.55
Minimum	37.8	0.88	4260	164.74	1.25
Maximum	42.7	0.97	4510	178.51	1.58
Mean	40.2	0.93	4400	169.86	1.44
S.D.	1.95	0.04	96.02	5.37	0.15

Table B.4: Three-point bending test data of the Recell-Peach composite with $x_{\text{fibre}} = 0.2411$, $x_{\text{filler}} = 0.1258$ and $x_{\text{dry}} = 0.6923$

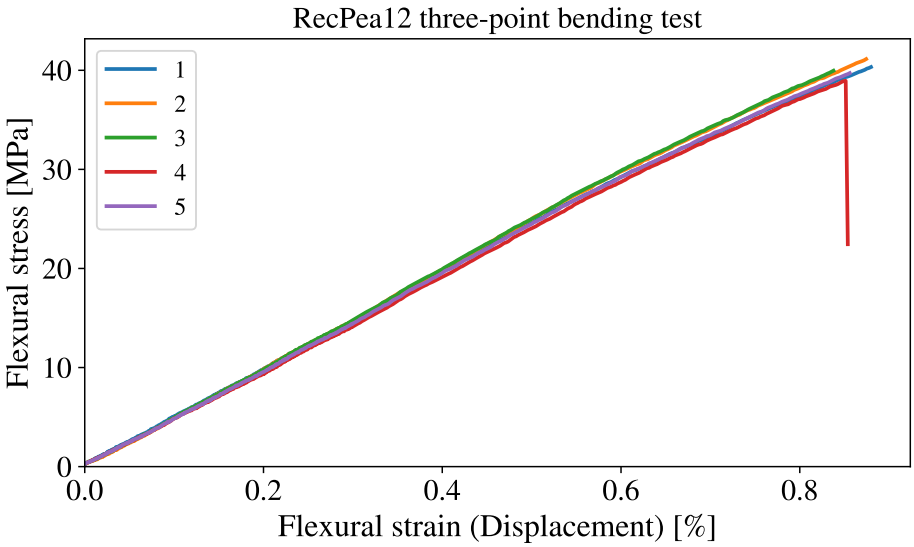


Figure B.2: Stress-strain curve of the Recell-Peach composite with $x_{\text{fibre}} = 0.1236$, $x_{\text{filler}} = 0.2198$ and $x_{\text{dry}} = 0.4653$

Sample	Flexural Strength (MPa)	Flexural Strain (%)	Flexural Modulus (MPa)	Force at maximum Flexure load (N)	Displacement at Break (mm)
1	43.8	0.98	4880	144.34	1.57
2	46.6	1	5020	152.51	1.7
3	47	1	4960	147.79	1.79
4	38.9	0.85	4680	125.12	1.51
5	46.8	1.1	4810	150.48	1.85
Minimum	38.9	0.85	4680	125.12	1.51
Maximum	47	1.1	5020	152.51	1.85
Mean	44.6	0.99	4870	144.05	1.68
S.D.	3.43	0.08	131.38	11.02	0.14

Table B.5: Three-point bending test data of the Recell-Peach composite with $x_{\text{fibre}} = 0.1236$, $x_{\text{filler}} = 0.2198$ and $x_{\text{dry}} = 0.4653$

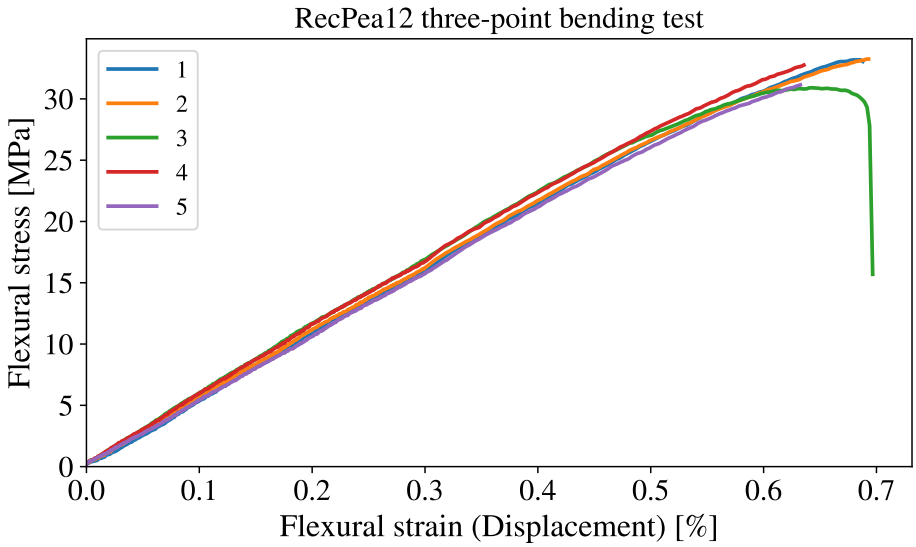


Figure B.3: Stress-strain curve of the Recell-Peach composite with $x_{\text{fibre}} = 0.0925$, $x_{\text{filler}} = 1.0000$ and $x_{\text{dry}} = 0.5688$

Sample	Flexural Strength (MPa)	Flexural Strain (%)	Flexural Modulus (MPa)	Force at maximum Flexure load (N)	Displacement at Break (mm)
1	33.2	0.69	5500	108.05	1.78
2	33.4	0.7	5480	109.31	1.92
3	30.9	0.64	5610	101.77	1.76
4	33.1	0.65	5620	101.2	1.86
5	31.4	0.64	5300	95.21	2.14
Minimum	30.9	0.64	5300	95.21	1.76
Maximum	33.4	0.7	5620	109.31	2.14
Mean	32.4	0.66	5500	103.11	1.89
S.D.	1.15	0.03	130.09	5.72	0.15

Table B.6: Three-point bending test data of the Recell-Peach composite with $x_{\text{fibre}} = 0.0925$, $x_{\text{filler}} = 1.0000$ and $x_{\text{dry}} = 0.5688$

C

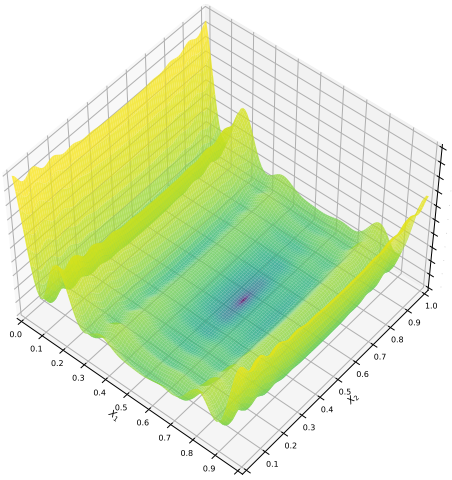
Support information for data-driven optimization

C.1. Analytical equations of optimization problems

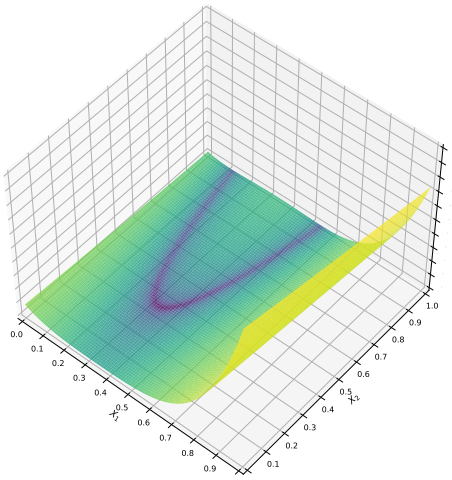
The analytical equations for the three sources of optimization benchmark problems mentioned in section 4.1 are discussed in greater detail.

C.1.1. Well-known optimization benchmark functions

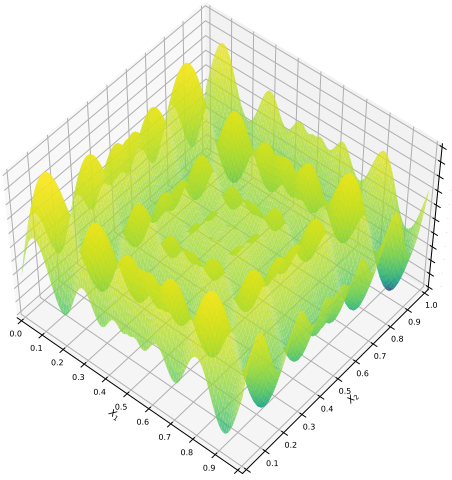
Table C.1 displays the analytical equations and search domain of several well-known optimization benchmark functions. Figure C.1 shows a smooth two-dimensional response surface for six of the functions mentioned in table C.1.



