Carl von Ossietzky
Universität
Oldenburg

Fakultät II – Informatik, Wirtschafts- und Rechtswissenschaften
Department für Informatik

Dynamic communication topologies for distributed energy system optimization heuristics

Von der Fakultät für Informatik, Wirtschafts- und Rechtswissenschaften
der Carl von Ossietzky Universität Oldenburg
zur Erlangung des Grades und Titels

Doktorin der Naturwissenschaften (Dr. rer. nat.)

angenommene Dissertation

von Stefanie Isabel Holly

geboren am 25. Oktober 1989 in Garmisch-Partenkirchen

1. Gutachter Prof. Dr.-Ing. Astrid Nieße
Department für Informatik
Carl von Ossietzky Universität Oldenburg

2. Gutachter Prof. Dr.-Ing. Sven Tomforde
Institut für Informatik
Christian-Albrechts-Universität zu Kiel

Tag der Disputation: 17. August 2023

Abstract

Power system operation involves a wide range of complex optimization problems that require the coordination of power generation and consumption across a large number of distributed units. For various reasons, such as scalability, robustness, or data privacy, a decentralized system is advantageous for controlling these processes. Self-organized agent-based control systems are a viable approach for this purpose. Thanks to special capabilities such as self-optimization or self-healing, they can flexibly adapt to changing environments. The distributed optimization problems that such control systems have to solve in energy systems are usually very complex, i.e., nonlinear, high-dimensional, and with a high degree of interdependencies among the decision variables. Using optimization heuristics instead of exact solution methods is therefore often a practical approach. For the application in distributed control systems, these optimization heuristics must operate in a distributed manner as well.

The communication or exchange topology, which determines the direct communication links between agents, is an important hyperparameter for such parallel and distributed optimization heuristics. It has a strong impact on the dissemination of information in the system and thus influences the degree of exploration and exploitation of the distributed search space. The balance between these two properties significantly affects the performance of optimization heuristics in terms of solution quality, computational effort and the communication overhead. Furthermore, each agent controls only a subset of the decision variables. Thus, solving the global problem by making choices for all decision variables requires cooperation among agents.

This makes the choice of the communication topology an important design aspect for such distributed optimization heuristics. In addition, the most beneficial degree of exploration and exploitation changes over the course of the optimization process. Runtime parameter tuning is a common method for improving algorithmic performance. Adapting the communication topology is therefore a promising strategy for improving the performance of distributed optimization heuristics. In the present work, a runtime adaptation of the communicating topology has been developed. Starting from an initial small-world topology, this method increases or decreases the connectivity of the topology at a configurable rate. The combination of the initial topology with the runtime adaptation is termed *Communication Topology Variant (CTV)*. The application of CTVs shows significant effects on the performance of the distributed optimization algorithms.

The characteristics of the optimization problem determine which type of CTV is beneficial. Thus, for the selection of an appropriate CTV, the properties of the distributed optimization problem must first be determined. A common approach for this is *Fitness Landscape Analysis (FLA)*, which includes a number of techniques typically based on drawing samples from a search space and then computing metrics based on the position of these samples, their fitness values, and neighborhood metrics. In this work, a method for spatially distributed execution of FLA has been developed. The resulting FLA metrics were successfully used as a basis for selecting appropriate CTVs. Machine learning models were trained to make predictions for various performance dimensions, such as solution quality, computational effort, and communication overhead, given the FLA metrics and CTV parameters. Based on these predictions, a problem-specific CTV can be selected that matches a predefined prioritization of performance dimensions to improve the optimization process in a targeted manner.

Zusammenfassung

Der Betrieb von Energiesystemen umfasst eine Vielzahl komplexer Optimierungsprobleme, die die Koordination von Energieerzeugung und -verbrauch einer großen Anzahl verteilter Anlagen erfordern. Verschiedene Aspekte wie Skalierbarkeit, Robustheit oder Datenschutz sprechen für ein dezentrales System zur Steuerung dieser Prozesse. Selbstorganisierende agentenbasierte Steuerungssysteme sind hierfür ein geeigneter Ansatz, da sie sich durch besondere Fähigkeiten wie Selbstoptimierung oder Selbstheilung flexibel an sich verändernde Umgebungen anpassen können. Die verteilten Optimierungsprobleme, die solche Steuerungssysteme in Energiesystemen lösen müssen, sind in der Regel sehr komplex, d.h. nicht linear, hochdimensional und mit einem hohen Grad an Abhängigkeiten zwischen den Entscheidungsvariablen. Aus diesem Grund lassen sie sich in der Praxis häufig mit Hilfe von Optimierungsheuristiken lösen, die jedoch für den Einsatz in verteilten Kontrollsystemen auch räumlich verteilt funktionieren müssen.

Die Kommunikationstopologie bestimmt die direkten Kommunikationsverbindungen zwischen Agenten und ist ein wichtiger Hyperparameter für solche parallelen und verteilten Optimierungsheuristiken. Sie hat einen starken Einfluss auf die Informationsverbreitung im System und damit auf den Grad der *Exploration* und *Exploitation* des verteilten Suchraums. Für die Leistungsfähigkeit von Optimierungsheuristiken in Bezug auf Lösungsqualität, Rechen- und Kommunikationsaufwand ist die Balance zwischen diesen beiden Eigenschaften von entscheidender Bedeutung. Jeder Agent hat dabei nur die Kontrolle über eine Teilmenge der Entscheidungsvariablen. Die Lösung des Gesamtproblems und die damit verbundene Festlegung aller Entscheidungsvariablen erfordert daher die Kooperation der Agenten.

Dies macht die Wahl der Kommunikationstopologie zu einem wichtigen Designaspekt für solche verteilten Optimierungsheuristiken. Darüber hinaus ändert sich im Laufe des Optimierungsprozesses, welcher Grad an *Exploration* und *Exploitation* am vorteilhaftesten ist. Eine gängige Methode zur Verbesserung der algorithmischen Leistungsfähigkeit ist die Anpassung von Hyperparametern zur Laufzeit. Daher ist die Adaption der Kommunikationstopologie zur Laufzeit eine vielversprechende Strategie, um die Performance von verteilten Optimierungsheuristiken zu verbessern. In dieser Arbeit wurde eine Methode zur Anpassung der Kommunikationstopologie zur Laufzeit entwickelt. Ausgehend von einer Small-World-Topologie wird die Konnektivität der Topologie mit einer konfigurierbaren Geschwindigkeit erhöht oder verringert. Die Kombination aus initialer Topologie und Laufzeitanpassung wird als *Kommunikationstopologievariante (CTV)* definiert. Die Anwendung dieser Adaptionsmethode zeigt signifikante Auswirkungen auf die Performance von verteilten Optimierungsalgorithmen.

Welche Art von CTV vorteilhaft ist, hängt von den Eigenschaften des Optimierungsproblems ab. Um eine geeignete CTV zu wählen, müssen daher zunächst die Eigenschaften des verteilten Optimierungsproblems bestimmt werden. Ein gebräuchlicher Ansatz hierfür ist die *Fitnesslandschaftsanalyse (FLA)*. Sie umfasst eine Reihe von Techniken, die typischerweise auf der Entnahme von Stichproben aus einem Suchraum und der anschließenden Berechnung von Metriken beruhen. Die Metriken basieren auf der Position dieser Stichproben, ihren Fitnesswerten und den Nachbarschaftsbeziehungen zwischen ihnen. In dieser Arbeit wurde eine Methode zur räumlich verteilten Ausführung von FLA entwickelt. Die resultierenden FLA-Metriken wurden erfolgreich als Grundlage für die Auswahl geeigneter CTVs verwendet. Dazu wurden Machine-Learning-Modelle auf Basis der FLA-Metriken und der Parameter der CTV trainiert, um Vorhersagen für verschiedene Performancedimensionen wie Lösungsqualität, Rechen- und Kommunikationsaufwand zu treffen. Anhand dieser Vorhersagen kann eine problemspezifische CTV ausgewählt werden, die einer vordefinierten Priorisierung von Performancedimensionen entspricht und den Optimierungsprozess gezielt verbessert.

Vorwort - Acknowledgement

Die vorliegende Dissertation entstand im Rahmen meiner Tätigkeit im Forschungsbereich Energie am OFFIS - Institut für Informatik in Oldenburg. An dieser Stelle möchte ich allen, die mich während meiner Promotion unterstützt und begleitet haben, meinen herzlichen Dank und meine große Wertschätzung aussprechen.

An erster Stelle möchte ich mich bei Frau Prof. Dr.-Ing. Astrid Nieße für die unermüdliche Unterstützung, die wertvollen Anregungen und die inspirierende Betreuung bedanken. Sie war für mich der Leuchtturm, der mich auf Kurs hielt, wenn abseits meines Weges allzu spannende Forschungsfragen lockten. Ihre Expertise und ihre konstruktiven Ratschläge waren entscheidend für den erfolgreichen Abschluss dieser Arbeit.

Des Weiteren möchte ich mich bei Herrn Prof. Dr.-Ing. Sven Tomforde für seine wertvollen und zielführenden Anregungen im Rahmen seiner Tätigkeit als Zweitgutachter bedanken. Insbesondere in der finalen Phase meiner Promotion war er mir eine große Hilfe. Herrn Prof. Dr. Sebastian Lehnhoff danke ich für die Leitung der Prüfungskommission und für die langjährige Unterstützung meines Werdegangs im OFFIS. Herrn Dr. Jörg Bremer danke ich für seine Mitwirkung am Promotionsverfahren als Mitglied der Prüfungskommission.

Mein besonderer Dank gilt Herrn Dr. Martin Tröschel, der mir als Kollege, Freund und Mentor zur Seite stand. Seine Unterstützung hat wesentlich zu meinem persönlichen, wissenschaftlichen und beruflichen Wachstum beigetragen. Mein Dank gilt auch meinen Kolleginnen und Kollegen am OFFIS. Der inspirierende Austausch und die kollegiale Unterstützung waren für mich eine Quelle der Motivation. Besonders hervorheben möchte ich meine Forschungsgruppe *Distributed Artificial Intelligence (DAI)*. Die Zusammenarbeit und Freundschaft mit den Mitgliedern der Gruppe war eine unschätzbare Bereicherung für meine Forschungsarbeit und meine persönliche Entwicklung. Dies gilt insbesondere für die Mitglieder meines Erfolgsteams Frauke Oest, Stephan Balduin, Carsten Krüger und Sanja Stark, die mir geholfen haben, auch mit herausfordernden Situationen umzugehen und beherzt weiterzumachen.

Nicht zuletzt danke ich meiner Familie und meinen Freunden für ihre uneingeschränkte Unterstützung, ihr Verständnis und ihre Geduld in dieser anspruchsvollen Zeit. Ich danke Jürgen und Isabella Schöning, die mir den Rücken freigehalten haben, so dass ich mir nie Sorgen um das Wohlergehen meines Vierbeiners machen musste, auch wenn viel zu wenig Zeit für ihn blieb. Außerdem danke ich Patrick für seine Geduld und sein offenes Ohr, das ich sowohl in besonders schlechten als auch in besonders guten Zeiten ausgiebig in Anspruch genommen habe. Ich möchte mich bei meiner Mutter Hella bedanken, die mir von klein auf gezeigt hat, dass man alles lernen kann und dass es keine Grenzen für das geben sollte, was man sich zutraut.

Contents

I Introduction 1

1 Introduction 3
1.1 Motivation: Distributed power system optimization problems 4
1.2 Context: Self-organized agent-based control systems 7
1.2.1 Agent-based control systems 7
1.2.2 Organic and autonomic computing 9
1.3 Research Goal: Optimized search in distributed search spaces 12
1.3.1 Heuristic optimization 12
1.3.2 Networked heuristics and domain decomposition 14
1.3.3 Hypotheses and research questions 18
1.4 Outline: Development objective and structure of the thesis 22

II Fitness Landscape Analysis 27

2 Global Fitness Landscape Analysis 29
2.1 Fitness landscape features 31
2.1.1 Degree of variable separability 32
2.1.2 Noise 32
2.1.3 Characterization of fitness distribution 33
2.1.4 Modality, basins of attraction and deception 34
2.1.5 Global landscape structure 35
2.1.6 Ruggedness, smoothness and neutrality 36
2.2 Choosing fitness landscape techniques and metrics 37
2.2.1 Variable interaction and sensitivity 38
2.2.2 Fitness and state distribution 41
2.2.3 Dispersion and global landscape structure 42
2.2.4 Walking through unpaved landscapes 43
2.2.5 Summary of applied fitness landscape features 48
2.3 Coupled and dynamic fitness landscapes 48

3 Distributed Fitness Landscape Analysis 53
3.1 Distributed computation of metrics 54
3.2 Composite search spaces 59
3.3 Distributed sensitivity analysis 59

3.4 Distributed computation of fitness distributions 64
3.5 Distributed computation of structure exploring metrics 68
3.6 Summary of distributed computation of feature groups 70
3.7 Correlation analysis of metrics 72

III Communication Topologies for distributed Optimization Algorithms 77

4 Distributed Optimization Algorithms 79
4.1 Parallel cooperative metaheuristics 80
  4.1.1 Levels of parallelism 81
  4.1.2 Information-sharing cooperation mechanism 83
4.2 Spatially distributed optimization of decomposed domains 86
  4.2.1 Combinatorial optimization heuristic for distributed agents (COHDA) 86
  4.2.2 Island model differential coevolution (IDICE) 91
4.3 Impact of communication topologies 97

5 Dynamic Communication Topology Adaptation 101
5.1 Dimensions of algorithm performance 102
5.2 Topology variants 105
  5.2.1 Starting in a small world 105
  5.2.2 Adaptation strategies 108
5.3 Observed effects of topology variants 112

IV Learning Optimal Topology Variants 117

6 Topology variant optimization (ToVarO) 119
6.1 Experimental setup 122
  6.1.1 Problem instances, FLA metrics, and optimization parameters 122
  6.1.2 Measures for performance dimensions 125
  6.1.3 The labeling problem 129
6.2 Evaluation of feature importance 132
  6.2.1 Coefficients of determination 133
  6.2.2 Correlation Analysis 135
  6.2.3 Permutation importance 144
6.3 Evaluation of predictive capability of ML models 153
  6.3.1 Choosing models for ToVarO 153
  6.3.2 Analysis of predictions 155
  6.3.3 Prediction based topology selection 160
  6.3.4 Sensitivity to noise 165

V Conclusion **173**

7 Conclusion **175**
7.1 Results and research contributions 175
7.1.1 Part II: Distributed Fitness Landscape Analysis 176
7.1.2 Part III: Communication Topologies for distributed Optimization Algorithms 177
7.1.3 Part IV: Learning Optimal Topology Variants 178
7.2 Future work 179
7.2.1 Extension of distributed FLA 179
7.2.2 Refining communication topology adaptation strategies 180
7.2.3 Advancement of algorithmic aspects 181
7.2.4 Transfer to real-world energy system optimization problems 183

Acronyms **187**

List of Symbols **189**

Bibliography **191**

A Appendix A **209**
A.1 Benchmark functions 209
A.1.1 Function set 1 212
A.1.2 Function set 2 217
A.2 Algorithm Setup 222
A.2.1 COHDA 222
A.2.2 IDICE 223

B Appendix B **227**
B.1 Parameter Tuning for distributed FLA 227
B.1.1 Distributed sensitivity analysis 227
B.1.2 Distributed computation of fitness distributions 233
B.1.3 Distributed computation of structure exploring metrics 236
B.1.4 Final parameter setup 243
B.2 Correlation Analysis of distributed FLA metrics 244

C Appendix C **249**
C.1 Correlation Analysis 249
C.1.1 Interpretation of correlation coefficients 249
C.1.2 P-Values 250
C.2 Permutation Importance 250
C.3 ML Model Parameters 250

Part I

Introduction

This introduction presents the motivational context of optimization problems in power systems and highlights their particularities and challenges (section 1.1). These motivate the use of agent-based and self-organized control systems (section 1.2), as well as the use of truly distributed network-based optimization heuristics. In section 1.3, the research hypotheses and research questions of this thesis are derived, which deal with the problem-specific optimization of an important hyperparameter of these optimization heuristics: the communication topology. Finally, section 1.4 outlines the development objective and the subsequent organization of this thesis.

1

Introduction

The electrical energy system is currently in a process of profound change as a large proportion of conventional power plants is replaced by renewable energy sources. The energy transition is a complex and multi-faceted process that involves changes at many levels, from the way energy is produced and consumed to the policies and regulations that govern it [Gie+19]. This demands new approaches, such as decentralized control at device level, distributed coordination of energy sources, or real-time optimization at system level [Dör+19].