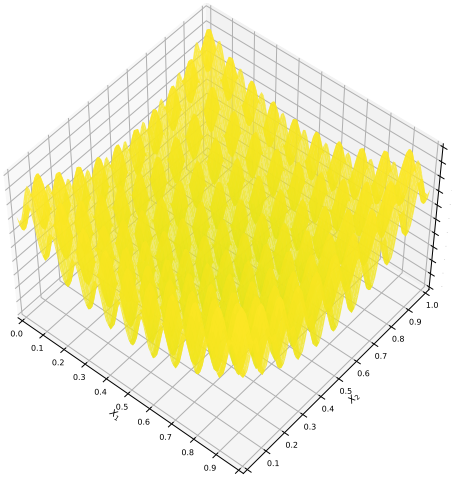
(a) Levy function.



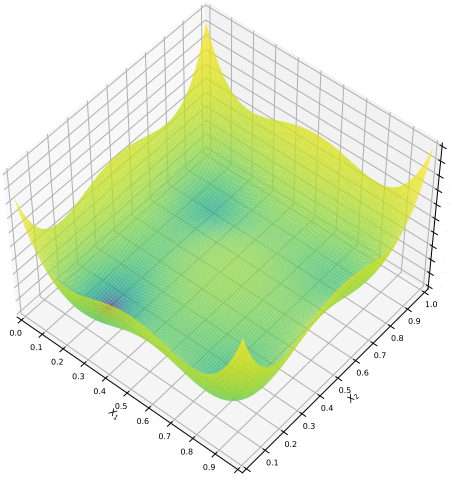
(b) Rosenbrock function.



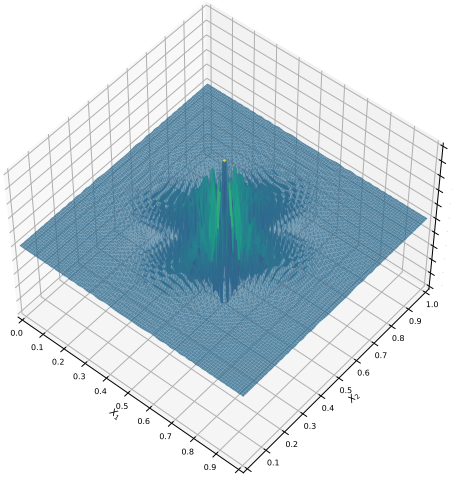
(c) Schwefel function.



(d) Rastrigin function.



(e) Styblinski function.



(f) Schaffer's F6 function.

Figure C.1: 2-dimensional response surfaces of several well-known benchmark functions. The input parameters and response surface have been normalized.

name	function	domain
Ackley	$f_2 = -a\exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - \exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d}\cos(cx_i)}\right) + a + \exp(1)$	$[-40, 40]^d$
Rosenbrock	$f_3 = \sum_{i=1}^{d-1}[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	$[-5, 10]^d$
Schwefel	$f_4 = 418.9829d - \sum_{i=1}^{d} x_i \sin(\sqrt{\lvert x_i\rvert})$	$[-500, 500]^d$
Rastrigin	$f_5 = 10d + \sum_{i=1}^{d}[x_i^2 - 10\cos(2\pi x_i)]$	$[-5.12, 5.12]^d$
Easom	$f_6 = -\cos(x_1)\cos(x_2)\exp(-(x_1 - \pi)^2 - (x_2 - \pi)^2)$	$[-100, 100]^2$
Styblinski	$f_7 = \frac{1}{2}\sum_{i=1}^{d}(x_i^4 - 16x_i^2 + 5x_i)$	$[-5, 5]^d$
Branin	$f_8 = (x_2 - 5.1x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos(x_1) + 10$	$[-5, 10], [0, 15]$
Schaffer F6	$f_9 = 0.5 + \frac{\sin^2(\sqrt{x^2+y^2}) - 0.5}{[1+0.001(x^2+y^2)]^2}$	$[-100, 100]^d$
Beale	$f_{10} = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2$	$[-4.5, 4.5]^2$
Leon	$f_{11} = 100(x_2 - x_1^3)^2 + (1 - x_1)^2$	$[-5, 5]^2$
Ackley N. 2	$f_{12} = -200\exp(-0.2\sqrt{x_1^2 + x_2^2})$	$[-4, 4]^2$
Bohachevsky	$f_{13} = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$	$[-100, 100]^2$
Matyas	$f_{14} = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2$	$[-10, 10]^2$
Zakharov	$f_{15} = \sum_{i=1}^{d} x_i^2 + (\sum_{i=1}^{d} 0.5ix_i)^2 + (\sum_{i=1}^{d} 0.5ix_i)^4$	$[-5, 10]^d$
McCormick	$f_{16} = \sin(x_1 + x_2) + (x_1 - x_2)^2 - 1.5x_1 + 2.5x_2 + 1$	$[-3, 4]^2$

Table C.1: Analytical form of several well-known test functions. The dimensionality is given by the variable d.

C.1.2. Rönkkönen parametrized multimodal functions

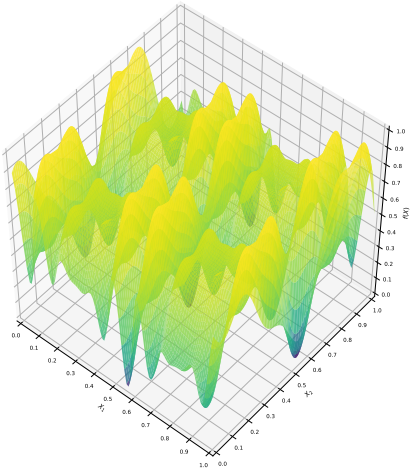
Table C.2 displays the analytical equations and search domain of the Rönkkönen parametrized multimodal functions.

The parameters $\vec{G}$, $\vec{L}$, α, $\vec{p}$, $\vec{v}$, $\vec{h}$, $\vec{r}$ and the number of minima q for the quadratic and plate family functions q are defined according to the random seed of the optimization problem. The range of each parameter is given in table C.3.

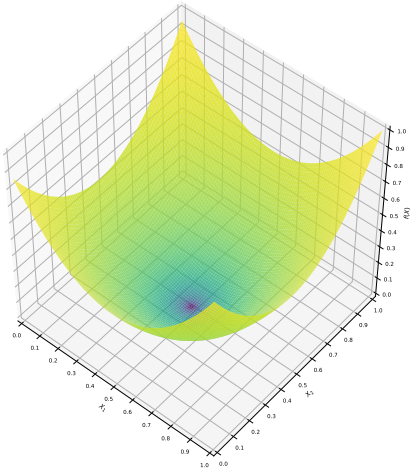
Figure C.2 shows a smooth two-dimensional response surface for each of the six family functions.

name	function	domain
Cosine family	$f_{\cos} = \frac{\sum_{i=1}^{d} -\cos((G_i-1)2\pi x_i) - \alpha\times\cos((G_i-1)2\pi L_i x_i)}{2d}$	$[0, 1]^d$
Bowl family	$f_{\text{bowl}} = \sum_{i=0}^{d} 20x_i^2$	$[0, 1]^d$
Cosine-bowl family	$f_{\text{cosbowl}} = 5f_{cos} + f_{bowl}$	$[0, 1]^d$
Quadratic family	$f_{\text{quad}} = \min((\vec{x} - \vec{p}_i)^\alpha + \vec{v}_i)$	$[0, 1]^d$
Plate family	$f_{\text{plate}} = \min((\vec{x} - \vec{p}_i) + \vec{v}_i)$	$[0, 1]^d$
Steep family	$f_{\text{steep}} = \begin{cases} h_i\left[1 - \left(\frac{d(\vec{x},i)}{r_i}\right)^\alpha\right], & \text{if } d(\vec{x}, i) \le r_i \\ 0, & \text{otherwise} \end{cases}$	$[0, 1]^d$

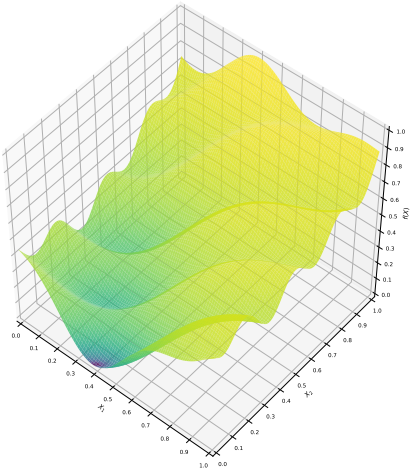
Table C.2: Analytical form of modified Rönkkönen test functions [156].



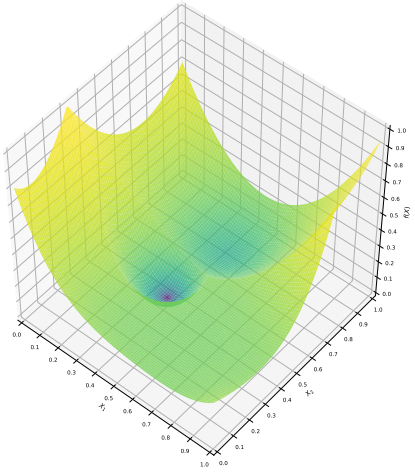
(a) Cosine family function.



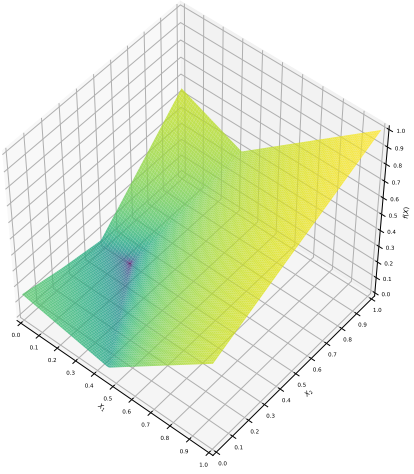
(b) Bowl family function.



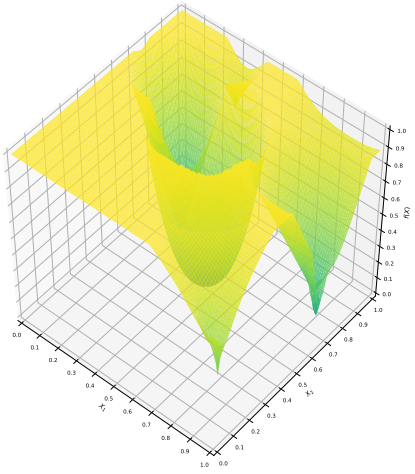
(c) Cosine-bowl family function.



(d) Quadratic family function.



(e) Plate family function.



(f) Steep family function.

Figure C.2: 2-dimensional response surfaces of the Rönkkönen test functions. The input parameters and response surface have been normalized.

parameter	symbol	form	lower bound	upper bound
Amount of global optima (cosine family)	$\vec{G}$	discrete	1	5
Amount of local optima (cosine family)	$\vec{L}$	discrete	0	7
Shape-factor of minima	α	continuous	0.0	1.0
Positions of minima	$\vec{p}$	continuous	0.0	1.0
Depths of minima	$\vec{v}$	continuous	-5.0	-3.0
Depths of minima (steep family)	$\vec{h}$	continuous	-100	-3
Area of attraction (steep family)	$\vec{r}$	continuous	0.05	0.3
Number of minima (quadratic family)	q	discrete	3	10

Table C.3: Boundaries of the parameters of the Rönkkönen parametrized multimodal functions.

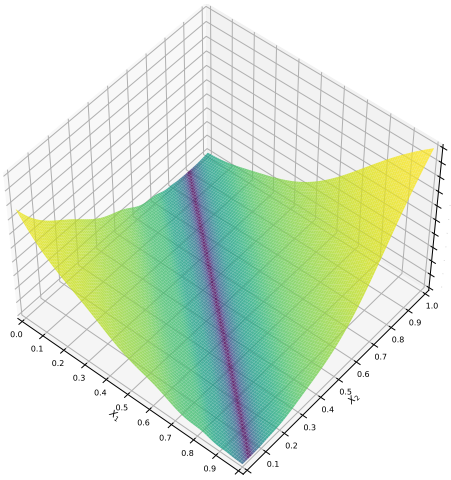
	No.	Functions
Unimodal functions	1	Sphere Function
	2	Rotated High Conditioned Elliptic Function
	3	Rotated Bent Cigar Function
	4	Rotated Discus Function
	5	Different Powers Function
Basic multimodal functions	6	Rotated Rosenbrock's Function
	7	Rotated Schaffers F7 Function
	8	Rotated Ackley's Function
	9	Rotated Weierstrass Function
	10	Rotated Griewank's Function
	11	Rastrigin's Function
	12	Rotated Rastrigin's Function
	13	Non-Continuous Rotated Rastrigin's
	14	Schwefel's Function
	15	Rotated Schwefel's Function
	16	Rotated Katsuura Function
	17	Lunacek Bi_Rastrigin Function
	18	Rotated Lunacek Bi_Rastrigin
	19	Expanded Griewank's plus Rosenbrock's
	20	Expanded Schaffer's F6
Composition functions	21	Composition Function 1
	22	Composition Function 2
	23	Composition Function 3
	24	Composition Function 4
	25	Composition Function 5
	26	Composition Function 6
	27	Composition Function 7
	28	Composition Function 8

Table C.4: Summary of the 28 CEC'13 test functions. The functions are categorized as unimodal, basic multimodal or composition functions [157].

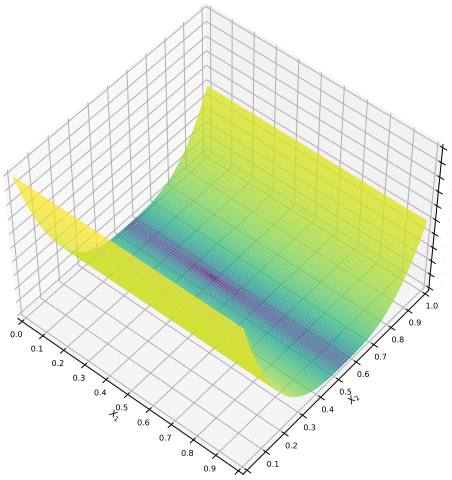
C.1.3. CEC 2013 competition benchmark functions

The 28 single-objective benchmark functions present in the CEC 2013 benchmark competition are categorized as unimodal, basic multimodal or composition functions [157]. The search range of all the functions is defined between $[-100, 100]^d$ for each dimension d.

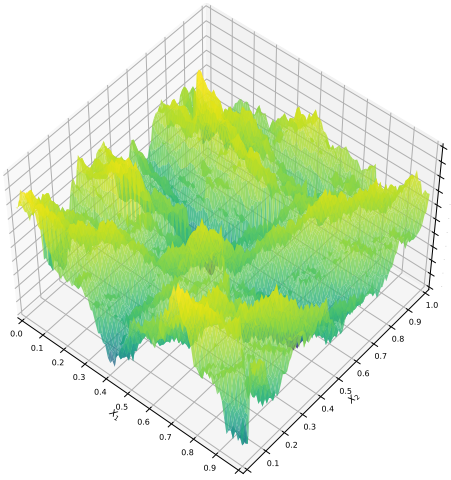
Figure C.3 shows a smooth two-dimensional response surface for six of the functions mentioned in table C.4. The analytical equations of these benchmark functions can be found in the original paper [157].



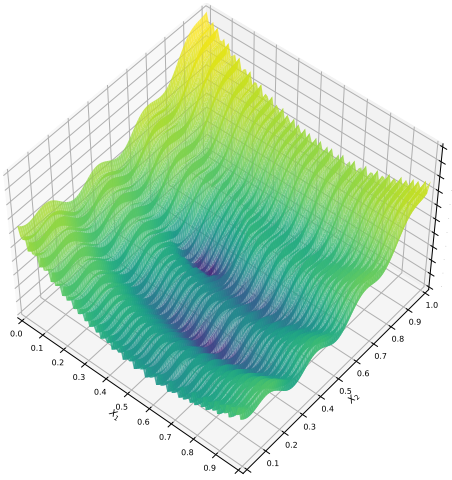
(a) Rotated bent cigar function.



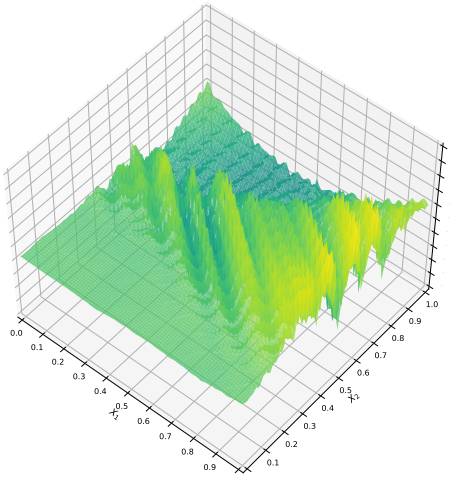
(b) Different powers function.



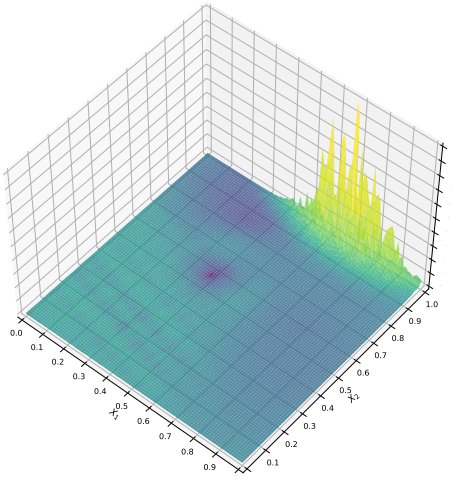
(c) Rotated Weierstrass function.



(d) Lunacek Bi_Rastrigin function.



(e) Composition function 4.



(f) Composition function 8.

Figure C.3: 2-dimensional response surfaces of several CEC 2013 benchmark functions. The input parameters and response surface have been normalized.

C.2. Implementations of selected algorithms

The set of selected meta-heuristics implemented in the data-driven heuristic decision strategy are written in Python. The variants used and the hyper-parameters will be presented in greater detail in the upcoming sections.