Power system operation involves many optimization problems that aim to balance generation and consumption. Other factors, such as the operating limits of power plants and grid resources, or economic aspects, often need to be considered as well. With the increase in generators and controllable loads at the distribution grid level, such optimization problems become more complex and are further compounded by other challenges such as increased uncertainty due to volatile weather-dependent feed-in, the distributed nature of the required data, and real-time requirements. These types of problems serve as motivating use cases for this thesis. Therefore, they are presented in more detail below (section 1.1), along with an example to illustrate the challenges. Self-organized agent-based control systems are a suitable and practical approach to meet these challenges [Dör+19; Kan+15; TAA20; Mol+17]. By implementing so-called self-x capabilities, such systems are able to adapt dynamically to changing requirements and conditions. They provide the systems engineering context for this thesis. As such, they are introduced in more detail in section 1.2. Although agent-based control systems provide the systems engineering paradigm, implementing the optimization algorithm or choosing an appropriate optimization method is a separate design decision. Due to problem characteristics such as nonlinearity, high dimensionality, and multimodality, the use of optimization heuristics instead of exact methods is often an adequate choice [Tal09]. However, they pose their own unique challenges, especially when implemented in a distributed or networked control system. One of them is the selection of a suitable Communication Topology (CT) that determines the information exchange between the distributed solvers and thus the information dissemination during the optimization process [Cra19; CT07]. The modeling of (dynamic) communication topologies and the impact of different topology variants on algorithm performance is a major focus of this thesis. Since the impact also depends on the problem characteristics [HN21b; HN21a], the second main focus is on the systematic analysis of distributed optimization problems via Fitness Landscape Analysis (FLA). Section 1.3 gives a brief introduction to the fundamental concepts, highlights the underlying relationships, and presents the derived research hypotheses and research questions. Finally, section 1.4 presents the development objective and the further outline of the thesis.

1.1 Motivation: Distributed power system optimization problems

Power system optimization problems are manifold. They range from planning problems for future power system expansion to everything needed for day-to-day operations. Safe and reliable operation entails a variety of optimization problems, such as classical unit commitment, economic dispatch, optimal power flow, distribution system reconfiguration, and maintenance scheduling [CM20]. Furthermore, many ancillary services like voltage and frequency control, reactive power supply, and congestion management involve optimization. The focus of this work is on optimization problems that require the coordination of generation and consumption in distribution grids, such as the aforementioned ancillary services or economic dispatch. The large-scale expansion of distributed energy resources (DERs), among them PV systems, controllable loads, or battery storages, also in the form of e-mobility, raises the complexity of coordination tasks in distribution grids, but also offers opportunities [Lop+20]. Intelligent control strategies can harness the flexibility of these multifaceted resources. For example, marketing flexibility in (future) ancillary services markets can provide additional benefits to unit operators and ensure the safe and reliable operation of the power system in an efficient manner [Hol+20]. Virtual Power Plants (VPPs) are already in operation today and are a concept for pooling small-scale flexibility. A VPP is an aggregation of distributed units in the power grid, usually coordinated by a central control system. The purpose of the VPP is the joint marketing of electricity and the flexibility of the combined pool of units[1]. This type of aggregation also requires coordination of the units' power supply and consumption. To illustrate the characteristics of this type of problem, a simple formalization for such scheduling in virtual power plants is described in more detail below.

The goal of operational planning in a VPP is to jointly achieve a target schedule involving all units in the pool. This target schedule is a time series, typically at 15-minute resolution, that contains power value targets. To formalize this, the flexibility of the units must first be modeled. A simple form of flexibility modeling is

---

[1] https://www.next-kraftwerke.com/knowledge/what-is-a-virtual-power-plant

to determine a set of feasible schedules for each unit. These schedules are again time series with the same resolution as the target schedule, containing feasible values for power generation or consumption. This results in a multiple-choice combinatorial optimization problem in which one of these schedules must be selected for each unit, and the combination of all chosen schedules should be as close as possible to the target schedule [HLS13a]. The following formalization is based on the work of Hinrichs et al. in [HLS13a; Hin14]. Table 1.1 specifies the definitions required for the subsequent objective function and constraints.

u	identifies a single unit of the VPP, with $i \in \{1, 2, \ldots, k\}$ and k being the number of units
j	index of a single schedule of a unit, with $j \in \{1, 2, \ldots, m\}$
i	index of a single interval of a schedule, with $i \in \{1, 2, \ldots n\}$
p_{uj_i}	power of schedule j of unit u at time interval i
x_{uj}	boolean that indicates if schedule j is selected for unit u, with $x_{uj} \in \{0, 1\}$
t_i	power of the target schedule at time interval i

Tab. 1.1.: Definitions of terms used in eq. (1.1) and eq. (1.2)

Equation (1.1) is the objective function that aims to minimize the deviation from the target schedule for each time interval.

$$\min \left(\sum_{i=1}^{n} \left(|t_i - \sum_{u=1}^{k} \sum_{j=1}^{m} p_{uj_i} \cdot x_{aj}| \right) \right) \tag{1.1}$$

Equation (1.2) is the constraint that ensures that for each unit exactly one schedule is chosen.

$$s.t. \sum_{j=1}^{m} x_{aj} = 1 \tag{1.2}$$

The presented problem is similar to the well-known multiple-choice knapsack problem that can be solved in polynomial time using dynamic programming [HLS13a]. However, it serves as a simple entry point to demonstrate the basic properties of such optimization problems involving the coordination of power generation and demand in any form. Simply modeling flexibility differently can significantly change the optimization problem. For example, the set of schedules can be replaced by a flexibility model that reflects the technical degrees of freedom and constraints of the units in detail and unfolds the possibly continuous space of feasible schedules, e.g., [Tie+22], [BRS11]. The choice of each unit's schedule still represents a multidimensional decision variable (number of time intervals) for the global problem. However, depending on the constraints of the unit and the modeling of flexibility, the search space for this unit may be discrete, continuous, or a combination of these.

Coordinating Distributed Energy Resources (DERs) and controllable loads often leads to optimization problems in which each unit must contribute to a global objective, as illustrated in the motivational example. Thus, how well the contribution of one unit serves the overall goal depends on the contributions of other units. Consequently, there are dependencies between the decision variables. However, each unit has its own constraints, and the operating options form a separate subsearch space for the global optimization problem. This corresponds to an inherent decision set decomposition, i.e., a partition of the search space into several subsearch spaces. Search space decomposition is usually applied to very complex and large-scale problems in order to make them solvable. Due to the number of units and the length of the planning horizon, energy optimization problems are usually high-dimensional. Thus, the existing decomposition of the search space can be used to achieve scalability in problem solving. As explained above, the subsearch spaces in the energy system are also interdependent, which means that the subproblems cannot be solved independently. These subsearch spaces, i.e. the plant flexibilities, can be very heterogeneous, and the information needed to compute them is usually distributed. This information consists of a static component, the plant master data, and dynamic components, such as the current state of the plant or other local conditions or requirements. At least the second part of this information is originally only available at the plant site and can change constantly.

In summary, energy optimization problems that aim at coordinating the power generation and consumption of a large number of decentralized units are typically *high-dimensional, information is distributed, and the entire search space is inherently distributed over a large number of heterogeneous and interdependent subsearch spaces*. Such natively distributed optimization problems can be solved using distributed optimization techniques, which can handle many of the potentially problematic problem characteristics. In such a distributed optimization system each unit can be represented by an agent. This agent knows the local measures, cost functions, and constraints of its unit. The agents only share information that is actually needed to find a common solution. This solves the problem of distributed information, increases the privacy of data, and reduces the required communication effort, since not all information has to be sent to a central system. Furthermore, an agent-based system can exploit the existing decomposition of the search space. Interdependencies can be taken into account by exchanging information between agents. In addition, such decentralized systems increase robustness since there is no single point of failure, and the scalability of a distributed system is one of the biggest advantages [Mol+17]. Distributed control systems and the implementation of distributed optimization are a central systems engineering context of this thesis. Therefore, the basic concept will be explained in the following sections.

1.2 Context: Self-organized agent-based control systems

The use of self-organized agent-based control systems has become increasingly popular in recent years due to their ability to provide efficient and robust control in complex and dynamic environments. In the following, the basics of software agents, Multi-Agent Aystems (MAS), and their use as control systems are briefly described. This is followed by an introduction to self-organizing systems and software engineering paradigms that allow MAS to be designed with the necessary flexibility and reliability through the implementation of self-management capabilities in combination with control mechanisms.

1.2.1 Agent-based control systems

The agent paradigm is a software engineering paradigm that enables the modeling of software architectures that contain numerous dynamically interacting components, with each component having its own thread of control and employing complex coordination protocols [WC00]. An agent is a software program that is able to perceive its environment via sensors, model the acquired knowledge internally if necessary, and act autonomously in this environment to achieve its design objectives [Rus10; WJ95]. According to Wooldridge [Woo01], autonomous action in this context means that no intervention by humans or other systems is necessary, i.e. the agent has control over its internal state and behavior. Furthermore, *intelligent* agents are capable of *flexible* autonomous actions and thus *react timely* to changes in their environment or even take *pro-active actions* to meet their design objectives. In addition, intelligent agents possess social abilities and can interact with other agents (or even humans). These social capabilities enable agents to negotiate and cooperate with others. The resulting Multi-Agent System (MAS) allows agents to collectively achieve goals that would be unattainable for a single agent.

In [JB03], Jennings and Bussmann argue that agent-oriented approaches are well suited for engineering complex control systems. The agent paradigm provides the fundamental means to manage complexity, i.e., decomposition, abstraction, and organization. Agent-oriented decomposition is an effective method to partition the problem space of a complex system as it modularizes the components with respect to their intended objectives. In addition, agent-oriented approaches model the interaction between subsystems and among their components in terms of high-level social interactions providing an intuitive abstraction. Modeling these social interactions also allows for intuitive and flexible modeling and management of organizational relationships, enabling different levels of aggregation and straightforward extensibility. Application areas of agent-based control systems include many distributed optimization problems such as sensor networks, building automation, smart manufacturing, and especially energy systems [Yan+19]. Spatially distributed systems, such as power systems, pose a particularly difficult control challenge as they often are nonlinear and have distributed information and control structures that include sensors, controllers, and actuators [TÇT07].

In addition to increasing complexity, increasing volatility poses challenges for control systems. These systems must operate in dynamic and heterogeneous environments. Moreover, requirements can change frequently. Therefore, the control systems must be robust and able to flexibly adapt to evolving environments [SGK06]. The concept of *self-organization* allows the design of agent-based control systems that are capable of dynamically adapting to changing requirements and conditions. In [SGK05], Serugendo et al. define *self-organization* as "as the mechanism or the process enabling a system to change its organization without explicit external command during its execution time". The key properties of self-organizing systems are thus autonomy, i.e., no explicit external control, and dynamic operation, i.e., the ability of the system to evolve over time. Serugendo et al. further distinguish between weak and strong self-organization, depending on the degree of centralization or decentralization of the system's internal control mechanisms. The concept of *self-organization* is also often linked to the phenomenon of *emergence*. According to Serugendo et al. "emergence is the fact that a structure, not explicitly represented at a lower level, appears at a higher level"[SGK06]. In the case of a MAS, this means that the global behavior of the system often emerges from the local interactions of individual agents. Self-organization and emergent behavior often coincide in systems characterized by decentralised control and local interactions. However, both concepts can occur independently [SGK06].

The ability of control systems to flexibly adapt to external and internal changes is crucial for use in safety-critical systems such as power systems. However, the self-organizing behavior of control systems also involves risks such as negative emergence, i.e., the interaction of autonomous systems or agents leading to undesired behavior. Therefore, several concepts have been developed to monitor such dynamic systems and to intervene in a corrective way if necessary, while at the same time enhancing the systems' capabilities for self-adaptation and self-improvement. In the following, such concepts from the fields of organic computing and autonomic computing are presented.

1.2.2 Organic and autonomic computing

Organic Computing (OC) is a systems engineering paradigm that seeks to address increasing complexity by employing concepts similar to those found in nature, i.e., "found in living things". The *organic* behavior of systems aims to achieve certain capabilities, including robustness, continuous optimization, adaptivity, flexibility, and efficiency, even when subject to internal or external disturbances [MT17]. These capabilities are to be achieved by equipping systems with a set of so-called self-x capabilities. Such properties increase the autonomy of the system and shift decisions from design-time to run-time and from the system engineer to the systems themselves [MT17]. To ensure the necessary trustworthiness of the system, it should still be possible to monitor and explicitly interfere with these self-x processes in order to prevent undesired behavior. This leads to the notion of "controlled self-organization", which combines the organic capability of the system with a control mechanism. This control mechanism is usually implemented by the Observer/Controller architecture [Tom+11]. The generic Observer/Controller design pattern is a regulatory feedback mechanism that continuously monitors the behavior and environmental conditions of the productive system (responsible for the system's basic purpose). The behavior of the productive system can be influenced, if necessary, by the adjustment of parameters that have an impact on the performance of the system with respect to the given system goals. Figure 1.1 illustrates this structure following [Tom+11]. The productive system is referred to as the *System under Observation and Control* (SuOC). Instead of being centralized as shown in the figure, the Observer/Controller structure itself may be implemented in different distribution variants.

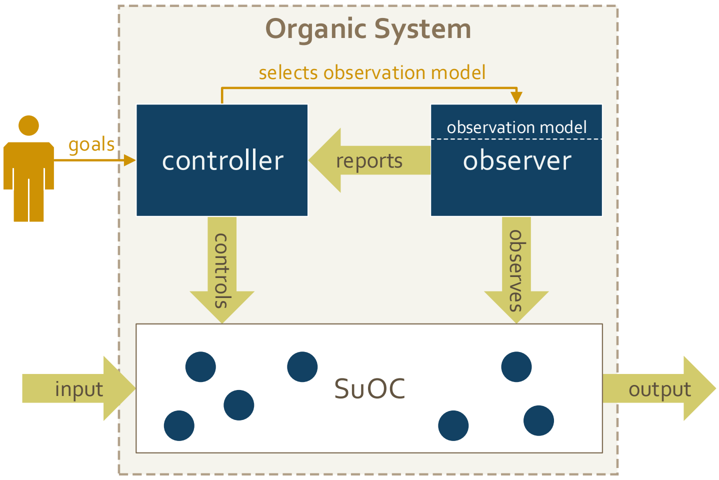


Fig. 1.1.: Observer/Controller architecture that forms an organic system with a *System Under Observation and Control* (SuOC); Figure adapted from [Tom+11]

Autonomic Computing (AC) [KC03] is another computing paradigm that aims to address the increasing complexity of ICT systems by introducing self-management capabilities. Inspiration again comes from biological principles, as reflected in the naming after the autonomic nervous system. The primary concern is to reduce the required level of human intervention in the installation, configuration, optimization and maintenance of systems. For this purpose, the productive parts of the system (*Managed Resources*) are equipped with a control mechanism (*Autonomic Manager*) that takes over the management of its resource and also adjusts the interactions with other resources. AC focuses more on automating management and control functions to ensure reliable and efficient operation, while OC focuses more on the emergence of behavior from component interactions, allowing the system to evolve and adapt over time. However, both approaches are based on the self-management of systems consisting of a set of autonomous and interacting components combined with control mechanisms that can be centralized or decentralized to varying degrees.

There is no generally accepted definition of the self-x properties that constitute the self-managing behavior of systems. Below are some of the most prominent self-x properties relevant to this thesis, as defined by Loeser et al. in [Loe+22]. They are based on the terminology used in OC [MT17] and AC [KC03].

Self-Configuration

Loeser et al. [Loe+22] define "self-configuration as the ability of a control mechanism to change the parameterization of components and systems following high-level policies" according to [KC03]. This can be performed in a static or an adaptive manner. Static approaches require fully predictable environments, while adaptive self-configuration mechanisms must ensure reliable and accurate operation under uncertainty. The aspect of decision techniques in the context of self-configuration is further emphasized by Loeser et al. Decision-making techniques for self-configuration include predefined decisions, decision-making driven by machine learning (ML), and constraint programming. Predefined decisions imply a fixed behavior for a given situation determined at design time. ML approaches either integrate the self-configuration decision into a learning algorithm, or use learning techniques to obtain predefined decision options. Constraint programming employs solvers to search for an optimal solution. Regarding dynamic control problems, Loeser et al. attribute the most potential to ML-based methods.