C.2.1. Covariance Matrix Adaptation Evolution Strategy

The Covariance Matrix Adaptation Evolution Strategy used is the `pygmo.cmaes` implementation of the scientific Python library `pygmo` [173]. The CMAES variant used is the classic interpretation described in Hansen's review paper [123]. The hyper-parameters are equal to the default strategy parameters. The initial step size σ_0 is equal to 0.5. Because a normal distribution determines the position of new solutions, solutions may be sampled outside the box-constrained boundaries. A restriction is set, so only solutions within the box-constrained boundaries are proposed to narrow our search for an optimum within the given limits.

Algorithm C.1 Covariance Matrix Adaptation Evolution Strategy (CMAES)

Require: $\lambda \leftarrow 4 + 3\ln(d)$: Population size
Require: $\mu, w_{i=1\dots\mu}, c_\sigma, d_\sigma, c_c, \mu_{\text{cov}}, c_{\text{cov}}$: Hyper-parameters
Require: $\sigma_0 = 0.5$: Initial step size
Require: $f(\vec{x})$: Objective function with parameters $\vec{x}$
Require: $\vec{x}_{0\dots i}$: Initial parameter guesses
$t \leftarrow 0$ (Initialize time-step)
$\vec{m}_0 \in \mathbb{R}^d$: Initialize distribution mean
$\vec{p}_0^{\,\sigma} \leftarrow 0$: Initialize step-size control
$\vec{p}_0^{\,c} \leftarrow 0$: Initialize covariance matrix adaptation
$\mathbf{C}_0 \leftarrow \mathbf{I}$: Covariance matrix initialization
while $t < t_{\max}$ **do**
$t \leftarrow t + 1$
for all candidates **do**
$\vec{x}_{t+1} \sim \mathbb{N}(\vec{m}_t, (\sigma_t)^2 \mathbf{C}_t)$ (Sample new population of search points)
end for
$\vec{m}_{t+1} = \sum_{i=1}^{\mu} \omega^i \vec{x}_t^{\,i}, \;\; \sum_{i=1}^{\mu} \omega^i = 1, \omega^i > 0$ (Selection of best candidates)
$\vec{p}_{t+1}^{\,\sigma} = (1 - c_c)\vec{p}_t^{\,\sigma} + \sqrt{c_\sigma(2 - c_\sigma)\mu_{\text{eff}}}\mathbf{C}_t \frac{-0.5\vec{m}_{t+1} - \vec{m}_t}{\sigma_t}$ (Step-size control)
$\vec{\sigma}_{t+1} = \vec{\sigma}_t \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{|\vec{p}_{t+1}^{\,\sigma}|}{\mathbb{E}|\mathbb{N}(O, \mathbf{I})|} - 1\right)\right)$ (Step-size control)
$\vec{p}_{t+1}^{\,c} = (1 - c_c)\vec{p}_t^{\,c} + h_{t+1}^{\sigma} + \sqrt{c_c(2 - c_c)\mu_{\text{eff}}}\frac{\vec{m}_{t+1} - \vec{m}_t}{\sigma_t}$ (Covariance matrix adaptation)
$\mathbf{C}_{t+1} = (1 - c_{\text{cov}})\mathbf{C}_t + \frac{c_{\text{cov}}}{\mu_{\text{cov}}} \vec{p}_{t+1}^{\,c}(\vec{p}_{t+1}^{\,c})^T + c_{\text{cov}}(1 - \frac{1}{\mu_{\text{cov}}}) \times \sum_{i=1}^{\mu} \omega^i \left(\frac{x_{t+1}^i - \vec{m}_t}{\sigma_t}\right)\left(\frac{x_{t+1}^i - \vec{m}_t}{\sigma_t}\right)^T$ (Covariance matrix adaptation [123])
end while
return $\vec{x}_t$ (Resulting parameters)

The classical variant is chosen with a static population size of $\lambda = 4+3\ln(d)$.

C.2.2. Generational Particle Swarm Optimization

Particle Swarm Optimization is also extracted from the scientific library `pygmo` [173], explicitly the `pygmo.pso_gen` implementation. This variant is mostly identical to the conventional version in the original paper [114]. However, it utilizes a constriction coefficient, introduced in Clerc's 2002 study [179]. This coefficient ω restricts the particles' velocity to increase indefinitely, assures convergence and thus eliminates the need to limit the velocity with an additional hyper-parameter $v_{\max}$ manually. The values of ω, ϕ_1 and ϕ_2 are 0.7298, 2.05 and 2.05, respectively. These are the default hyper-parameters used in `pygmo`. The canonical variant is selected, in which the random vectors R_{1t} and R_{2t} have random elements for all its components.

Algorithm C.2 Generational Particle Swarm Optimization (PSO-Gen)

Require: $\lambda \leftarrow 30$: Population size
Require: $\phi_1 = 2.05, \phi_1 = 2.05$: Social and cognitive components
Require: $\omega = 0.7298$: Constriction coefficient
Require: $f(\vec{x})$: Objective function with parameters $\vec{x}$
Require: $\vec{x}_{0\ldots i}$: Initial parameter guesses
  $\vec{v}_0 \leftarrow 0$ (Initialize all particle velocities)
  $t \leftarrow 0$ (Initialize time-step)
  $\vec{p} \leftarrow \vec{x}_0$ (Initialize local best positions)
  $\vec{g} \leftarrow \min(\vec{x}_0)$ (Initialize global best position)
  while $t < t_{\max}$ **do**
    $t \leftarrow t + 1$
    $R_{1t} \leftarrow \text{random}[0,1]^d$
    $R_{2t} \leftarrow \text{random}[0,1]^d$
    for all particles **do**
      if $f(\vec{x}_t^i) < f(\vec{p}^i)$ **then**
        $\vec{p}^i \leftarrow \vec{x}_t^i$ (Update local best position)
      end if
      if $f(\vec{x}_t^i) < f(\vec{g})$ **then**
        $\vec{g} \leftarrow \vec{x}_t^i$ (Update global best position)
      end if
    end for
    for all particles **do**
      $\vec{v}_{t+1}^i \leftarrow \omega(\vec{v}_t^i + \phi_1 R_{1t} \cdot (\vec{x}_t^i - \vec{p}_i) + \phi_2 R_{2t} \cdot (\vec{x}_t^i - \vec{g}))$ (Update velocity)
    end for
    for all particles **do**
      $\vec{x}_{t+1}^i \leftarrow \vec{x}_t^i + \vec{v}_t^i$ (Update particle position)
    end for
  end while
  return $\vec{x}_t$ (Resulting parameters)

Because our problem set also contains stochastic and deterministic problems, PSO must be resistant to noisy objective function values. The Generational Particle Swarm Optimization variant is identical to PSO, but updates each particle's velocities before new particle positions are computed (taking into consideration all updated particle velocities).

Several studies on the choice of population size of PSO concluded that for lower dimensional problems ($d < 10$), a swarm size of $\lambda = 30$ is appropriate. For higher dimensional problems ($d > 30$), increasing the population size above $\lambda = 50$ will not lead to better performance [174–176].

C.2.3. Adaptive Moment Estimate

The implementation of Adam's gradient-based algorithm is a self-implemented version based on the pseudo-code in the original paper of Kingma [116]. The step size is set to $\alpha = 10^{-2}$. The exponential moving average hyper-parameters β_1 and β_2 are 0.9 and 0.999, respectively. The small floating-point $\epsilon = 10^{-8}$ is introduced to counter a division by zero while updating the parameters.

As we assume that information about the derivative of the objective function is not known, the gradient $\frac{df}{dx}$ must be estimated. The numerical approximation of each derivative is made by central difference, according to equation C.1:

$$\frac{df}{dx} \approx \frac{f(x + dx) - f(x - dx)}{2dx} + O(dx^2) \tag{C.1}$$

For estimating the gradient by central difference, the objective value at $f(x + dx)$ and $f(x - dx)$ has to be obtained for each search-space dimension d. Hence, the overall cost of estimating the gradient results in $2d$.

As Adam is proposed as a single-solution optimizer in its original paper. However, for each update step, $2d$ additional function evaluations are required. Therefore, the population size will be set to $\lambda = 1 + 2d$, of which the population's first solution denotes the Adam iterative step and the remaining of the population the parameters requested from the gradient estimation.

Algorithm C.3 Adaptive Moment Estimate (Adam) [116]

Require: $\alpha = 10^{-2}$: Learning rate
Require: $\beta_1 = 0.9, \beta_2 = 0.999$: Exponential decay rates for the moment estimates
Require: $\epsilon = 10^{-8}$: Floating-point to counter division by zero
Require: $f(\vec{x})$: Objective function with parameters $\vec{x}$
Require: $\vec{x}_0$: Initial parameter guess
  $m_0 \leftarrow 0$ (Initialize 1st moment vector)
  $v_0 \leftarrow 0$ (Initialize 2nd moment vector)
  $t \leftarrow 0$ (Initialize time-step)
  while $t < t_{\max}$ **do**
    $t \leftarrow t + 1$
    for all candidates **do**
      $g_t^i \leftarrow \nabla_{\vec{x}} f_t(\vec{x}_{t-1}^i)$ (Estimate gradient at time-step t)
      $m_t^i \leftarrow \beta_1 \cdot m_{t-1}^i + (1 - \beta_1) \cdot g_t^i$ (Update biased first moment estimate)
      $v_t^i \leftarrow \beta_2 \cdot v_{t-1}^i + (1-\beta_2) \cdot (g_t^i)^2$ (Update biased second raw moment estimate)
      $\hat{m}_t^i \leftarrow m_t^i / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
      $\hat{v}_t^i \leftarrow v_t^i / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
      $\vec{x}_{t+1}^i \leftarrow \vec{x}_t^i - \alpha \cdot \hat{m}_t^i / \sqrt{\hat{v}_t^i + \epsilon})$ (Update parameters)
    end for
  end while
  return $\vec{x}_t$ (Resulting parameters)

C.2.4. Bayesian Optimization

The sequential model-based optimization library `GPyOpt` is used for constructing a Bayesian Optimization algorithm [131]. A surrogate model is created from a Gaussian Process with an RBF kernel. Subsequently, an 'Lower confidence bound' acquisition function is optimized with an 'L-BFGS' optimizer to propose the next iteration location.

Algorithm C.4 Bayesian Optimization

Require: $\lambda = 1$: Population size
Require: $f(\vec{x})$: Objective function with parameters $\vec{x}$
Require: $\vec{x}_0$: Initial parameter guess
Require: $M \leftarrow 20$: Number of inducing points
  $t \leftarrow 0$ (Initialize time-step)
  while $t < t_{\max}$ **do**
    $t \leftarrow t + 1$
    Construct GP prior surrogate model with RBF kernel from $\vec{x}_{0\ldots t}$
    Construct Lower confidence bound acquisition function $\mathrm{LCB}(\vec{x})$
    $\vec{x}_{t+1} \leftarrow \max_{\vec{x}} \mathrm{LCB}(\vec{x})$ (Maximize acquisition function with 'L-BFGS' optimizer)
  end while
  return $\vec{x}_t$ (Resulting parameters)

Unlike the bio-based composite optimization model, no parallelization technique such as the 'constant liar' method discussed in section 3.3 will be used, as this is not the standard method of implementing Bayesian Optimization. Bayesian Optimization will therefore be implemented as a single-solution optimizer.

C.2.5. Random Search

The random search algorithm is intended as a baseline for other heuristics. The individual elements of a new input vector are sampled through a naive random number generator.

Algorithm C.5 Random Search

Require: $f(\vec{x})$: Objective function with parameters $\vec{x}$
Require: $\vec{x}_{0...i}$: Initial parameter guesses
  $t \leftarrow 0$ (Initialize time-step)
  while $t < t_{\max}$ **do**
    $t \leftarrow t + 1$
    for all candidates **do**
      $\vec{x}_t^i \leftarrow \text{random}[0,1]^d$
    end for
  end while
  return $\vec{x}_t$ (Resulting parameters)

C.3. Experiments from 'learning to optimize' study

The logistic regression and robust linear regression problems are part of the benchmarking problems depicted in the 'Learning to optimize' paper from Li et al. [141]. In the following sections, their attributes and implementation in the data-driven heuristic framework are discussed.

C.3.1. Logistic regression

A logistic regression estimates the curve of a binary logistic model with a dependent variable y_i that has one of two possible values (0 or 1). The quality of fitting is determined by minimizing equation C.2:

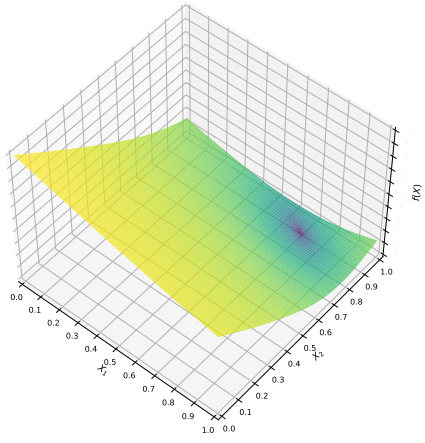
$$\min_{\vec{w},b} -\frac{1}{n}\sum_{i=1}^{n} y_i \log\sigma(\vec{w}^T\vec{x}_i + b) + (1-y_i)\log(1-\sigma(\vec{w}^T\vec{x}_i + b)) + \frac{\lambda}{2}\|\vec{w}\|_2^2 \quad \text{(C.2)}$$

A dimensionality of $d = 4$ is selected, and equation C.2 is minimized by altering the parameters $\vec{w}$ and $\vec{b}$. The vector $\vec{w} \in \mathbb{R}^{d-1}$ and $b \in \mathbb{R}$ denote the weight vector and bias respectively and are considered the input parameters. The parameter $\sigma(z)$ is described in equation C.3 and the coefficient on the regularizer $\lambda = 0.0005$ [141].

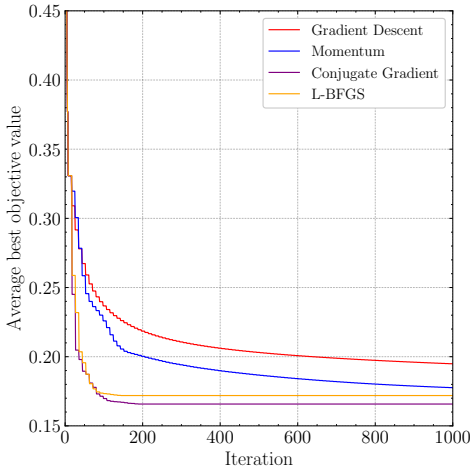
$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{C.3}$$

For each instance of the objective function, a dataset of 100 points $\vec{x}_{0\ldots100}$ is randomly sampled from two multivariate Gaussians with random means and covariances. Exactly half of the points are sampled from each multivariate Gaussian. The data-points sampled from the same Gaussian are labelled $y_i \in \{0, 1\}$ likewise, and instances from different Gaussians are assigned different labels [141].

As the framework presented in section 4.1 needs box-constraint boundaries, we have altered the logistic regression experiment by constraining $\vec{w}$ and b between $[-5.0, 5.0]^d$ for every dimension. These boundaries are selected so that the underlying calculations are not subjected to overflow. Figure C.4a shows a two-dimensional slice of the four-dimensional logistic regression problem. The results from one logistic regression problem for various local optimizers are plotted in figure C.4b.



(a) Response surface of a two-dimensional slice.



(b) Performance from various optimizers.

Figure C.4: Response surface and performance of various local optimizers on the logistic regression problem. The input parameters boundaries are normalized.

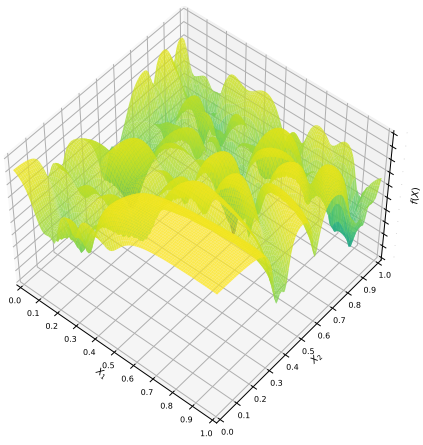
C.3.2. Robust linear regression

A linear regression is a linear approach to model the relationship of a scalar output y_i to a number of variables. A robust loss function is used to assess the quality of the fit. The Geman-McClure M-estimator is used for the parameter estimation. The objective is to minimize equation C.4:

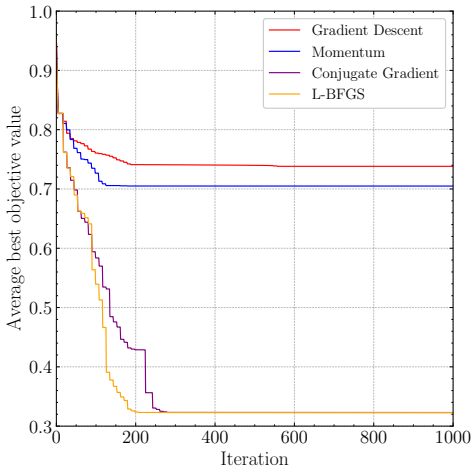
$$\min_{\vec{w},b} \frac{1}{n} \sum_{i=1}^{n} \frac{(y_i - \vec{w}^T \vec{x}_i - b)^2}{c^2 + (y_i - \vec{w}^T \vec{x}_i - b)^2} \tag{C.4}$$

Again, a dimensionality of $d = 4$ is selected, and equation C.4 is minimized by altering the parameters $\vec{w}$ and $\vec{b}$. The vector $\vec{w} \in \mathbb{R}^{d-1}$ and $b \in \mathbb{R}$ denote the weight vector and bias respectively and are considered the input parameters. The value $c = 1$ is used for this experiment. 25 random samples $\vec{x}_{0\ldots25}$ are drawn from one of four multivariate Gaussian, each with a randomized mean and identity covariance matrix. The labels $y_{0\ldots25}$ are generated by projecting the samples along the corresponding random mean vector. Subsequently, a random bias and Gaussian noise ($\sim \mathbb{N}(0, 1)$) is added [141].