Self-Organization

Self-organization is defined by Loeser et al. based on [MT17] as follows: "Self-organization is the ability of distributed controllers to modify the overall system's structure (i.e., relations between components and the corresponding interaction schemes) depending on current conditions and based on the particular system goal". This is consistent with the definition of Serugendo et al. [SGK05] referenced in section 1.2.1. The organizational aspects can be grouped into the main categories of task or resource allocation, relation adaptation, organisational design, and collective decision making. Self-organising task allocation deals with how individual agents or subsystems are assigned specific tasks or resources. Existing work often addresses this aspect in the context of wireless sensor networks [LLA04] and multi-robot systems [Liu+07], using decentralized algorithms to assign specific tasks to individual sensor nodes or robots. Relation adaptation is concerned with how autonomous subsystems modify their relationships with other subsystems. Organizational design focuses on the interaction and relationship of the agents or actors in the system, which is often implemented through the assignment of roles. Collective decision making is concerned with how members of a group reach consensus or make decisions in a decentralized manner. Consensus typically reflects a compromise among the various members and also involves the challenge of balancing the conflicting goals of efficiency, accuracy, fairness, and robustness in dynamic and uncertain environments.

Self-Optimization

According to Loeser et al. self-optimization typically deals "with the improvement of control parameters of the productive system or the management of the overall system structure based on a given set of goals". The goal is to achieve better system performance or efficiency without the need for external intervention or human input. The two main approaches to self-optimizing systems are ML-based and optimization-based techniques. Optimization techniques are often used to generate new system configurations or adaptation plans, including techniques from probabilistic, combinatorial, evolutionary, stochastic, mathematical, and metaheuristic optimization. Self-optimization can also be achieved through autonomous learning, with reinforcement learning being the most prominent variant.

In the motivational context of power system optimization problems, the primary task for an organic control system is to optimize one or more aspects of the power system. Of course, such optimization processes can also be optimized by adjusting relevant parameters of the organic control system depending on the current conditions. This corresponds to a self-optimization of the optimization process and is one of the main aspects of this thesis. In the following section, basic principles of optimization in distributed systems will be presented. Subsequently, the type of optimization algorithms and the parameters to be optimized are discussed. Furthermore, challenges and prerequisites are outlined, which eventually leads to the hypotheses and research questions addressed in this thesis.

1.3 Research Goal: Optimized search in distributed search spaces

Distributed optimization is often an important functionality in distributed control systems. Due to the complexity of the optimization problems, heuristic methods are often preferred over exact methods. Distributed optimization also requires multiple distributed solvers that interact with each other over networks. In the following, the basics of optimization problems in general and of heuristic optimization methods in particular are briefly presented. Afterwards, the special aspects of distributed execution of heuristics are discussed with emphasis on the the so-called Communication Topology (CT). In addition, the concept of characterizing optimization problems through Fitness Landscape Analysis (FLA) is introduced. The last subsection puts the presented challenges back into the motivational context of distributed energy problems and self-organized control systems. This leads to the hypotheses and research questions underlying this thesis.

1.3.1 Heuristic optimization

In classical optimization models, the goal is to find the best solution to a problem from the set of all feasible solutions. The optimization problem can thus be defined by the couple (S, f), where S is the set of feasible solutions, also called search space, and $f : S \rightarrow \mathbb{R}$ is the objective function, that aims to maximize or minimize the cost, utility, or fitness function [Tal09]. Optimization models can be classified according to different aspects. First, it must be considered whether an optimization is constrained or unconstrained, i.e., whether the problem contains constraints that must be satisfied in addition to optimizing the objective function. The constraints limit the space of feasible solutions. Second, the type of decision variables varies. They can be continuous, discrete (integer), or even mixed. Finally, the type of function of the objective function and the constraints is of importance. A basic distinction in this case is the one between linear and nonlinear functions. For linear optimization models, i.e., models with linear objective functions and linear constraints [Tal09], there are efficient exact algorithms, such as the simplex-type method [Dan51] or interior-point methods [Kar84]. *Nonlinear programming models* (NLP) have a nonlinear objective function and/or nonlinear constraints. They are much harder to solve than linear models, although linearization techniques can be used and problems of moderate size can be solved by some efficient exact algorithms. Especially when other factors are added, such as high dimensionality, multimodality, epistasis (interaction of parameters) and non-differentiability, this type of optimization problem quickly becomes very difficult or impossible to solve with exact methods [Tal09]. However, many optimization problems in the energy system possess exactly such properties, as presented in section 1.1.

Since the solution with exact methods would either take a very long time or is not possible at all, optimization heuristics are often used for this type of problem models. A heuristic is an optimization algorithm that does not aim for an exact solution, but instead approximates the solution. It often leads to *good* solutions in a reasonable amount of time, even for large problem instances. However, there is no guarantee that globally optimal solutions will be found, nor that minimum solution quality or run time constraints apply. Optimization heuristics can be developed specifically for a problem or problem instance. *Metaheuristics*, on the other hand, are general-purpose algorithms. They are often inspired by natural phenomena, such as the behavior of ant colonies, the cooling of metal, or the process of evolution. These methods can be considered as high-level general procedures that serve as guiding strategies for the development of problem-specific algorithms [Tal09]. Most metaheuristics are iterative algorithms. They start with a complete solution (or a set of solutions), i.e. with a value for each decision variable, which is transformed in each iteration by applying operators.

The metaheuristic's general procedure, including transformation operators, specifies how the search space S is sampled. Heuristic search involves two opposing aspects: *exploration* of the search space (diversification) and *exploitation* of the best solutions found (intensification) [Tal09]. A search algorithm must achieve a good balance between exploration and exploitation to be successful [ČLM13]. That such a balance is necessary can be easily realized if we consider the two extreme cases: A search algorithm that maximizes exploration, i.e., aims to visit completely new regions of a search space at each iteration, resembles a random search. However, if a search algorithm heavily exploits the regions of a search space within the neighborhood of previously visited points, it comes closer to a local search, e.g., a hill-climbing algorithm. The balance between exploration and exploitation determines the way in which the search space is covered, i.e., which solutions can be found at all, and thus has a significant impact on the solution quality and the time and computational effort required for optimization. This balance is usually achieved by setting appropriate control parameters (*hyperparameters*) for metaheuristics. However, even for the same metaheuristic, the effect of the hyperparameter settings is problem dependent and requires different choices for different problems. In [ČLM13], Črepinšek et al. provide a survey about exploration and exploitation in Evolutionary Algorithms (EAs). They give several examples of how different aspects in EAs, such as selection, mutation operators, population size, and representation, can influence the degree of exploration and exploitation. In general, two different approaches to controlling parameters can be distinguished [EHM99]:

- **parameter tuning**: selection of the hyperparameter set in advance (offline), e.g. by following general guidelines or employing design of experiments techniques [RK10; Kra08].
- **parameter control**: hyperparameters are adjusted during runtime (online); this can be done in different ways [EHM99]:
  - *deterministic*: the value of a parameter is modified by a deterministic policy, e.g. after a predefined number of iterations, without regard to the current state of the search.
  - *adaptive*: based on the current progress of the optimization process, the direction and/or magnitude of the change in the hyperparameter is determined.
  - *self-adaptive*: hyperparameters can be encoded in the representation of the solution, e.g. in the chromosomes for EAs, and are subject to metaheuristic operators such as mutation and recombination. The idea is that better values of these encoded parameters will lead to better individuals. These individuals will have a higher probability to survive and thus propagate these better parameter values.

For many optimization problems, it has been shown that it is advantageous to start with exploration and move to exploitation later in the search. This can be achieved by deterministic parameter control. However, certain problem characteristics, such as multimodal or dynamic environments, may favor (self-)adaptive approaches. Furthermore, there are many optimization problems that do not require runtime parameter control to be solved efficiently [ČLM13].

1.3.2 Networked heuristics and domain decomposition

As outlined above, the power system poses a number of distributed optimization problems for which a distributed control system is expected to be advantageous for various reasons. Optimization heuristics used for this purpose must meet the special requirements of distributed search spaces, given the flexibility of various DERs that are intended to collectively solve a higher-level problem. This is a use case for which parallel metaheuristics based on communication topologies are well suited. In such a heuristic, several solvers run in parallel. They exchange information via a communication or exchange topology. This is usually modeled by a graph, where the nodes correspond to the solvers and an edge indicates direct information exchange between the solvers located at the respective nodes. This Communication Topology (CT) is an important hyperparameter for this type of heuristics. In [KHE14], Karafotias et al. cite several examples of work examining the effects of communication topologies on parallel EAs. This includes the work of Arnaldo et al. [Arn+13], which supports the fact that different problem characteristics require different communication topologies. Another work cited by Karafotias is that of Cantú-Paz [Can99], in which he presented a theoretical analysis of various hyperparameters that determine the information exchange in parallel EAs, such as the CT, and their interdependencies. One of Cantú-Paz's main findings was that the degree of connectivity of the topology strongly influences the quality of the solution.

The CT determines the information flow in the system. Low connectivity leads to little exchange between solvers and thus to diversification (exploration) of the search. High connectivity leads to strong information diffusion and thus exploitative search behavior. Figure 1.2 illustrates this relationship. This effect is not exclusive to parallel search heuristics. In [LF07], Lazer and Friedman examined how the structure of communication networks among actors might influence systemic performance, especially collective problem-solving, with respect to social and organizational networks. Their results show a similar effect of strong and weak interconnection of topologies. Moreover, they highlight the trade-off between these two designs. Both contribute to system performance, either by maintaining the necessary diversity or by rapidly propagating powerful strategies.

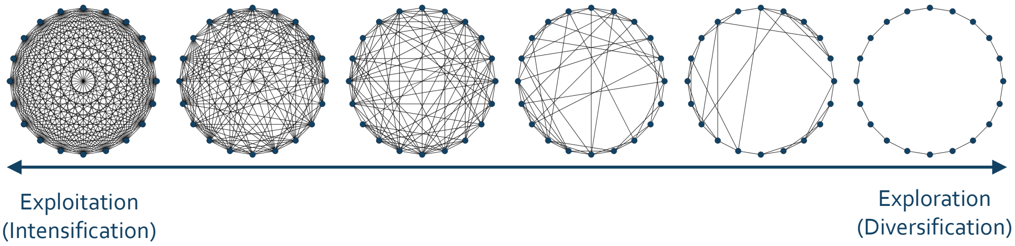


Fig. 1.2.: Effect of CT connectivity on parallel metaheuristics: A high degree of connectivity leads to a high rate of information exchange and thus to more intensive search in certain regions of the search space, while a low degree of connectivity leads to a higher diversification of information and thus to exploratory search behavior.

However, existing work on communication topologies rarely considers problems that are truly spatially distributed and whose search space S is composed of multiple distributed subsearch spaces $\mathbb{S} = \{S_1, S_2, \dots, S_n\}$. But this is exactly the case we encounter in distributed optimization problems in the energy sector. These optimization problems often involve coordinating the energy supply and demand of a large number of DERs and controllable loads, as discussed in section 1.1. Figure 1.3 illustrates this with the motivational example from section 1.1. An overall operation schedule for a grid area, is composed of the individual schedules of all involved plants. The goal in the example is to reach a target schedule for the grid area. Thus, the flexibility of each of these plants is a part of the solution. In the chosen example, the flexibility that each agent can contribute to solving the optimization problem is represented by a set of feasible schedules for its asset portfolio[2]. The choice of a particular schedule by an agent thus represents a decision variable for the global optimization problem. Therefore, the flexibility of each plant may be considered as a separate dimension of the overall search space $\mathbb{S}$. When this global optimization problem is solved in a distributed way, e.g. by an agent-based control system (see section 1.2), each agent controls only its own plant or portfolio of plants. Thus, it can choose a schedule for its plant(s) and must cooperate with the other agents to achieve the overall target schedule. This is a *decision set decomposition* from the perspective of the global optimization problem. This means that each solver

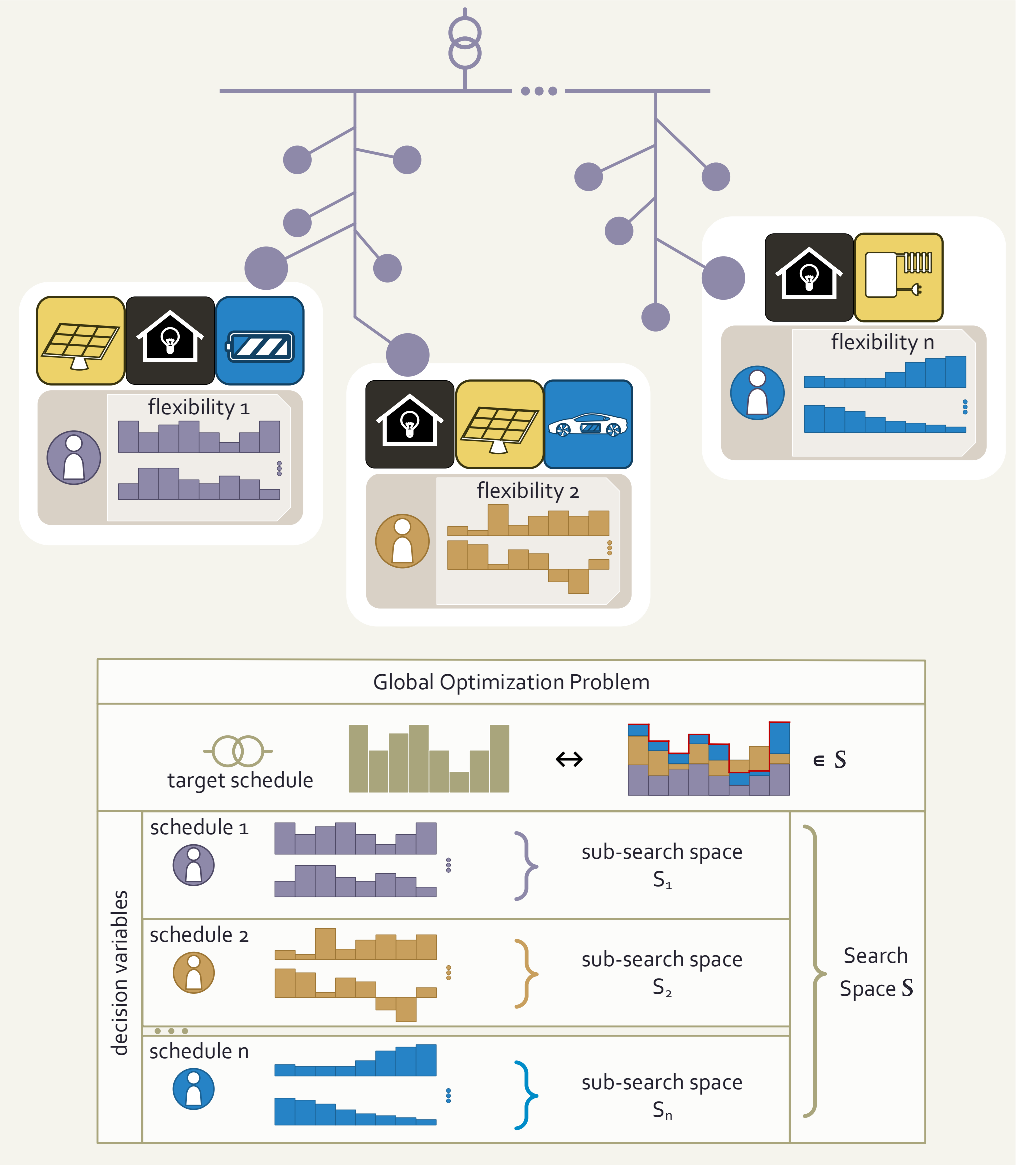


Fig. 1.3.: Illustrating example of the inherent decomposition of the search space $\mathbb{S}$ in energy system optimization problems

---

[2] Note that flexibility can be represented in several ways. For the sake of clarity, and following the example in section 1.1, a set of feasible schedules has been chosen for the illustration.

or agent has exclusive control over a set of decision variables. Without decision set decomposition, the CT influences how information is exchanged between self-contained solvers to improve the overall search process. However, each solver is capable of finding a complete solution on its own. With decision set decomposition, each solver can only solve a part of the problem and is thus highly dependent on information about the partial solutions of the other solvers. Moreover, unless the decomposed decision variables are separable, which is not the case for the motivating problems from the energy domain, one agent's choice affects how another agent must assess its own set of choices. For example, when an agent chooses a schedule for its plant, this changes which possible schedules from another plant best complement it and thus come closest to the overall target schedule. As a result, the impact of the CT is strongly pronounced.

Previous work on designing optimized communication topologies in an algorithmic context mainly addresses parallel metaheuristics without decision set decomposition (see section 4.3 for more references). In addition, there is little discussion of adjusting the CT at runtime as part of parameter control. There is clear evidence that the CT has a strong impact on the degree of exploration and exploitation. Therefore, finding a way to induce a beneficial shift between explorative and exploitative search behavior by adjusting the connectivity of the topology at runtime is an essential part of this thesis. However, which topology, or which topology adaptation at runtime, yields the best results is likely to depend heavily on the properties of the specific optimization problem.

In order to determine an appropriate topology or topology adaptation for a given problem in advance, the relationship between problem characteristics and the effect of different topology variants must be investigated. For this purpose, the characteristics of the optimization problems have to be captured and quantified in order to make them comparable. So-called Fitness Landscape Analysis (FLA) can be applied for exactly this purpose. FLA is the study of the relationships between the possible solutions in the search space and the optimization objective function (or fitness function). It involves a variety of methods, most of which first draw samples from the search space and then compute metrics based on the position of those samples in the search space and the associated values of the objective function. These metrics allow to characterize the problem with respect to different properties, such as modality or the degree of interdependence of decision variables. FLA is often used to estimate a particular optimization problem's hardness for a particular type of algorithm, and therefore which algorithms are best suited to solve the problem [Mal21]. It can also be used as a basis for parameter tuning or parameter control.

The special case of distributed search spaces with dependencies between decision variables corresponds to the concept of so-called coupled fitness landscapes in FLA.

Each subsearch space in $\mathbb{S}$ is considered a distinct fitness landscape. The values of the objective function for the solutions in a subsearch space change depending on the values assumed for the decision variables from other subsearch spaces. This is also the case for the example given in fig. 1.3. When an agent changes its schedule choice, this changes the way other agents' schedules complement that schedule to achieve the target schedule. FLA for coupled fitness landscapes is mostly studied in the context of coevolutionary algorithms (see section 2.3 for more details). The methods commonly used to compute fitness landscape metrics are not suitable for computation in spatially distributed systems, since they require consolidated knowledge about all subsearch spaces in a central location. Distributed computation of such metrics poses its own challenges, such as the tradeoff between metric accuracy and computational and communication overhead. However, the characteristics of the problem are an essential input for a problem-specific design of the CT or an appropriate parameterization of the topology adaptation. Therefore, the development of a distributed computation method for an appropriate set of fitness landscape metrics is another important part of this thesis. The following section summarizes the main points outlined above and highlights the resulting hypotheses and research questions.

1.3.3 Hypotheses and research questions

Previously, real-world optimization problems were presented as a motivating context and the concepts of self-organizing agent-based control systems were introduced as a suitable solution framework. Furthermore, the challenges at the algorithmic level and the special role of communication topologies for distributed optimization heuristics were highlighted.

The story in a nutshell is as follows:
Power system operation involves a wide range of complex optimization problems, requiring the coordination of power generation and consumption across a large number of distributed units. For various reasons, such as scalability, robustness or data privacy, a decentralized system is advantageous for controlling these processes. Self-organized agent-based control systems are a promising approach, since they can flexibly adapt to changing environments through special capabilities such as self-configuration or self-optimization. The distributed optimization problems are usually very complex, i.e., nonlinear, high-dimensional, and with a high degree of interdependencies among the decision variables. Therefore, optimization heuristics are a practical solution method. In turn, for use in a distributed control system, the optimization heuristic must also operate in a distributed manner.

The Communication Topology (CT), which determines the direct communication links between agents, is an important hyperparameter for such parallel and distributed optimization heuristics. It has a strong impact on the dissemination of information in the system and thus influences the degree of exploration and exploitation of the distributed search space. The balance between these two properties significantly affects the performance of optimization heuristics in terms of solution quality and computational effort. For spatially distributed systems that rely heavily on network communication, this effect is even more pronounced. Moreover, the motivating optimization problems from the energy domain exhibit an inherent decision set decomposition. In other words, each agent controls only a subset of the decision variables, and thus solving the global problem and making choices for all decision variables requires cooperation among agents.

This makes the choice of CTs an important design aspect for such distributed optimization heuristics. In addition, the most beneficial degree of exploration and exploitation changes over the course of the optimization process. Adjusting hyperparameters at runtime in the context of parameter control is a common method for improving algorithmic performance. Therefore, adjusting the CT at runtime is a promising strategy for improving the performance of distributed optimization heuristics to meet the self-optimizing aspirations of organic computing.

This leads to the first hypothesis:

Hypothesis 1
An adaptation of the communication topology at runtime of a distributed optimization heuristic solving an optimization problem with inherent decision set decomposition affects the search behavior of the algorithm and can improve its performance in terms of solution quality, computational effort and communication overhead.

The first research question follows from this hypothesis:

Research Question 1

---

*How can dynamic **communication topology variants** composed of an **initial topology** and a **topology adaptation strategy** be modeled, and how do these variants affect the various performance dimensions of distributed heuristics?*

In addition, it is very likely that there is no such thing as a "one size fits all" approach. The effects of a Communication Topology Variant (CTV) depend on the optimization heuristic as well as on the properties of the problem at hand. In order to predetermine a good CTV for a given problem, first a relationship between the problem properties and the effects of different CTVs must be established. As pointed out, fitness landscape analysis (FLA) can be applied to determine the characteristics of optimization problems. However, there are few approaches to perform FLA for search spaces with inherent decision set decomposition, and none at all when FLA must be performed in a distributed manner. The presumption is that this is feasible, with suitable sampling methods. However, it is also likely that there is a compromise between computational cost and the expressive power of the features. This leads to the second hypothesis:

Hypothesis 2
Distributed fitness landscape analysis for search spaces with decision-set decomposition can determine meaningful features that can serve as the basis for parameter tuning and control of distributed optimization heuristics.

This hypothesis, in turn, leads directly to the second research question:

Research Question 2

How can ***fitness landscape analysis*** *be performed in a* ***distributed manner*** *while creating a reasonable tradeoff between computational and communication effort and the expressiveness of the computed features and which metrics need to be included into a set to provide a reasonable characterization of distributed optimization problems?*

Addressing research question 2 leads to the development of a methodology for the distributed computation of fitness landscape metrics and a selection of a set of metrics that provide a comprehensive characterization of optimization problems with inherent decision set decomposition. The final goal is to combine this with the modeling of CTVs to enable an optimized selection of these variants according to the problem characteristics. This is only possible if there is a causal relationship between the problem characteristics and the effects of the CTV on algorithm performance. The performance has several dimensions, namely the solution quality, the required computational effort and the communication overhead. The priority of these performance dimensions may vary depending on the use case of the optimization heuristic, but the CT is likely to affect all of them. The existence of this causal relationship is the subject of the final hypothesis under investigation:

Hypothesis 3
There is a causal relationship between the characteristics of the distributed optimization problems and the effects of the communication topology variants. Thus, the communication topology variant of a distributed optimization heuristic can be selected in a

problem-dependent manner to increase the optimization performance according to a predefined prioritization of performance dimensions based on a set of distributedly computed fitness landscape metrics.

It should be possible to *learn* which CTV is most appropriate for a given problem if the assumed causal relationship between problem characteristics and the effects of CTVs exists, if the distributed set of fitness landscape metrics is appropriate for characterizing the problems, and if the model for CT adaptation is adequate. Therefore, the final research question not only aims to address hypothesis 3, but also serves to evaluate the artifacts developed in response to research questions 1 and 2. The final research question is as follows:

Research Question 3

How can machine learning models be used to ***show*** *the* ***causal relationship*** *between the characteristics of distributed optimization problems and the impact of communication topology variants, thus enabling the problem-specific selection of communication topology variants?*

Note that the focus is clearly on using machine learning models to support the idea that there is a relationship that can be learned. The goal is not to develop an optimized learning model. Therefore, the investigation of this learning capability also serves as a proof-of-concept and even an evaluation for the methods developed to answer research questions 1 and 2.

Reference to own preliminary work

Some of these aspects have already been addressed in preliminary work. In [HN20], first investigations on the effects of different static CTs on the optimization runs with the heuristic COHDA were carried out for a selected continuous optimization problem. The effects of the CTs on the algorithm performance and the correlation with certain graph properties, such as mean degree, diameter, and Fiedler eigenvalue, were examined. This work was continued in [HN21b] and [HN21a].

In [HN21b] a first version of the dynamic CT adaptation (RQ 1) was presented. Using the heuristic *COHDA* (see section 4.2.1), optimization runs were performed on a set of continuous benchmark functions with these adaptive CTs and several variants of static CTs, such as path graph, ring topology, 2-D lattice topologies, small world, and fully connected graphs. Then, based on centrally performed fitness landscape analyses, decision trees were trained to select the best CT or type of CT for different performance dimensions.

In [HN21a] the focus was on the (spatially) distributed execution of the Fitness Landscape Analysis (FLA) (RQ 2). The concept was presented and tested with several FLA metrics. In addition, the construction of composite spaces (see section 3.2) was introduced. These composite spaces can be used to easily create optimization problems with domain decomposition and different subsearch spaces based on continuous benchmark functions. The set of distributed FLA metrics was tested for its suitability as a basis for selecting appropriate static CTs tailored to the composite optimization problems.

These papers provide the essential foundation for this dissertation. However, all of the concepts and results published in these papers are extended and examined in much greater detail in this thesis, both individually and in combination. In the following section, the development objective that serves to investigate the research questions is described in more detail. Afterwards, the remainder of the dissertation's structure is outlined.

1.4 Outline: Development objective and structure of the thesis

Optimization problems in energy systems provide the motivational context for this thesis. In particular, the properties of the optimization problems such as high dimensionality, multimodality, dependencies between decision variables, and the inherent decision set decomposition are of interest. Increasing complexity and volatility highlight the benefits of concepts such as OC to implement controlled self-organization. These concepts provide the systems engineering context for this work. Despite this contextual embedding, the focus of the thesis is clearly on the algorithmic aspects, such as the influence of communication topologies on distributed optimization heuristics and the handling of decision set decomposition in the FLA. Therefore, the additional complexity that would be needed to build power system scenarios and transfer fitness landscape concepts to modeling plant flexibility is refrained from and left for future work. Instead, the studies are conducted on synthetic problem instances based on well-established benchmark functions.

A number of different artefacts need to be developed as part of the investigation of the research questions raised. These artefacts, when combined, represent the development objective. This development objective is best explained in the context of an application scenario. In the scenario, a distributed optimization problem is to be solved using a distributed cooperative optimization heuristic. The objective is to select the optimal CTV for a given prioritization of performance dimensions.

To design such a scenario, suitable optimization problems are first needed. In this thesis, continuous optimization problems are used. These are well suited because there is a wide range of benchmark functions and some aspects can be easily visualized by 3D plots. In particular, it is possible to easily create derived problems with decision set decomposition, such that different agents have different search spaces. Such problems are called composite search spaces in this thesis (see section 3.2). Second, a CT-based cooperative optimization heuristic that can handle decision set decomposition is needed. Topology variants are expected to have different effects on different heuristics. For reasons of feasibility, only two different heuristics are used in this work, namely the *Combinatorial Optimization Heuristic for Distributed Agents (COHDA)* [HS17] and the *Island Model Differential Co-Evolution (IDICE)* (which was developed as part of the thesis). If adjusting the CT at runtime can have positive effects, if these effects are problem-specific, and if it is possible to choose the CTV tailored to the problem for these two optimization heuristics, then it is at least likely that this is also true for other CT-based cooperative optimization heuristics. The heuristics are integrated into an observer-controller architecture (cf. OC). Observer and controller take over the selection of the initial CT and, if necessary, the adaptation at runtime. Figure 1.4 shows the development objective in context.

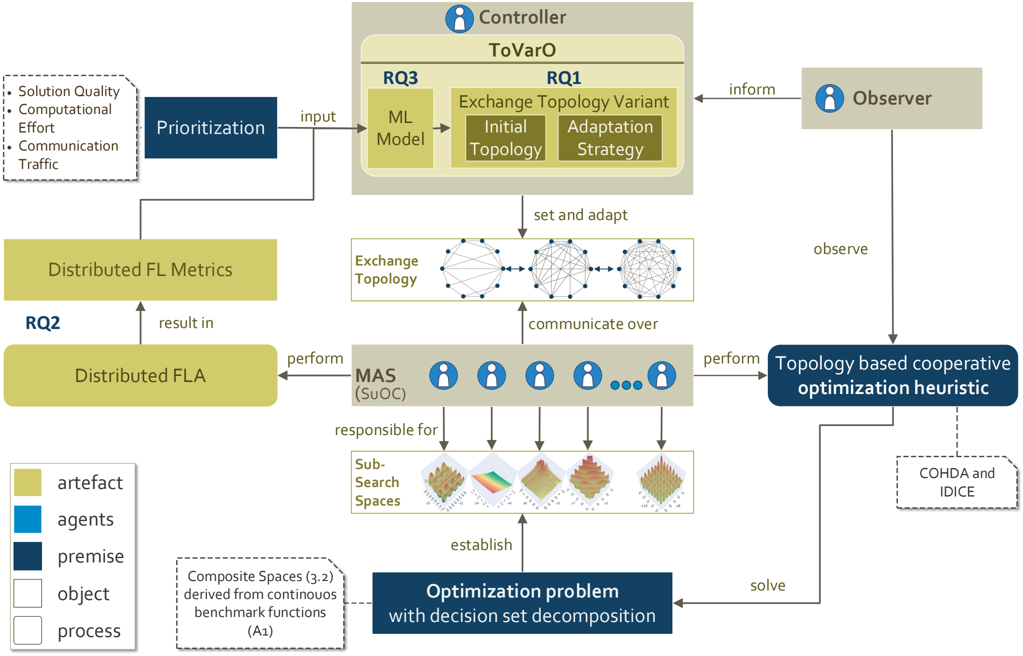


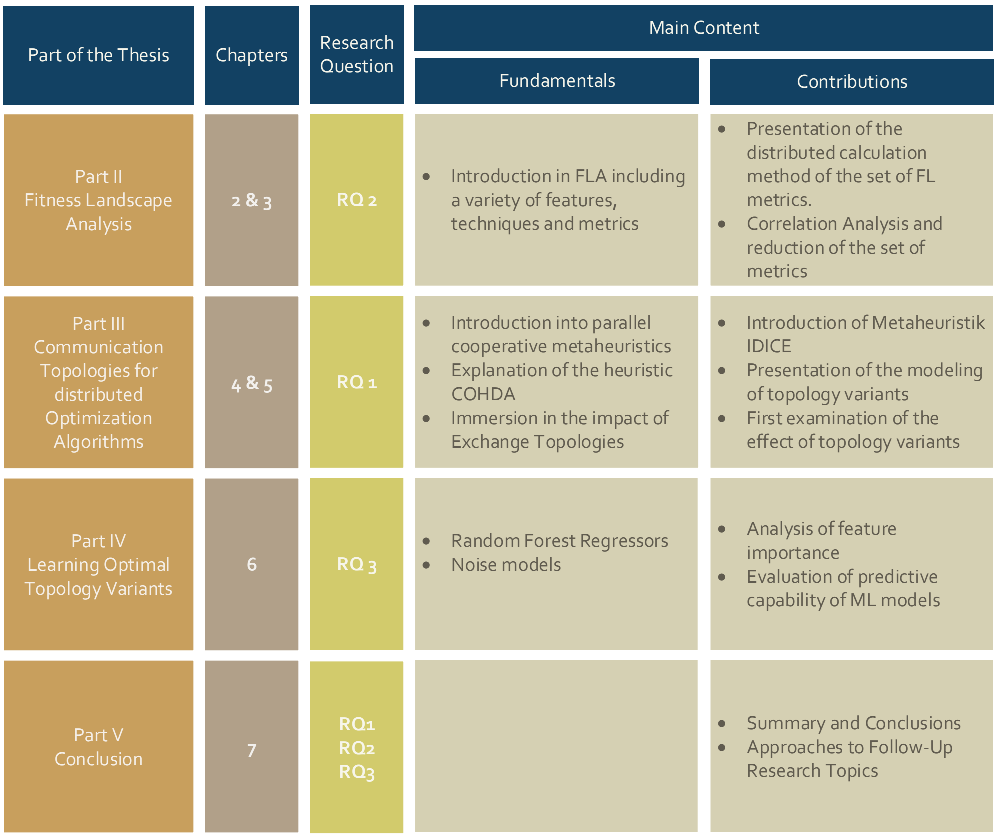
Fig. 1.4.: Development Objective: given an optimization problem, an optimization heuristic, and a prioritization of performance dimensions as premise, the agents first perform a distributed FLA for their decomposed search spaces, based on which the controller can select an initial CT and later adjust the topology at runtime if necessary (Topology Variant Optimization (ToVarO)). This serves to improve the performance of the algorithm in a problem-specific way.