Likewise, box-constrained boundaries are added to the input parameters $\vec{w}$ and b equal to $[-10.0, 10.0]^d$. Figure C.5a shows a two-dimensional slice of the four-dimensional robust linear regression problem. The results from one linear regression problem for various local optimizers are plotted in figure C.5b.



(a) Response surface of a two-dimensional slice.



(b) Performance from various optimizers.

Figure C.5: Response surface and performance of various local optimizers on the robust linear regression problem. The input parameters boundaries are normalized.

C.4. Confusion matrices and decision bar graphs.

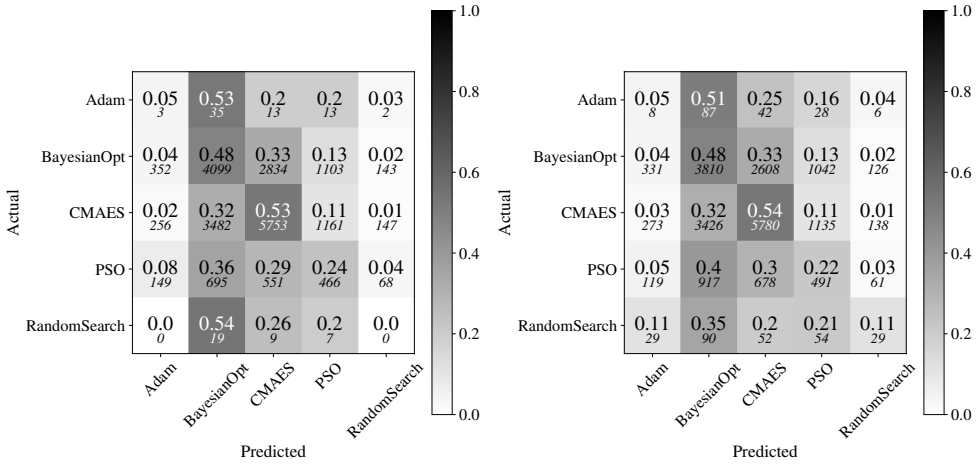
C.4.1. Confusion matrices

A confusion matrix is a table the allows the visualization of multi-class classification. Each row of the square matrix represents the actual target classes' labels, while the columns denote the class predictions. The diagonal elements contain the true positive (tp) and true negative predictions (tn), whereas the upper and lower triangular parts of the matrix denote the false positive (fp) and false negative (fn) predictions.

Each element of the confusion matrix is occupied by a number between 0 and 1,

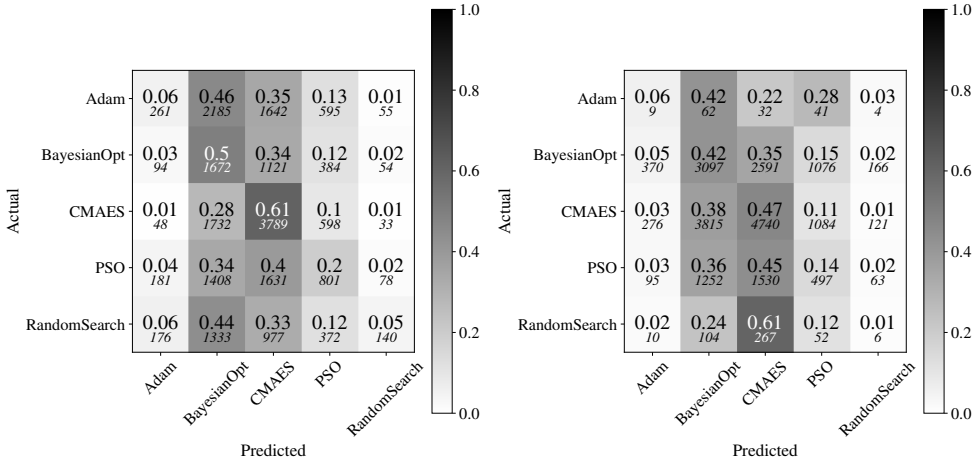
which denotes the fraction of predictions of the total number of target labels. The smaller number under this normalized accuracy denotes the number of samples that are predicted in this manner.

Online training set



(a) k-NN one-versus-one.

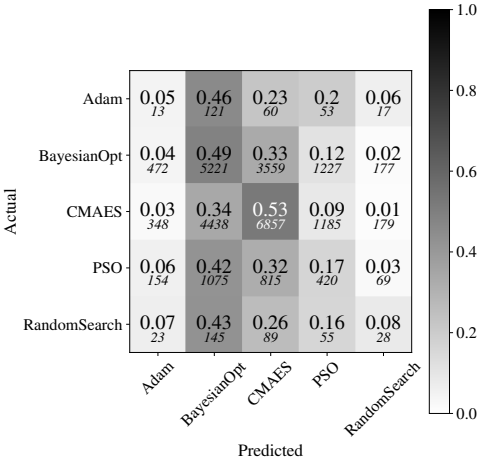
(b) k-NN one-versus-all.



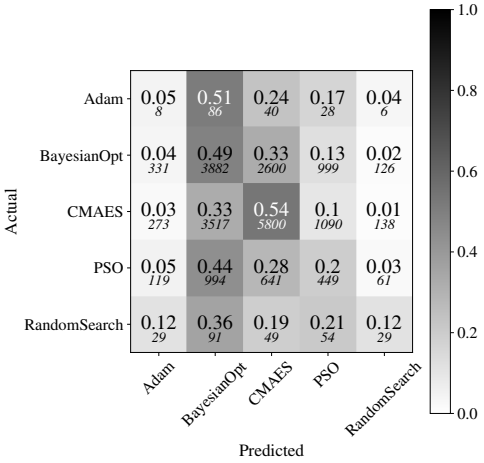
(c) C-Support Vector Classification.

(d) AdaBoost

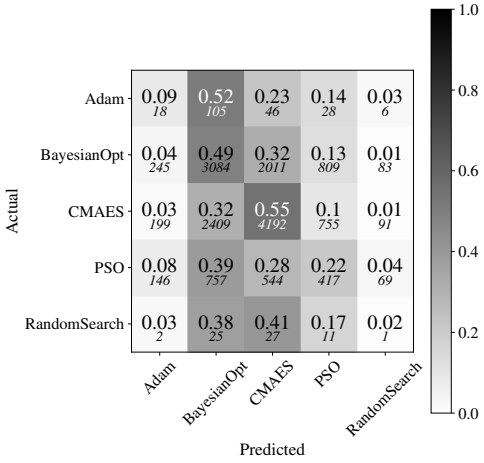
Figure C.6: Online training confusion matrices for the data-driven heuristic decision strategies with varying classifier. The horizontal axis denotes the predicted labels and the vertical axis denotes the actual heuristic decision.



(a) 4 windows with 125 iterations each.



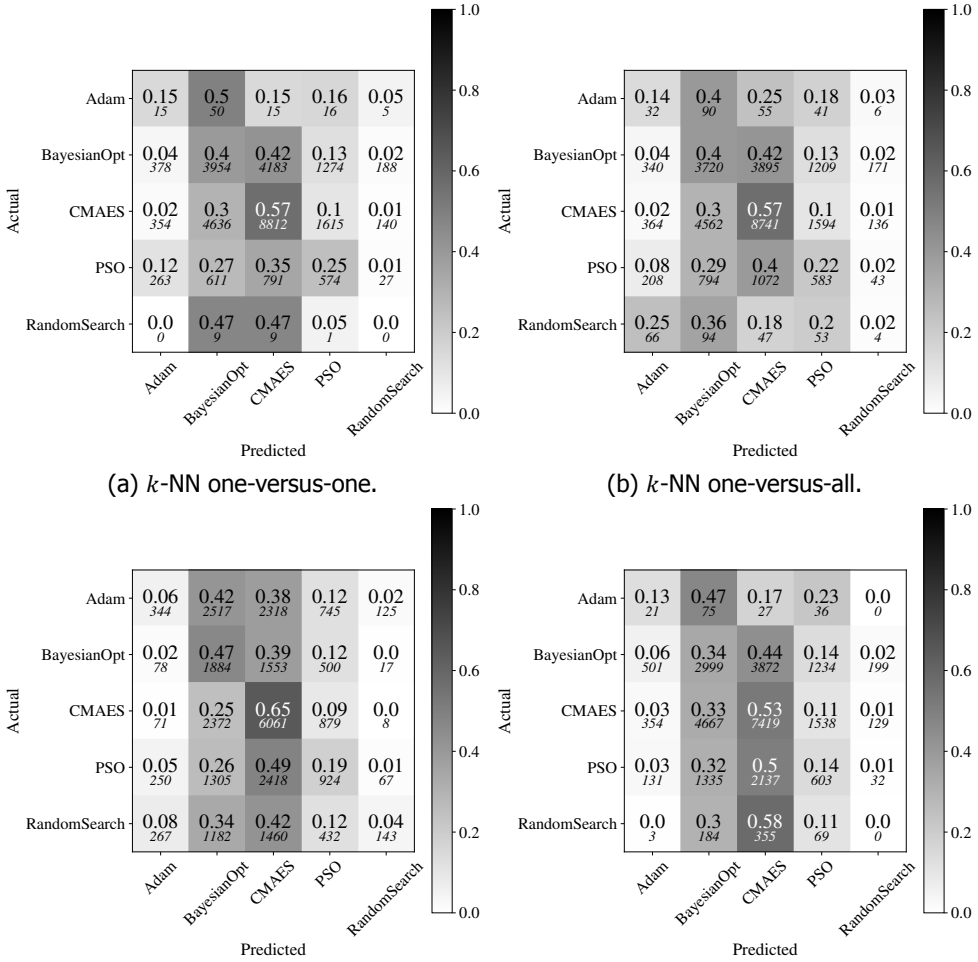
(b) 5 windows with 100 iterations each.