The choice of the CTV needs to be tailored to the problem. Therefore, the agents must first perform the FLA in a distributed manner. The methodology for this is the first artefact of the thesis (research question 2). The set of metrics is also an artifact, since it should be composed in such a way that it provides a good overall picture of the optimization problem, but at the same time the effort should remain proportionate. This set of metrics, together with a predefined prioritization of the performance dimensions (use case specific), serves as input for the ToVarO executed by the controller. This requires two artifacts: on the one hand, the modeling of the CTV that emerges from the investigation of research question 1. On the other hand, one or more trained machine learning models that relate the problem characteristics to the most advantageous CTV for a given prioritization. Based on this, the controller can determine an initial topology and communicate it to the agents. The agents can then start the actual optimization and execute the cooperative heuristic. The observer monitors the activities during the optimization. Based on these observations, the controller can adjust the topology at runtime if necessary.

In terms of the concepts of OC, the following picture emerges: The agent-based control system represents the productive part of the considered organic system, whose main task is to execute the distributed optimization heuristic. The optimization heuristic itself can be considered as a form of distributed decision making and thus as part of the self-organization of the system. The selection and adaptation of the CT at runtime is a form of self-optimization that is handled by the Observer/Controller mechanism via ML-based decision making. At the same time, it also contains aspects of self-configuration, since the parameterization of components is modified according to high-level policies, namely the given prioritization of performance dimensions.

In order to achieve the development objective, each of the three research questions (RQ) will be the focus of one part of the thesis. The order of RQ1 and RQ2 is reversed, since the fitness landscape analysis is a prerequisite for the subsequent work and the modeling of the CTV already requires knowledge about the problem properties for an intermediate analysis of the effect of the variants. Figure 1.5 summarizes the structure of the thesis.

Each part contains a chapter on the basics and a chapter with original contributions. First, part II is dedicated to fitness landscape analysis. It is divided into a basic part introducing FLA features, techniques and metrics (chapter 2). Then, in chapter 3, the methodology for distributed computation is presented and the initial set of distributedly computed metrics is subjected to a correlation analysis to determine the reduced final set. In part III, the focus is on the algorithmic implications of communication topologies. To this end, chapter 4 first presents the basics of parallel cooperative metaheuristics. This is followed by a presentation of the two applied heuristics, COHDA and IDICE, where IDICE was developed in the context of the



Part of the Thesis	Chapters	Research Question	Main Content	
			Fundamentals	Contributions
Part II Fitness Landscape Analysis	2 & 3	RQ 2	• Introduction in FLA including a variety of features, techniques and metrics	• Presentation of the distributed calculation method of the set of FL metrics. • Correlation Analysis and reduction of the set of metrics
Part III Communication Topologies for distributed Optimization Algorithms	4 & 5	RQ 1	• Introduction into parallel cooperative metaheuristics • Explanation of the heuristic COHDA • Immersion in the impact of Exchange Topologies	• Introduction of Metaheuristik IDICE • Presentation of the modeling of topology variants • First examination of the effect of topology variants
Part IV Learning Optimal Topology Variants	6	RQ 3	• Random Forest Regressors • Noise models	• Analysis of feature importance • Evaluation of predictive capability of ML models
Part V Conclusion	7	RQ1 RQ2 RQ3		• Summary and Conclusions • Approaches to Follow-Up Research Topics

Fig. 1.5.: Structure of the remainder of the thesis, including the relevant research questions for each part and an overview of the fundamental topics and own contributions in the respective chapters.

thesis by combining well-known principles of parallel EAs. The chapter concludes by reviewing related work that has addressed communication topologies in the context of cooperative metaheuristics, highlighting the research gap. Chapter 5 introduces the concept for modeling CTVs and presents a preliminary assessment of the impact of these variants.

Part IV of the thesis integrates the concepts developed in the previous two parts. In section 6.2, the relevance of the FLA metrics and the CTV for the prediction of the performance dimensions is evaluated using statistical methods as well as machine learning models. Section 6.3 then examines the patterns that ML models learn in this setup, draws conclusions about a possible causal relationships, and finally demonstrates the model-based selection of appropriate topology variants. Furthermore, the impact of noisy data on the predictions is investigated. In the final part V, a summary of the results and conclusions is given, as well as an outlook on possible future research topics that would continue the presented work.

Part II

Fitness Landscape Analysis

This part is dedicated to Fitness Landscape Analysis (FLA). Chapter 2 presents the fundamentals of FLA, starting with section 2.1, which describes a number of fitness landscape features that are often associated with the difficulty of problems. Then, in section 2.2, concrete techniques and metrics for quantifying such features are presented. Finally, section 2.3 discusses the specifics of dynamic and coupled fitness landscapes.

Chapter 3 presents the proposed approach for spatially distributed computation of fitness landscape metrics. The specifics of transferring this approach to different types of the previously presented techniques are discussed. Finally, the set of distributedly computed metrics is subjected to a correlation analysis to obtain a final reduced set for further usage.

2

Global Fitness Landscape Analysis

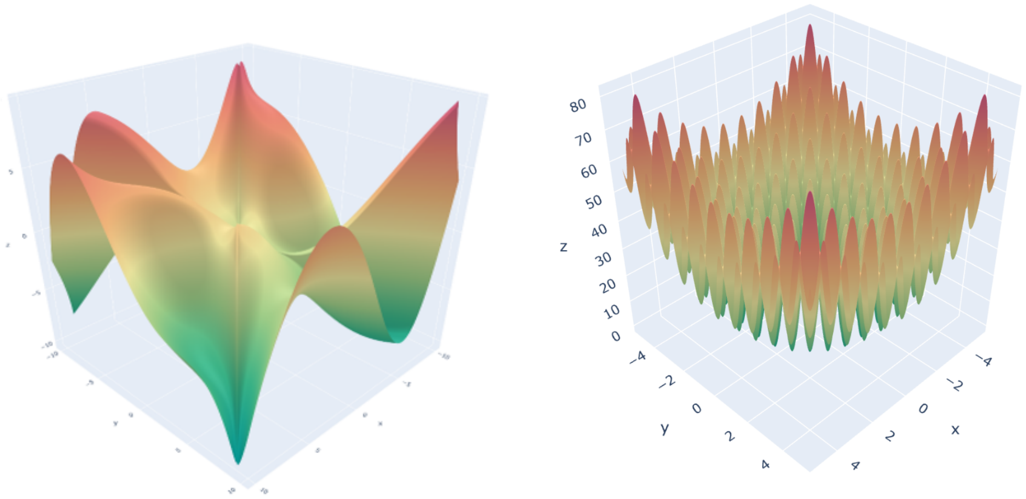
The aim of FLA is to gain a better understanding of optimization problems. It includes a large set of techniques to calculate various features that quantify different properties of a landscape, such as the level of ruggedness or modality. In many cases, the purpose is to estimate how hard a particular optimization problem is for a particular type of algorithm, and hence which algorithms are best suited for solving the problem [Mal21]. A Fitness Landscape (FL), as presented e.g. in [Tal09], is defined by the search space X and the fitness function $f : X \rightarrow \mathbb{R}$. The search space is formally defined as directed graph $G = (S, E)$, with S representing all possible solutions of the optimization problem as nodes. The set of edges E represents a notion of neighborhood $\mathcal{N}_{FLA}$[1]. An edge between two nodes indicates the two solutions are neighboring. The understanding of $\mathcal{N}_{FLA}$ may depend on a general or neutral concept of distance between solutions such as the Hamming distance or the Euclidean distance. However, $\mathcal{N}_{FLA}$ can also be characterized by the specific move operator of an algorithm. This implies that one solution can be obtained by a direct transition of the move operator from the other solution [ME13a].

Fitness landscapes for continuous problems, i.e. with continuous search spaces and continuous fitness values, often use euclidean distance measures to define the $\mathcal{N}_{FLA}$. Thus they can be described as landscapes with the search space as the bottom floor and the landscape surface being elevated according to the values of the fitness function f [Tal09]. In analogy to a geographical landscape, fitness landscapes can consist of peaks, valleys, plains, ravines, cliffs, plateaus, basins and the like. Investigating these landscape characteristics provides clues as to how difficult it is to find an optimum, i.e., the highest mountain peak (maximization) or the lowest valley (minimization). Figure 2.1 shows 3-D plots of benchmark functions for the simplified example of 2-D problems demonstrating some manifestations of such landscape features. Both their global shape and fine granular nature differ substantially. Figure 2.1a shows a multimodal landscape with deep valleys but also high peaks. At the same time, the surface of the landscape is smooth and without asperities. In contrast, fig. 2.1b shows a landscape with a global parabolic shape, making the search for the minimum seem easy for many algorithm types, e.g.

---

[1] To distinguish the neighborhood in the later context of communication topologies, the symbol will be used for neighborhoods in FLA in the following

gradient based methods. However, the search can be significantly impeded by the highly rugged surface.



(a) Rana function in the range [-10, 10] **(b)** Rastrigin in the range [-5.12, 5.12]

Fig. 2.1.: Exemplary 2D Fitness Landscapes of continuous benchmark functions

The analysis of such landscape features requires considerable effort. In [PA12], Pitzer et al. argue that fitness landscape analysis is therefore not practical for solving a single problem instance, but when a deeper understanding of an entire problem class is desired. In addition to the general properties of a problem class, the variations between different problem instances can be examined. This is also a reasonable approach to the energy system optimization problems that motivate this work. Although the analysis of the problem properties is considered as a preliminary step for the actual optimization, it will not be performed before every optimization, but only if essential parameters change.

To illustrate the structure of the following chapters, it is first necessary to clarify several terms related to FLA as they are used in this thesis. A FLA *feature* is a general characteristic of the fitness landscape. Section 2.1 introduces several features that have been shown to affect the difficulty of optimization problems. A FLA *metric* is a measure that quantifies the degree to which a feature is present in a given landscape. FLA *techniques* are used to determine such metrics. They involve some sort of sampling method and subsequent computational operations on the derived samples. The FLA techniques, including the sampling procedures and the derived metrics chosen to capture the presented features, are presented in section 2.2. Figure 2.2 summarizes the terms and their interdependencies. In addition, Section 2.3 expands the notion of fitness landscapes by introducing the concept of dynamic and coupled fitness landscapes. Aspects of this section were published in [HN21b] and [HN21a].

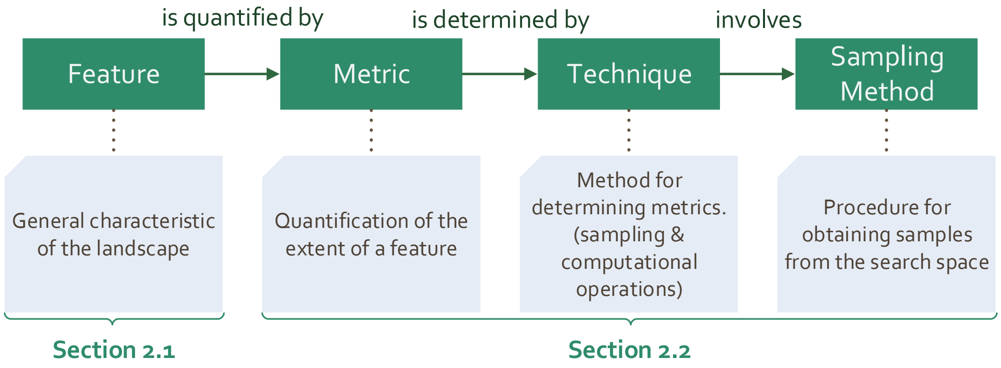

is quantified by
is determined by
involves
Feature
Metric
Technique
Sampling Method
General characteristic of the landscape
Quantification of the extent of a feature
Method for determining metrics. (sampling & computational operations)
Procedure for obtaining samples from the search space
Section 2.1
Section 2.2

Fig. 2.2.: Summary of terms used to structure FLA aspects

2.1 Fitness landscape features

The goal of various initial approaches to characterizing optimization problems was to divide them into easy and hard to solve problems, using only a single measure [ME13a]. He et al. [He+07] proved that such a general measure of the difficulty of black-box optimization problems - should it exist - could only be computed in polynomial time if certain complexity-theoretic assumptions do not hold ($P = NP$ or $BPP^2 = NP$). One explanation for this is that while several properties of the fitness landscape affect the difficulty of an optimization problem, none of them causes the degree of difficulty by itself [ME13a]. For instance, modality is often used as an indicator for difficulty. However, there are also numerous counterexamples, such as the work of Kallel et al. [KNR01] in which they showed for both hill-climbers and genetic algorithms that multi-modality neither inevitably leads to nor is essential for difficult problems. Therefore, it is generally assumed that a combination of several properties of a fitness landscape determines the degree of difficulty in solving the optimization problem for a given algorithm [ME13a; Sun+14; PA12].

This section presents features of fitness landscapes that have been shown to influence the difficulty of optimization problems. The collection is by no means exhaustive, but aims to cover a wide range of characteristics that, when combined, can give a good overall picture of the landscape. Features that can only be determined for a particular search algorithm, are not considered. The features and their descriptions are mainly taken from [ME13a] and [Sun+14]. In [ME13a], Malan et al. first present an overview of the characteristics of fitness landscapes and then give a survey of different FL techniques and link them to the characteristics presented. In [Sun+14], Sun et al. describe several basic features of fitness landscapes and argue

---

[2] BPP: bounded-error, probabilistic, polynomial time

that these features must be considered when selecting a set of metrics to properly characterize a landscape.

2.1.1 Degree of variable separability

The decision variables of an optimization problem are separable if there is no interaction between them and hence the problem can be divided into subproblems that can be solved independently:

$$arg_x \; min \; f(x) = (arg_{x_1} \; min \; f(x), \ldots, arg_{x_m} \; min \; f(x)) \tag{2.1}$$

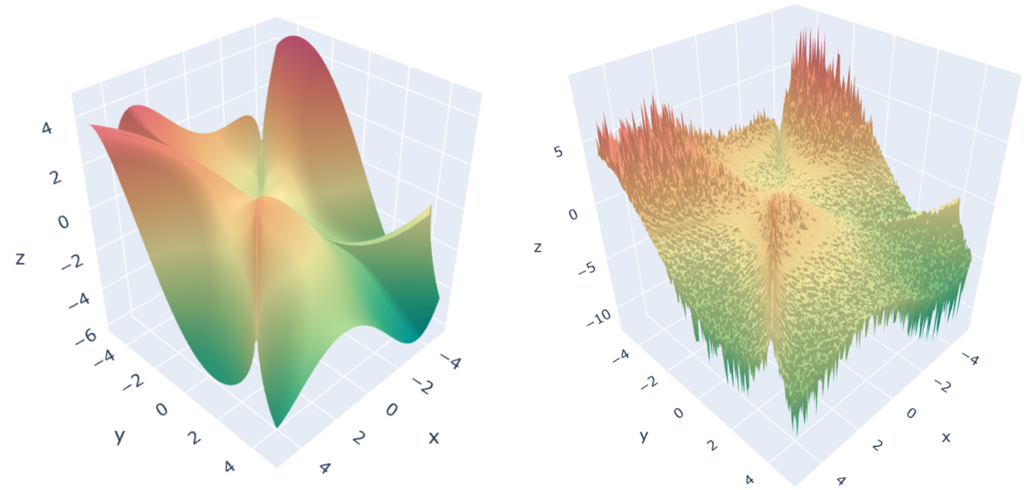
where $x = \langle x_1, \ldots, x_D \rangle$ is a decision vector of D dimensions and $x_1, \ldots, x_m$ are disjoint sub-vectors of x and $2 \leq m \leq D$ [Li+13]. If all sub-vectors $x_1, \ldots, x_m$ are 1-dimensional and thus $m = D$, the function is fully separable, else it is partially separable. When decision variables are non-separable, they interact with each other, which means that changing the value of one variable has an effect on how the values of another variable alter the outcome. In the case of complex problems, the interaction can take many different forms, resulting in different orders of non-linearity. For some classes of algorithms, e.g. genetic algorithms, studies demonstrated that linearly separable functions are easier to solve than non-linearly separable functions [ME13a].

The degree of variable interdependence can be related to the concept of *epistasis* from genetics. *Epistasis* describes the impact one gene has on the effect of another gene [PA12]. Various measures of epistasis have been developed as part of research in genetic algorithms, e.g. epistasis variance [Dav90], bit-wise epistasis [FRP98], graded epistasis and graded epistasis correlation [Roc97]. Originating in the field of genetic algorithms, these techniques assume binary representations. However, there are also techniques for determining the degree of interaction of variables for continuous problems as in [Wai+19; WMZ20] or [SKH16]. The relevance of variable interaction in the context of distributed optimization is elaborated in section 2.3.

2.1.2 Noise

In real optimization problems, the same choice of input parameters does not always lead to the same result. This may be due to incomplete knowledge or control of the system or inaccuracies in the measurement. In theoretical optimization problems, this property is modeled by noise, i.e., by adding randomness to benchmark functions. Figure 2.3 shows a segment of the Rana function with and without noise.

Fig. 2.3.: Rana function in the range [-5, 5]



(a) Noiseless

(b) With high resolution gaussian noise

In [LK95], Levitan and Kauffman showed that noise can have both negative and positive effects on search performance when using hill-climbing algorithms. For FLA, noise could be quantified by resampling data points multiple times and calculating some measure of difference such as the variance or standard deviation [ME13a]. The effects of noise on an optimization algorithm need to be evaluated to ensure its practicality for real-world problems.

2.1.3 Characterization of fitness distribution

The fitness distribution of a landscape can be viewed in two different ways. First, the fitness distribution denotes the frequency with which each fitness value occurs. This is estimated for example in the density of states technique [REA96] or the *fitness variance* from [WMZ20]. A wide distribution of fitness values may indicate a harder problem, while a low variance could imply that probabilistic methods perform equally well as more sophisticated ones [WMZ20].

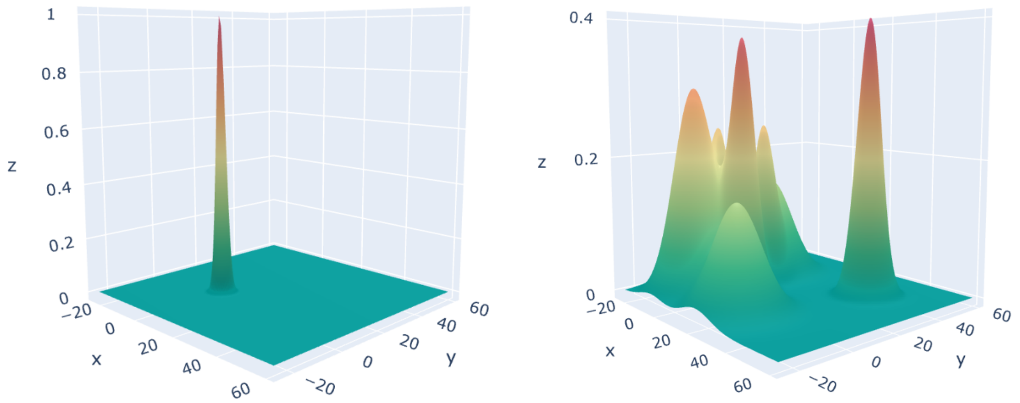
The second aspect is the fitness distribution across the search space. This differs from the first approach by additionally considering the position of the fitness values. Example techniques for this feature in binary landscapes are the *HDIL (Hamming Distance In a Level)* and *HDBL (Hamming Distance Between a Level)* measures from [BH00]. In [WMZ20], Waibel et al. presented the *state variance* as approach for continuous landscapes.

2.1.4 Modality, basins of attraction and deception

A local optimum is a point or set of points (plateau or ridge) whose fitness value is better than that of all neighboring points. For a maximization problem, this can be described by the connected set C of points with equal fitness: $\forall c_1, c_2 \in C : f(c_1) = f(c_2)$. All candidate solutions x which are neighboring points of C, are inferior to all points in C, i.e. $f(x) < f(c) \forall c \in C$. Unimodal problems have only one local and simultaneously global optimum, whereas multimodal problems are characterized by many local optima.

The number and frequency of local optima has often been linked to the difficulty of optimization problems, as search algorithms may be unable to escape inferior local optima and therefore converge prematurely. But the distribution of local optima and the size, shape and depth (or height) of their basins of attraction may be even more relevant [Ste99; KNR01; PA12; ME13a; Sun+14]. Basins of attraction are areas around the local optima where a local gradient-based search would lead directly to that optimum [PA12; Sun+14]. Local optima with relatively small areas of attraction are called isolated (the specifics of this a suspect to the chosen metric, e.g. section 2.2.3, section 2.2.4). Figure 2.5a shows an example of a *needle-in-the-haystack*-type landscape with a large neutral area (see section 2.1.6) and one isolated peak with a small basin of attraction. In contrast to this unimodal shape, fig. 2.5b shows a multimodal landscape with many local optima in a region of the search space whose basins of attraction merge. However, the global optimum is isolated here as well. Both landscapes in fig. 2.5 pose quite different challenges for search algorithms. Thus, it is likely that different, or differently parameterized, search algorithms would yield the best performance in each case. An example of FLA techniques related to modality are the so-called *local optima networks (LONs)* of Ochoa et al. [Och+08]. In this technique, key landscape features are mapped onto a graph. Further examples include the technique for estimating the number and distribution of local optima by Garnier and Kallel [GK01] and the information-theoretic technique by Vassilev et al. [VFM00] for determining the *partial information content*. With respect to basins of attraction, Merz's [Mer04] escape rate measure for estimating the size of basins can be cited as an example. However, the influence of basins of attraction can also be implicitly assessed using other FLA metrics, as the size of basins of attraction is related to both ruggedness and smoothness (see section 2.1.6), to the extent that larger basins of attraction tend to have smoother landscapes [Sun+14].

Large and deep basins of attraction of local optima combined with an isolated global optimum, as shown in fig. 2.5b, can prove fairly deceptive (as defined in section 2.1.4) for a search algorithm. The structure of the landscape guides the search away from the global optimum and towards the inferior local optima. This misleading



(a) Unimodal function with large neutral area and isolated global optimum with so called *needle-in-the-haystack* characteristic

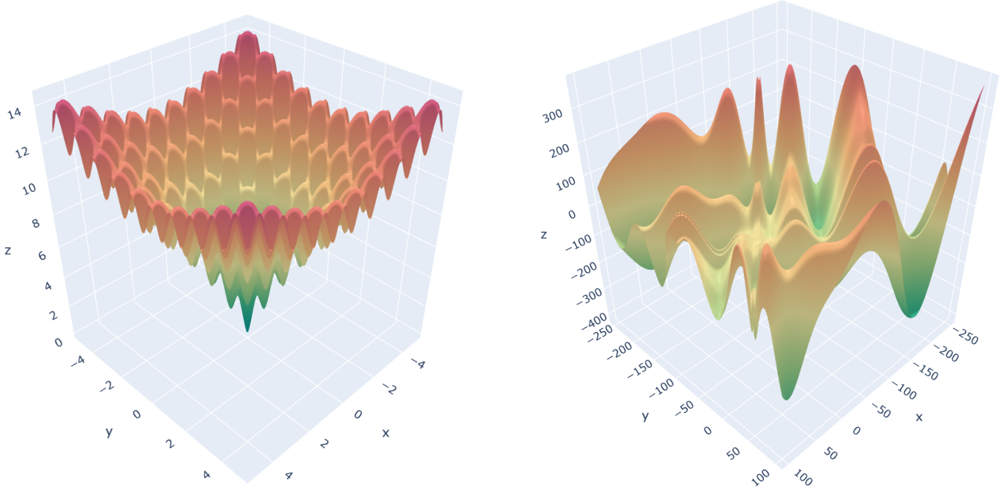
(b) Multimodal function with isolated global optimum and large possibly deceiving basins of attraction towards local optima

Fig. 2.5.: Different degrees of modality

information provided by the landscape, is perceived as deception [ME13a]. Which features of a landscape prove to be deceiving and which not depends on the specific search algorithm. A landscape that is deceiving for a Genetic Algorithm (GA) is not necessarily deceiving for Simulated Annealing (SA) or Particle Swarm Optimization (PSO). Thus, customized metrics have been developed for specific search algorithms, such as GA-deception [Gol89] and the *deceptiveness coefficient* for GAs [JTV99] or the relative size of basins of attraction (local versus global) for PSOs [XCP09]. In [Mal+14], Malan proposed gradient-based FLA metrics, also presented in [ME13b], as a way to capture the deceptiveness of landscapes for PSOs.

2.1.5 Global landscape structure

In contrast to aforementioned basins of attraction of local optima, a funnel is a global basin shape, within the regarded domain. It can therefore be understood as a cluster of local optima. The landscape is structured on the global scale by one or more funnels. Figure 2.6 shows two examples of different types of global landscape shapes. The 2-D Ackley function in fig. 2.6a clearly demonstrates the shape of a single funnel, although the landscape has a multimodal surface with many local basins of attraction. Figure 2.6b shows a section of the 2-D Eggholder function, which displays a multi-funnel shape. Just like local basins of attraction, landscapes with multiple funnels can increase the difficulty of optimization problems, as they may deceive search algorithms into basins with local optima where they might get trapped and converge prematurely [ME13a]. A well-known technique for estimating the presence of funnels in a fitness landscape is Lunacek and Whitley's dispersion metric [LW06].



(a) Ackley in the range [-4,4] as example for a single-funnel shape

(b) Eggholder-function in range [-250, 100] as example for a multi-funnel shape

Fig. 2.6.: Examples for single- and multi-funnel landscapes

2.1.6 Ruggedness, smoothness and neutrality

Ruggedness and smoothness pertain to the fitness differences between neighboring points. The neutrality of the landscape is determined by the flat fitness landscape areas [VFM02]. Hence, the features refer to the number and distribution of local optima in the search space. The fitness differences in a $\mathcal{N}_{FLA}$ can be large (rugged), small (smooth), or hardly present (neutral), with each of these surface shapes posing different challenges for optimization algorithms.

The ruggedness of landscapes has been extensively studied using Kauffmann's NK landscapes [Kau89]. The model allows to create landscapes with N decision variables and a varying degree of ruggedness defined by the parameter K. Thus, the effects of these degrees of ruggedness on optimization algorithms can be investigated. In general, rugged landscapes, such as the one depicted in fig. 2.1b, are quite difficult for optimization algorithms, as they again pose the risk of algorithms remaining trapped in local optima [ME13a]. However, smooth and neutral landscapes can also be challenging. Smoothness is related to the size of basins of attraction, as a landscape with a small number of optima and large basins of attraction is smooth. Figure 2.5b shows an example of such a smooth landscape, which can be fairly deceptive and thus challenging for a search algorithm, as discussed in section 2.1.4. The largely neutral landscape in fig. 2.5a with the isolated optimum and no clues in the landscape to guide the search towards it, is one example for the difficulties that neutrality in a landscape can cause. Neutral regions in landscapes can also be plateaus or ridges that might be misinterpreted as local optima, since the progress of a search algorithm's fitness improvement stagnates as it moves through such a region. However, as discussed in section 2.1.4, a plateau or ridge may in fact be a local or even a global optimum.

Several techniques for analysing all three features have been developed. Measures for ruggedness are e.g. autocorrelation measures [Wei90], correlation length [Lip91] and amplitude spectra [HS98]. The information theoretic technique by Vassilev et al. [VFM00; VFM02] determines measures for ruggedness (first entropic measure) and smoothness (second entropic measure). Neutrality can be measured by using neutral walks [RS01] or neutral network analysis [VPC06; Van+07].

2.2 Choosing fitness landscape techniques and metrics

The previous section presented a number of features of fitness landscapes that have been associated with problem difficulty. This section details the techniques used in this work for FLA. The selection was based on several criteria related to the presented FLA features and the requirements of the selected problem class as well as the motivating use case. Since the selection of the presented FLA features aimed at providing a good overall picture of the landscapes of optimization problems, these features should also be considered when selecting metrics and techniques. This coverage is therefore the first criterion. It is worth noting that some metrics refer to several characteristics. Furthermore, noise is not explicitly assessed by a FLA metric, but the effects of noise are evaluated in later in section 6.3.4. The second criterion is derived from the selected benchmark problems. These problems are continuous problems and thus have continuous search spaces and continuous fitness values, to which the techniques must be applicable. As third criterion the independence from specific algorithms is required. This serves to ensure the independence of the entire approach. Many FLA techniques require a distance measure. Here, a neutral measure like the euclidean distance can be used (for continuous search spaces) or the move operator of a heuristic. Depending on this, different landscapes with different properties are obtained. Just like the respective distance measure, the landscape is neutral or only applicable to a certain search algorithm. Furthermore, some techniques do not only use the move operator of an algorithm, but the sampling is done by the actual execution of the algorithm. In contrast, there are techniques where no distance measure is required and only the objective function is used. Such techniques are completely independent of a search algorithm [ME13a]. Lastly, the tradeoff between efficiently computing the full set of FLA metrics and covering the search space sufficiently by using different sampling methods must be addressed. On the one hand, the overall computational effort should be kept to a minimum, since the method is considered as a sporadically used preliminary step to the actual optimization (see the introduction of chapter 2 for further explanation). Therefore, synergies in computing different metrics for different features should be exploited. On the other hand, the result should be meaningful and usable for the intended use case. Therefore, different sampling techniques should be used to diversify the considered aspects.

The presented criteria can be summarized as follows:

1. **Covering features**: the presented features should be covered
2. **Continuous**: the techniques should be applicable to continuous landscapes
3. **Algorithm independent**: the metrics should not depend on a specific search operator
4. **Efficient**: during the calculation of the entire feature set, synergies should be used as much as possible to reduce the calculation effort (e.g. use a sample set to calculate multiple metrics)
5. **Diverse sampling methods**: different sampling methods should be used across the feature set to avoid bias caused by consistently capturing the same portion of the landscape

The following subsections present the FLA techniques and the resulting metrics selected to meet these criteria. They are organized according to the sampling techniques employed.

2.2.1 Variable interaction and sensitivity

In [Wai+19; WMZ20], Waibel et al. presented a technique for computing two measures related to the degree of variable separability using Sobol's sensitivity analysis [Sob01]. The aim of sensitivity analysis is to examine how the uncertainty in the output of a model can be attributed to different sources of uncertainty in the model inputs [Sal+10]. In the following, the basic concept of variance-based sensitivity analysis is first introduced, followed by an explanation of sampling based estimators for the computation of first order and total sensitivity indices, and finally the FLA metrics based on the indices are presented. Notation and definitions are taken from Saltelli et al. [Sal+10].

Variance-based sensitivity analysis

Sobol's sensitivity indices are based on the ANOVA (ANalysis Of VAriances) decomposition:

$$Y = f(x) = f_0 + \sum_i f_i + \sum_i \sum_{j>i} f_{ij} + \cdots + f_{12\ldots k} \tag{2.2}$$

which means that for a model with k input variables $(x_1, \ldots, x_k)$ the variance of the output is decomposed into 2^k factors. The values of the factors are obtained from:

$$\begin{aligned} f_0 &= E(Y) \\ f_i &= E_{X_{\sim i}}(Y|X_i) - E(Y) \\ f_{ij} &= E_{X_{\sim ij}}(Y|X_i, X_j) - f_i - f_j - E(Y) \end{aligned} \tag{2.3}$$

where x_i is the i-th variable and $X_{\sim i}$ denotes the matrix of all variables except x_i. Thus, $E_{X_{\sim i}}(Y|X_i)$ is the mean of Y over all possible values of $X_{\sim i}$ while keeping x_i constant. In other words, f_0 as the mean can be considered as a constant part of the output Y. The other factors are the contribution that a single variable or the interaction between several variables have on the output. The term f_i denotes the expected changes in the output due to the input x_i. Similarly, f_{ij} denotes the expected changes in output due to the interaction of x_i and x_j, excluding the effect of the two variables by themselves (f_i and f_j).

Based on these definitions, the first and the total order indices of the k variables are computed. The first order index S_i of a variable x_i indicates the amount of variation of Y caused solely by the individual variable i.e. without considering variable interactions:

$$S_i = \frac{V_{x_i}(E_{X_{\sim i}}(Y|X_i))}{V(Y)} \tag{2.4}$$

The outer variance is calculated over all possible values of x_i, while the expectation operator varies all possible values of $X_{\sim i}$ while x_i remains fixed (as in eq. (2.3)). In theory, for a fully separable problem $\sum_{i=1}^{k} S_i = 1$, since the entire variance of the output value could be explained by the effects of variations in individual variables' values without any interaction effects.

The total order index S_{Ti} measures the total effect of a variable x_i, comprising first and higher order effects:

$$S_{Ti} = \frac{E_{X_{\sim i}}(V_{x_i}(Y|X_{\sim i}))}{V(Y)} = 1 - \frac{V_{X_{\sim i}}(E_{x_i}(Y|X_{\sim i}))}{V(Y)} \tag{2.5}$$

As indicated, the total order effect of x_i can also be interpreted in the following way: $V_{X_{\sim i}}(E_{x_i}(Y|X_{\sim i}))$ indicates the whole share of the variance of Y related to $X_{\sim i}$ and thus unrelated to x_i. Subtracting this from the total variance $V(Y)$ gives the total contribution of x_i.