(c) 6 windows with 75 iterations each.

Figure C.7: Online training confusion matrices for the data-driven heuristic decision strategies with varying window sizes. The horizontal axis denotes the predicted labels and the vertical axis denotes the actual heuristic decision.

Online testing set



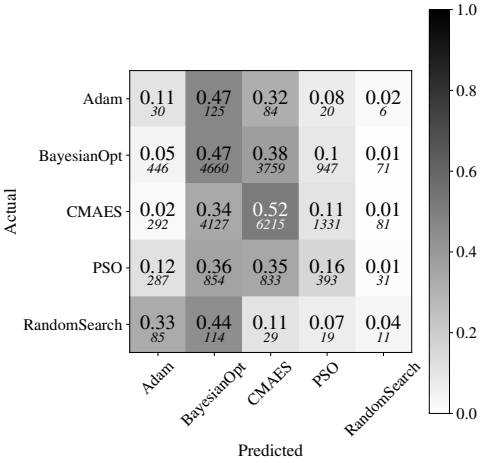
(a) k-NN one-versus-one.

(b) k-NN one-versus-all.

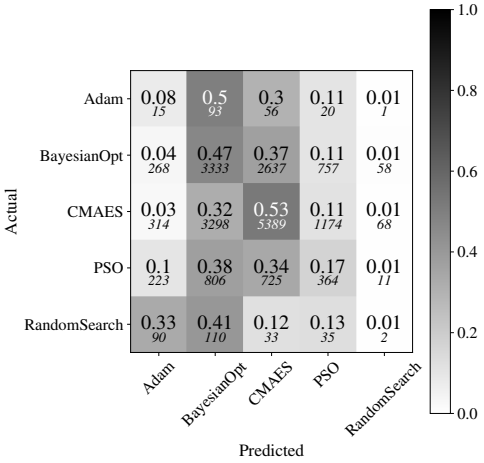
(c) C-Support Vector Classification.

(d) AdaBoost

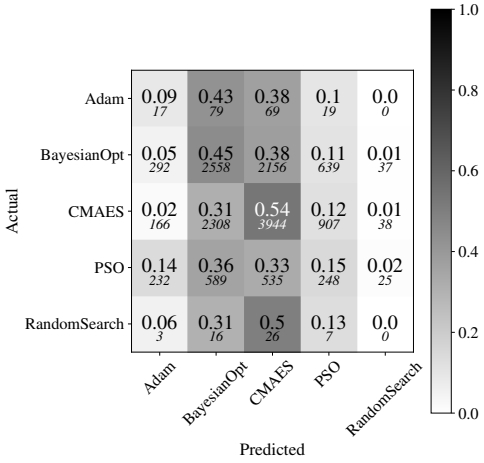
Figure C.8: Online testing confusion matrices for the data-driven heuristic decision strategies with varying classifier. The horizontal axis denotes the predicted labels and the vertical axis denotes the actual heuristic decision.



(a) 4 windows with 125 iterations each.



(b) 5 windows with 100 iterations each.

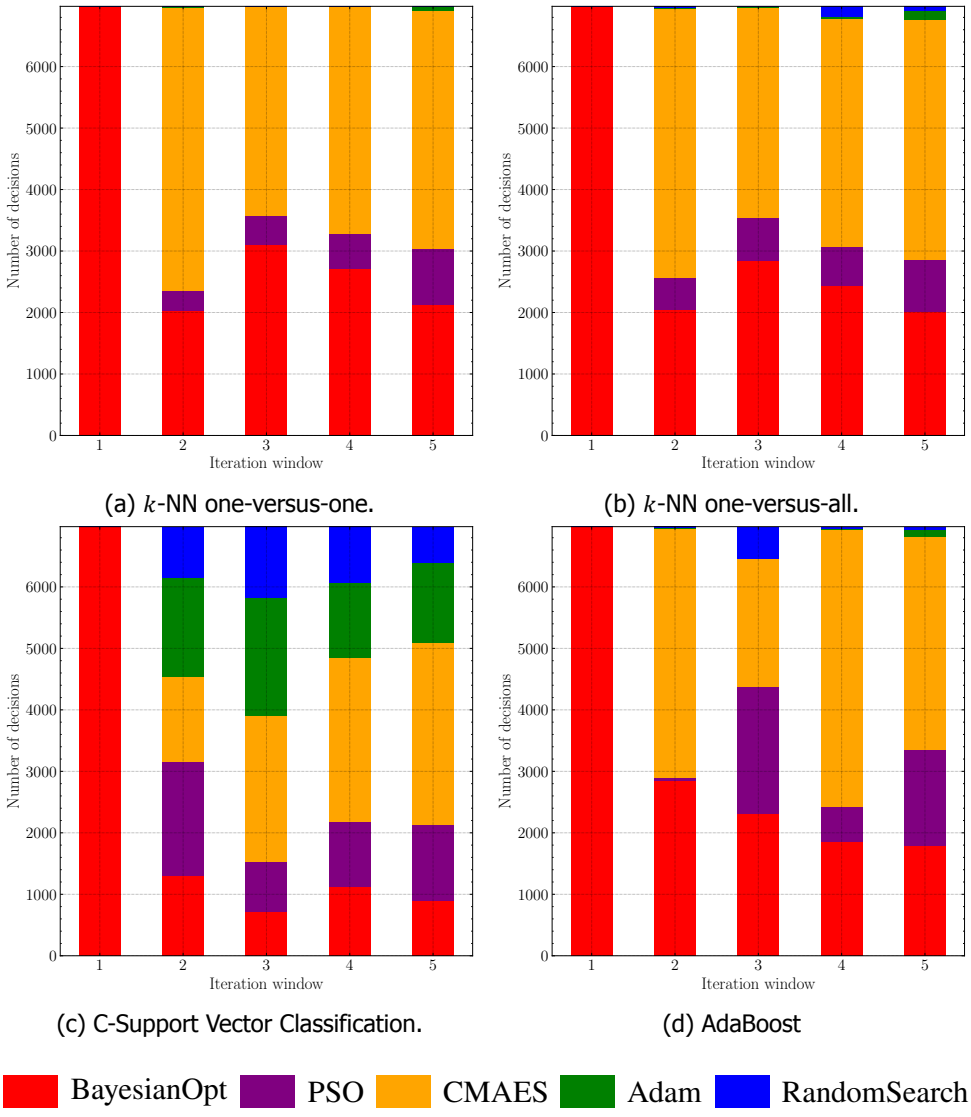


(c) 6 windows with 75 iterations each.

Figure C.9: Online testing confusion matrices for the data-driven heuristic decision strategies with varying window sizes. The horizontal axis denotes the predicted labels and the vertical axis denotes the actual heuristic decision.

C.4.2. Decision bar graphs

Figures C.10 and C.11 show the number of occurrences of each meta-heuristic in the heuristic strategy for each iteration window for the generated problem set.



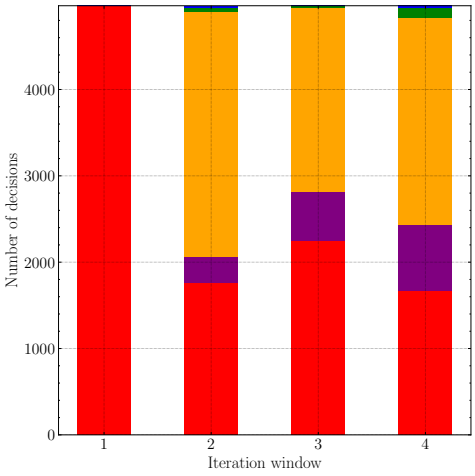
(a) k-NN one-versus-one.

(b) k-NN one-versus-all.

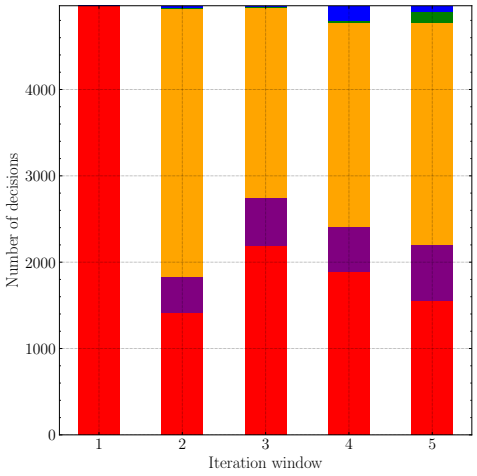
(c) C-Support Vector Classification.