Sampling based estimators

Since the calculation of first and total order indices requires the calculation of integrals over integrals and can therefore be very difficult, sampling-based approaches are usually used. The basis consists of two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ independently sampled with a Low Discrepancy Sequence (LDS), such as Sobol's quasi-random sequence [Sob67] or latin hypercube sampling [Ste87]. The entries in $\boldsymbol{A}$ and $\boldsymbol{B}$ are denoted by a_{ji} and b_{ji}, where index i ($1 \leq i \leq k$) belongs to a variable in x and index j ($1 \leq i \leq N$) indicates the number of the sample in N simulations. Various formulas have been developed to calculate the sensitivity indices on the basis of $\boldsymbol{A}$ and $\boldsymbol{B}$. Here, the formulas proposed by Saltelli et al. in [Sal+10] are used. The matrix $\boldsymbol{A}_B^{(i)}$ shares the same columns as $\boldsymbol{A}$, except for the i-th column, which is taken from $\boldsymbol{B}$.

The first order index $\boldsymbol{S_i}$ can be computed using:

$$V_{x_i}(E_{X_{\sim i}}(Y|X_i)) = \frac{1}{N}\sum_{j=1}^{N} f(\boldsymbol{B})_j \, (f(\boldsymbol{A}_B^{(i)})_j - f(\boldsymbol{A})_j) \tag{2.6}$$

And the total order index $\boldsymbol{S_{Ti}}$ can be computed using:

$$E_{X_{\sim i}}(V_{x_i}(Y|X_{\sim i})) = \frac{1}{2N}\sum_{j=1}^{N} (f(\boldsymbol{A})_j - f(\boldsymbol{A}_B^{(i)})_j)^2 \tag{2.7}$$

FLA metrics based on Sobol indices

Waibel et al. [Wai+19; WMZ20] proposed two FLA metrics based on the first and total order sensitivity indices: the degree of variable interaction v_{inter} and the coefficient of variation in variable sensitivity v_{cv}.

The metric v_{inter} describes how much the fitness value is varied due to interaction effects between multiple x_i

$$v_{inter} = 1 - \sum_{i=1}^{k} S_i \tag{2.8}$$

As explained earlier, the sum of all first-order effects approaches 1 when a problem is completely separable. The difference between this sum and 1 thus quantifies the influence that the interaction effects of the variables have on the variance of the output. Hence, $v_{inter} \in [0, 1]$ and a small value indicates variable separability while a large value indicates a strong influence of variable interaction on the output.

The metric v_{cv} describes how sensitive the cost value (cv) of a function f (x) reacts to individual decision variables x_i:

$$v_{cv} = \frac{\sigma_{S_T}}{\overline{S}_T} \tag{2.9}$$

where σ_{S_T} is the standard deviation and $\overline{S}_T$ the mean of all total order effects S_{ti}. A large value of v_{cv} indicates that a small subset of the decision variables x_i is responsible for most of the variance in output, implying an uneven distribution of impact. In contrast, a small value of v_{cv} implies a similar level of influence by most of the variables. In the remainder of this thesis, these metrics will be referred to as Sensitivity Analysis based Fitness Landscape Analysis (SA-FLA).

Waibel et al. compared the SA-FLA metrics with several established ones and showed a significant difference between the distributions in statistical tests. They concluded that the metrics reveal novel characteristics of fitness landscapes. However, they also noted that empirical and theoretical evidence of the usefulness of the proposed metrics for FLA tasks, such as hyperparameter tuning or algorithm selection, is still missing. In the context of the work at hand, the use of the metrics for the optimization of exactly such parameters is empirically tested.

2.2.2 Fitness and state distribution

Waibel et al. in [Wai+19; WMZ20] also proposed two metrics for measuring fitness and state variance. The first is a straightforward approach, since given a sample of the search space, the variance of the encountered fitness values is computed:

$$\mu_2(y) = E(y' - \mu_{y'})^2 \tag{2.10}$$

where y' is $y = f(x)$ after mean normalization and $\mu_{y'}$ the mean of y'. A low value for this fitness variance indicates a less rugged landscape where probabilistic search algorithms might also perform adequately, while larger values indicate more challenging problems.

The second proposed metric by Waibel et al. is the state variance. For this purpose, the samples are sorted into bins based on their fitness values, similar to the process used for a histogram. Per bin b the the mean normalized distance of all samples is calculated:

$$||\overline{d}_b|| = \frac{1}{\#b} \sum_{j=1}^{\#b} \left(\frac{1}{k} \sum_{i=1}^{k} \sqrt{(||x_{b,j,i}|| - ||\overline{x}_{b,i}||)^2} \right) \tag{2.11}$$

where j is the index of samples in bin b, $\#b$ the cardinality of b, k the problem dimension and $||\overline{x}_b||$ is the normalized mean point of b. The larger the value of $||\overline{d}_b||$, the further apart lie the points in a bin, i.e. samples with similar fitness values. Waibel et al. propose to take the variance of all values of $||\overline{d}_b||$ over all bins:

$$\mu_2(||d||) = E(||d|| - \mu_d)^2 \tag{2.12}$$

where $||d||$ is the distribution of all $||\overline{d}_b||$ and μ_d its mean. They argue that a high variance of the state distribution indicates that scattering of states occurs over a wide range of cost values, while a small variance indicates that scattered solutions cluster around a particular cost value.

In the work at hand, the mean over the distribution of $||\overline{d}_b||$ is additionally taken as metric, as it could give a good overall impression of the distribution of points with similar fitness in the search space:

$$\mu(||d||) = \frac{1}{\#B} \sum_{b=1}^{\#B} ||\overline{d}_b|| \tag{2.13}$$

where $\#B$ is the number of bins. In the remainder of this thesis, the three presented metrics are referred to as Fitness and State Distribution based Fitness Landscape Analysis (FSD-FLA) metrics.

2.2.3 Dispersion and global landscape structure

As explained in section 2.1.5, a funnel is a cluster of local optima that forms a global basin shape [ME13b]. The dispersion metric (DM) of Lunacek et al. [LW06] provides insight into the global topology of fitness functions and thus indirectly allows estimation of the presence of funnels. In [ME13b], Malan and Engelbrecht proposed a normalized version to allow comparison of functions with different domain sizes. To compute the dispersion metric, a random sample $\mathscr{S}$ of length n is drawn that is uniformly distributed over the search space. From this sample $\mathscr{S}$, a subset $\mathscr{S}^*$ is determined that contains the $\mathscr{N}$ best points by fitness values. To make functions with different domain sizes comparable, the position vectors of $\mathscr{S}^*$ are normalized in such a way that the search space is scaled to [0,1]. In addition, a comparison sample $\mathscr{C}$ also of size $\mathscr{N}$ is sampled uniformly across the search space. Let $disp(\mathscr{S})$ be the average pairwise distance between normalized positions in the sample $\mathscr{S}$. Then the dispersion metric DM is defined as follows:

$$DM = disp(\mathscr{S}^*) - disp(\mathscr{C}) \tag{2.14}$$

Thus, the metric quantifies how far points with high fitness values are away from each other compared to a large uniform random sample. It yields values in the range of [-1,1]. A low value ($DM < 0$) indicates a single funnel landscape with an underlying unimodal structure. A high value ($DM > 0$) indicates a multi-funnel landscape and underlying multimodal structure.

2.2.4 Walking through unpaved landscapes

The following FLA metrics are all based on information derived by randoms walks. A random walk is a series of steps in a fitness landscape that are connected by the chosen notion of $\mathcal{N}_{FLA}$. The resulting series of neighboring sample points in the search space with their respective fitness values serves as the basis for the computation of a range of FLA metrics, such as autocorrelation [Wei90], correlation length [Lip91], the entropic measures for ruggedness, smoothness and modality [VFM00; VFM02] and the gradient based metrics by Malan et al [ME13c]. Below, the type of random walks used in this study is presented first. This is followed by a description of the walk-based FLA techniques that have been selected.

Progressive random walks

In [ME14], Malan and Engelbrecht presented progressive random walks as a sampling method for neighbouring fitness values in continuous spaces. They argue that random walks should be stochastic in nature and should not be biased by the fitness values of neighboring points. Also, a generic definition of $\mathcal{N}_{FLA}$ should be used and a wide area of the search space should be covered at an acceptable computational cost. In summary, their goal was to create a sample of the search space that is independent of algorithms and provides sufficient coverage with reasonable effort.

Simple random walks in continuous spaces that take random steps in an arbitrary direction (isotropic), tend to yield clustered sample points and thus do not result in an adequate coverage of the search space. Therefore, Malan and Engelbrecht suggest a more directed method. They describe their approach as follows:

> "A walk starts on the edge of the multi-dimensional search space, progresses in a random way, but with a bias in direction towards the opposite side of the search space. If a search space boundary is reached, the bias is changed to the opposite direction. Multiple walks are generated from different random starting positions on the outer boundaries of the search space."

The first step is to determine the starting zones for the walks. The foundation for this are the corner points. A n-dimensional search space has 2^n corner points of the form $(c_1, \ldots, c_n)$, with $c_i \in \{x_i^{min}, x_i^{max}\}$. If each dimension is extended form the given extreme point (x_i^{min} or x_i^{max}) to the midpoints of the axis range the respective starting zone around a corner is determined. This definition results in 2^n non-overlapping starting zones. The four starting zones and the directions in which the walks are progressed accordingly for a 2-dimensional search space are depicted in fig. 2.7 from [ME14]. A starting point for a walk must be in a starting zone with one dimension being an extreme point. For each starting zone a predefined number of steps (usually 1000) is taken in the direction of the opposite corner. A step is determined as described in algorithm 1.

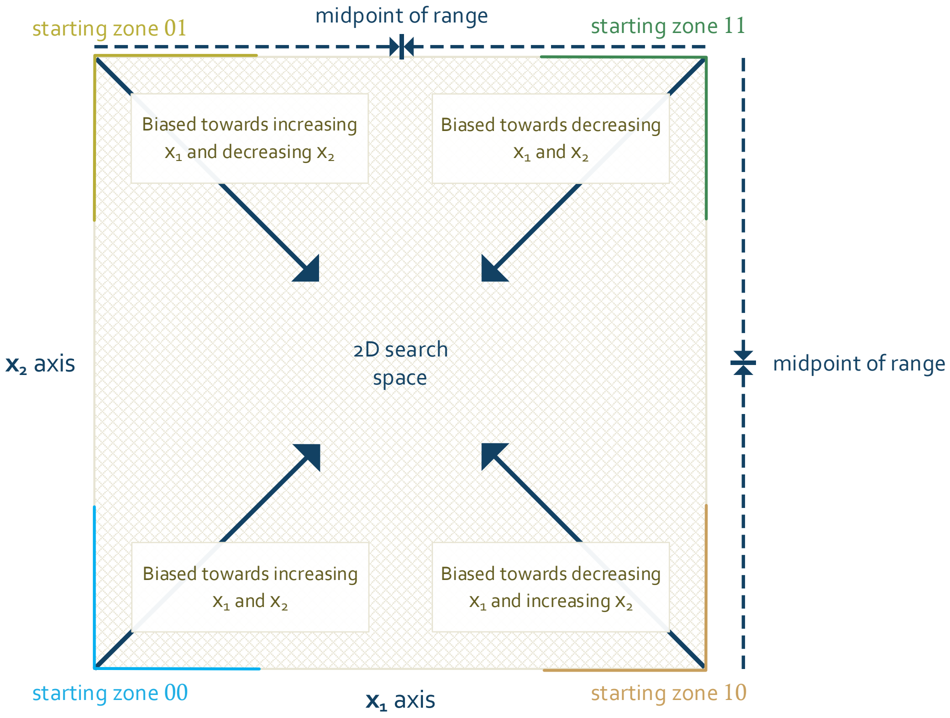

starting zone 01
midpoint of range
starting zone 11
Biased towards increasing x_1 and decreasing x_2
Biased towards decreasing x_1 and x_2
x_2 axis
2D search space
midpoint of range
Biased towards increasing x_1 and x_2
Biased towards decreasing x_1 and increasing x_2
starting zone 00
x_1 axis
starting zone 10

Fig. 2.7.: Four starting zones for a two-dimensional search space; adapted from [ME14]

Malan and Engelbrecht showed that the resulting directed, and thus anisotropic, walk provides much better coverage of the search space than simple isotropic approaches. However, high-dimensional search spaces pose a challenge as the size of the search space increases exponentially with the number of dimensions. Thus, more dimensions require more random walks to provide sufficient coverage. The ideal number of walks would be equivalent to the number of corners, and thus equal to 2^n. In high-dimensional settings, the computational effort for such a number of walks would be disproportionately high. Therefore, this number of walks should only be used for $n \leq 3$. For higher dimensional problems, Malan and Engelbrecht

Algorithm 1 Determination of one step in a progressive random walk

```
Given starting zone as binary array of size n:
    $S = (s_1, \ldots, s_n)$, with $s_i \in \{0, 1\}$
Given position of previous step:
    $step_p = (step_{p_1}, \ldots, step_{p_n})$, with $step_{p_i} \in [x_i^{min}, x_i^{max}]$
for all $i \in [1, \ldots, n]$ do
    $r \leftarrow random([0, stepsize])$                    ▷ generate a random number r
    if $s_i = 1$ then
        $r \leftarrow -r$                                   ▷ change direction if necessary
    end if
    $step_{(p+1)_i} \leftarrow step_{p_i} + r$               ▷ set value for next step in dimension i
    if $step_{(p+1)_i} \leq x_i^{min}$ then                  ▷ if step exceeds lower bound
        $step_{(p+1)_i} \leftarrow 2 * x_i^{min} - step_{(p+1)_i}$ ▷ Set to mirrored position inside bounds
        $s_i = 0$                ▷ Flip bit of starting zone to walk in opposite direction
    else if $step_{(p+1)_i} \geq x_i^{max}$ then             ▷ if step exceeds upper bound
        $step_{(p+1)_i} \leftarrow 2 * x_i^{max} - step_{(p+1)_i}$ ▷ Set to mirrored position inside bounds
        $s_i = 1$                ▷ Flip bit of starting zone to walk in opposite direction
    end if
end for
```

propose to perform n random walks, each walk starting from a different starting zone, to ensure linear growth in computation time with increasing dimensionality.

A special form of this random walk approach is used in the calculation of the gradient-based FLA metrics (see paragraph on *Gradient based measures*). The procedure for a *Manhattan progressive random walk* is quite similar, but in each step the position is only altered in one randomly chosen dimension and the step size for a dimension is always of equal size during the entire walk.

Information theoretic analysis

In [ME09] Malan and Engelbrecht adapted the entropy based measure for ruggedness, smoothness and modality that was first proposed by Vassilev et al. [VFM00; VFM02] for continuous fitness landscapes. The information theoretic technique is based on a random walk through the search space. The initial information derived by the random walk, i.e. a series of positions in the search space and the corresponding series of fitness values, is converted into a string representation. This is done with respect to the information stability measure ϵ. For this purpose, the time series is scanned from front to back. If the difference between the fitness values of two consecutive entries is less than ϵ, they are considered equal. The string representation is obtained as follows:

$$S_i(\epsilon) = \begin{cases} -1, & \text{if } f_i - f_{i-1} < -\epsilon \\ 0, & \text{if } |f_i - f_{i-1}| \leq \epsilon \\ 1, & \text{if } f_i - f_{i-1} > \epsilon \end{cases} \tag{2.15}$$

The entropy-based values for ruggedness, smoothness, and modality can be determined for the resulting string $S(\epsilon)$. To do so, the difference between the adjacent values in $S(\epsilon)$ is analyzed.

For ruggedness, the differences between the successive values are of interest. The corresponding entropy value is calculated for the resulting string $S(\epsilon)$ as follows:

$$H(\epsilon) = -\sum_{p \neq q} P_{[pq]} log_6 P_{[pq]} \tag{2.16}$$

where $P_{[pq]}$ is the frequency of occurrence of the block $[pq]$ in $S_i(\epsilon)$ with $p, q \in \{-1, 0, 1\}$. This entropy is calculated with various ϵ between 0 and ϵ_{max}, which is the value at which the resulting string consists only of zeros. The maximum of all attained entropy measures is taken as the final result [ME09]. This entropy measure reflects the information content of the random walk. This is naturally high for a rugged landscape, whereas a smoother landscape has a smaller entropy value. Ruggedness refers to fitness differences in a $\mathcal{N}_{FLA}$ and therefore depends heavily on the notion of $\mathcal{N}_{FLA}$ embodied in a landscape. This is reflected by the stepsizes in the random walks. Thus, depending on the step size, ruggedness can be viewed on different scales.