(d) AdaBoost

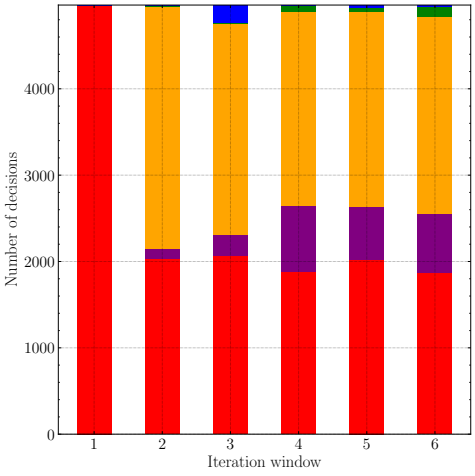
Figure C.10: Number of occurrences for each meta-heuristic in each iteration window. The decisions are displayed for the four different classifiers.



(a) 4 windows with 125 iterations each.



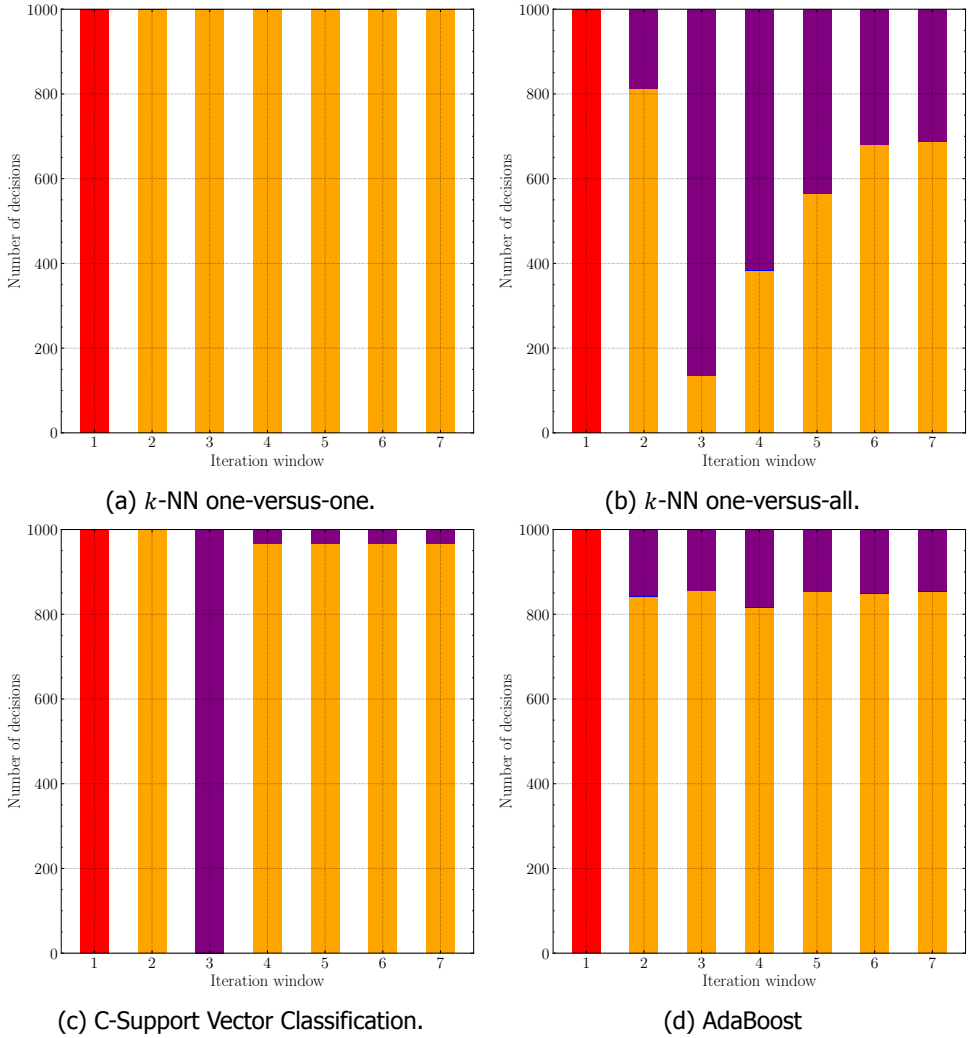
(b) 5 windows with 100 iterations each.



(c) 6 windows with 75 iterations each.

Figure C.11: Number of occurrences for each meta-heuristic in each iteration window. The decisions are displayed for 4, 5 and 6 iteration windows.

Figures C.12 and C.13 show the number of occurrences of each meta-heuristic in the heuristic strategy for each iteration window for the logistic regression and the robust linear regression problem respectively.



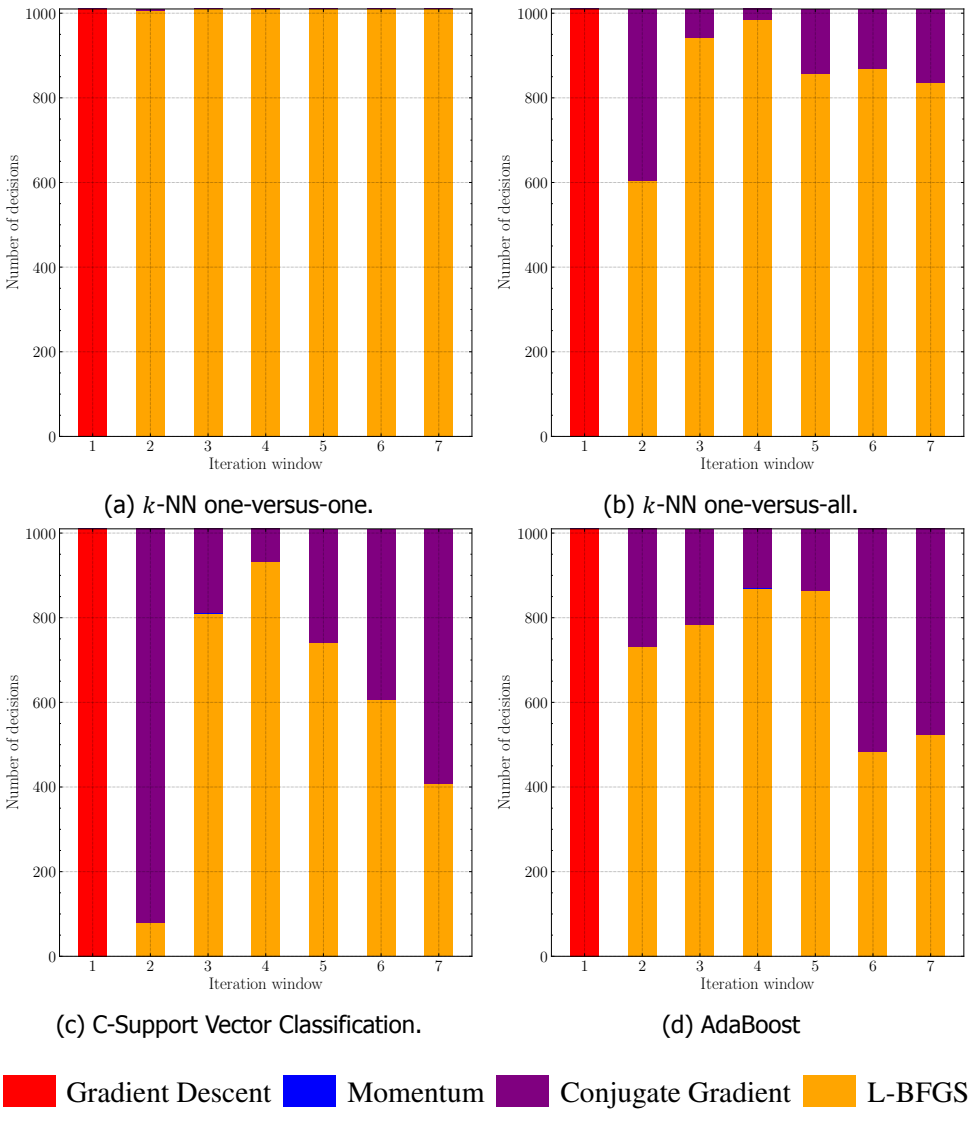
(a) k-NN one-versus-one.

(b) k-NN one-versus-all.

(c) C-Support Vector Classification.

(d) AdaBoost

Figure C.12: Number of occurrences for each meta-heuristic in each iteration window for the logistic regression problem.



(a) k-NN one-versus-one.

(b) k-NN one-versus-all.

(c) C-Support Vector Classification.

(d) AdaBoost

Figure C.13: Number of occurrences for each meta-heuristic in each iteration window for the robust linear regression problem.