Malan suggested to compute the metric once based on random walks with a maximum step size of 1% of the search space and once with 10%. The resulting metrics FEM_{micro} and respectively FEM_{macro} (First Entropic Measure as defined by [VFM02]) are values in $[0, 1]$ and reflect the relationship between ruggedness and neutrality on micro and macro scale.

Similarly to the FEM a second entropy measure can be calculated that estimates the smoothness of the function. Based on the string $S(\epsilon)$ the entropy of smooth blocks, i.e. two consecutive characters with the same sign, is calculated as follows:

$$h(\epsilon) = -\sum_{p = q} P_{[pq]} log_3 P_{[pq]} \tag{2.17}$$

The same approach as for FEM is applied by calculating $h(\epsilon)$ with different values for ϵ and keeping the maximum of all entropy values as SEM (Second Entropy Measure). The SEM_{micro} and SEM_{macro} (Second Entropy Measure) are again values in $[0, 1]$, but refer to the relation of smoothness and neutrality of the landscape [VFM02].

The *modality* of a function corresponds to the number of local optima. Modality and ruggedness are closely related, since a rugged landscape may also include many local optima. In [VFM00], Vassilev et al. also proposed a metric to quantify the modality of the random walk encoded by eq. (2.15). A new string $S'(\epsilon)$ is constructed by removing all zeros from $S(\epsilon)$ and reducing sequences of equal characters to one character. Thus, $S'(\epsilon)$ contains only information that is essential with respect to modality. The resulting modality measure is called partial information content (PIC) and is given by

$$PIC(\epsilon) = \frac{\mu}{n} \tag{2.18}$$

where n is the length of $S(\epsilon)$ and μ the length of $S'(\epsilon)$. If the random walk encountered a landscape with high modality, the length of $S'(\epsilon)$ is almost the same as that of $S(\epsilon)$, resulting in $PIC(\epsilon)$ being close to or equal to one. If the path is flat or only leading in one direction $PIC(\epsilon)$ tends to or is equal to zero. For PIC, the same procedure as for FEM and SEM is used by performing random walks with varying step sizes to look at modality at different scales. The respective metrics are PIC_{micro} and PIC_{macro}.

Gradient based measures

A characteristic that might be deceiving for some algorithms is the steepness of gradients. In [ME13c], Malan and Engelbrecht proposed a technique for estimating gradients. The technique is based on a Manhattan progressive random walk (as described in the section on *Progressive random walks*). Each step in this random walk is of equal step size. Between two consecutive points, e.g. solution vectors $x(t)$ and $x(t+1)$, the gradient is calculated in terms of fitness and position. The gradient is normalized to allow comparison between functions with different domain sizes and fitness ranges, as defined in the following equation:

$$g(t) = \frac{\frac{f(x(t+1)) - f(x(t))}{f^{max} - f^{min}}}{\frac{stepSize}{\sum_{i=1}^{n}(x_i^{max} - x_i^{min})}} \tag{2.19}$$

were f^{max} and f^{min} are the best and worst fitness values encountered in the walk and x_i^{max} and x_i^{min} are the bounds of the search space in dimension i. For the resulting sequence of gradient measures, the mean G_{avg} and standard deviation G_{dev} of the absolute values are calculated. A high G_{avg} indicates steep gradients that might deceive an algorithm if they lead into local optima. A high value for G_{dev} suggests that the landscape has irregularities such as steep cliffs, straight plains, or sudden peaks. Low values of G_{dev} imply that G_{avg} provides a good estimate over gradients.

2.2.5 Summary of applied fitness landscape features

The set of presented FLA techniques covers the features of fitness landscapes previously described in section 2.1. Furthermore, all of them are algorithm independent and can be used for continuous optimization problems. Some of the techniques are based on LDS samples, some on different kinds of random walks and one on uniform random sampling. Thus, a diverse set of sampling techniques is employed. At the same time one sample set can be used for the computation of more than one FLA metric, to make the computation of the overall set more efficient. Table 2.1 shows an overview of the applied FLA techniques, the corresponding landscape characteristics and the applied sampling method. Additionally, the techniques are classified into different *feature groups* depending on how they can be computed in a distributed manner, which is described in more detail in chapter 3. The mappings are added in table 2.1 to give a complete overview of the techniques and their relation to various other aspects. SA-FLA and FSD-FLA metrics were already introduced as such in the respective sections 2.2.1 and 2.2.2. All other techniques are computed in the same distributed manner and are referred to as Structure Exploring Fitness Landscape Analysis (SE-FLA) metrics. The umbrella term SE-FLA was chosen, since these are primarily random walk-based methods that explore the surface structure of the fitness landscape step by step.

2.3 Coupled and dynamic fitness landscapes

Most fitness landscapes investigated in classic FLA are static in nature. This means that the optimization takes place in a constant environment and an optimization heuristic explores this static landscape during its search. However, some types of optimization problems lead to dynamic landscapes that evolve over time. One cause of this might be a varying objective function. Another cause may be the coupling of landscapes. In this case, multiple fitness landscapes reflecting different but coupled optimization problems interact with each other. The objective functions of the individual optimization problems are coupled in such a way that the selection of a concrete value of a decision variable in one of the optimization problems affects the objective function values of the solution candidates of other optimization problems. The concept is mostly studied in the context of coevolutionary algorithms. These algorithms mimic the coevolution of multiple species in nature. The evolution of one species influences the fitness of another species, leading to adaptation to the changing environment, which in turn may influence the first species.

FL feature	**FL metric**	**description**	**sampling method**	**feature group**
degree of variable separability	v_{inter}	Degree of Variable Interaction	low-discrepancy sequence	SA-FLA
	v_{cv}	Coefficient of Variation in Variable Sensitivity		
fitness distribution (in the search space)	$\mu_2(y)$	Fitness variance		FSD-FLA
	$\mu(\|\|d\|\|)$	Mean state distance		
	$\mu_2(\|\|d\|\|)$	State variance		
global landscape structure	DM	dispersion metric that quantifies distances between good solutions and thus indicates the presence of funnels	uniform	SE-FLA
modality	PIC_{macro} PIC_{micro}	measure of partial information content concerning *modality* on macro and micro scale	progressive random walk	
ruggedness	FEM_{macro} FEM_{micro}	first entropy based measure of *ruggedness* on macro and micro scale		
smoothness	SEM_{macro} SEM_{micro}	second entropy based measure of *smoothness* on macro and micro scale		
deception	G_{avg}	mean gradient that indicates the steepness of gradients	manhattan random walk	
	G_{dev}	standard deviation of gradients, indicates whether the landscape has large irregularities		

Tab. 2.1.: Overview of the entire set of applied FLA techniques, consisting of the features covered, the metrics calculated, and the sampling methods employed. In addition, the assignment to *features groups* is specified, which are relevant for the distributed calculation in the following section.

Figure 2.8 illustrates the concept with a simple example in which three species are co-evolving. Each species has two evolvable characteristics that correspond to the assigned decision variables. The Rana function (see appendix A.1) was chosen as an exemplary fitness function for species A. When analyzing the fitness landscape of species A, assumptions must be made about other decision variables that are not controlled by A, or the values are given by the current specification of the other species. In the figure, four examples of different decision set specifications and the resulting fitness landscapes for species A are presented. These examples illustrate the extent to which a coupled fitness landscape can transform through the evolution of interdependent species or optimization problems respectively.

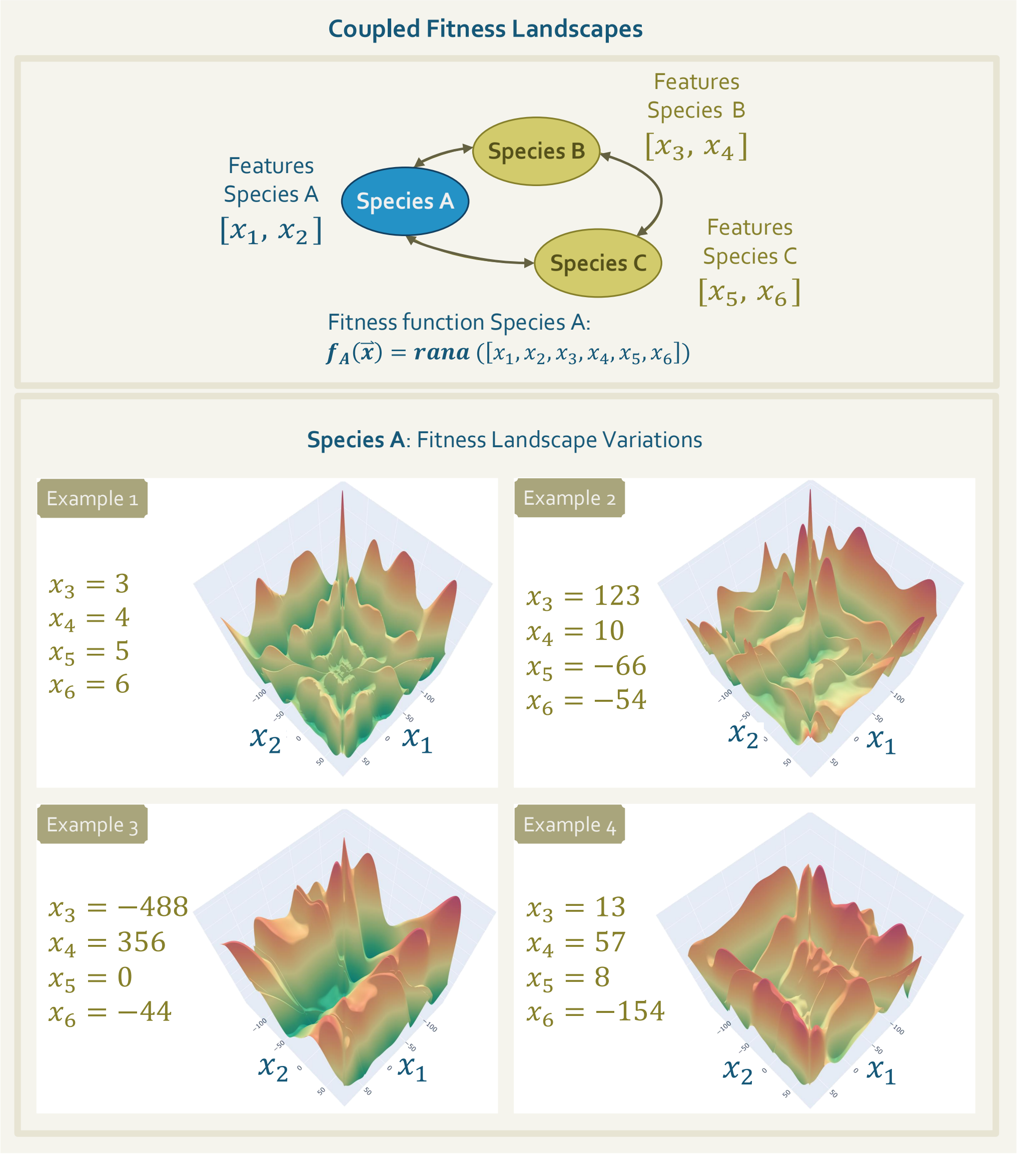


Fig. 2.8.: Simple example for coupled landscapes (see appendix A.1 for definition of used benchmark function)

Research on coupled fitness landscapes mostly targets the co-dynamics in such optimization problems. The behavior of the co-evolving species can be chaotic, lead to oscillation, or in the best case, analogous to a Nash equilibrium, lead to each species being at a peak that coincides with the peaks of the other species [HK05].

In [HK05], Hordijk and Kauffman perform a correlation analysis for so-called NKC models. These are an extension of the well-known NK models with N decision variables and a varying degree of ruggedness, defined by the parameter K. A NKC model consists of several coupled NK models, whose degree of interdependence can be specified by the parameter C. To investigate the effects of coevolution on the correlation structure of landscapes, Hordijk and Kauffman performed interlinked random walks. During a random walk in the search space of a species, every m steps the values in the search spaces of the coupled species are altered by performing steps in random walks there as well. The resulting random walk of the first species is then taken as basis for correlation analysis. One of the main findings was that intermediate or fast coevolution rates reduce the correlation of the fitness values in the FLA of a species significantly.

In [Ric06] and [Ric08], Richter used coupled map lattices for the construction of landscapes and the modeling of landscape dynamics. In this way spatio–temporal fitness landscapes are constructed. Richter analysed these landscapes regarding various FLA measures. Furthermore, he investigated the contribution of dynamic aspects, such as change frequency and dynamic severity, to problem hardness. In [Ric06], the analysis of the relationship between landscape measures and the performance of an evolutionary algorithm showed that at small change frequencies, the landscape measures modality and ruggedness were strongly correlated with algorithm performance. This correlation ceased at larger change frequencies. In [Ric08], Richter extended the approach to further FLA measures, namely information content (see section 2.2.4) and epistasis. The results confirmed the findings of [Ric06] and also showed a strong correlation between algorithm performance and information content.

In a later work [Ric14], Richter used simple and abstract models to study the relationship and codynamics between objective and subjective fitness in coevolutionary setups. The objective landscape represents the global view of the problem to be solved and thus concerns complete solutions. The subjective landscape is the coevolutionary algorithm's view of the problem and thus represents the fitness of partial solutions from a local perspective (of one species). Richter computes similarity measures between objective and subjective landscapes and argues that these are suitable for quantifying and discriminating the codynamics between these fitness types. In [De 07], de Jong suggested the *Objective Fitness Correlation* as a measure for the correlation between the objective fitness and the subjective fitness used in coevolutionary algorithms. His results suggest that a high correlation between the two landscapes is associated with an increase in objective quality, while decoupling can lead to the failure of coevolutionary algorithms.

The presented research in the field of coupled landscapes is mostly focused on better understanding the dynamics of coevolution. Furthermore, it is assumed that all information about the subsearch spaces is available at a central point. Accordingly, the employed methods for calculation of FLA features are not meant to be implemented in a spatially distributed system. They are either unsuitable for this purpose by requiring a central instance with complete knowledge [Ric06; Ric08; De 07; Ric14] or the distributed computation would be very time consuming and involve extensive communication, especially for higher dimensional problems, such as the interlinked random walks of Hordijk and Kauffman [HK05].

The concept of coupled and dynamic landscapes is also applicable to optimization problems in the real world, such as the energy optimization problems in spatially distributed systems that motivate this thesis. In such systems, optimization can be performed by multi-agent systems in which each agent controls an asset relevant to the optimization problem, e.g., energy resources, controllable loads, or electrical devices such as transformers. The agents share the same global optimization problem, but each of them controls only a small part of the decision variables. This natural decomposition of the search space leads to *coupled and dynamic fitness landscapes*.

An analysis of the properties of such distributed optimization problems should provide valuable hints for the selection and parameterization of distributed optimization heuristics. The same reasons that favor the use of distributed optimization in such a real-world problem also imply that the information needed for a FLA of the entire search space is not available in a central location. In other words, the analysis of the fitness landscape in a spatially distributed optimization setup should also be performed in a distributed manner. In the following chapter, the aspects that have to be considered for a spatially distributed computation of FLA metrics are highlighted and the solution approach developed in this thesis is presented.

3

Distributed Fitness Landscape Analysis

The spatial distribution of entities jointly performing a FLA of a common overall problem, respectively a search space, imposes its own requirements. On the one hand, spatial distribution makes communication of information an important aspect in itself. On the other hand, not all knowledge about the local search spaces can be transmitted to a central instance. This would result in a very high communication overhead. In addition, distributed optimization may be chosen for reasons such as data privacy or scalability requirements. In such a case, it would be unreasonable to collect all the information in a central location in a preparatory step for distributed optimization.

FLA techniques, as explained in chapter 2, generally rely on sampling the search space in a certain way and computing metrics based on the position of the samples and their respective fitness values. In a distributed computation, no agent knows the full search space, and thus the resulting metrics are based on limited information. The goal of this chapter is to present a distributed computation method that considers a trade-off between the expressiveness of the metrics and the computational effort. This corresponds to an investigation of hypothesis 2 and research question 2 as presented in section 1.3.3:

Hypothesis 2
Distributed fitness landscape analysis for search spaces with decision-set decomposition can determine meaningful features that can serve as the basis for parameter tuning and control of distributed optimization heuristics.

Research Question 2

---

*How can **fitness landscape analysis** be performed in a **distributed manner** while creating a reasonable tradeoff between computational and communication effort and the expressiveness of the computed features and which metrics need to be included into a set to provide a reasonable characterization of distributed optimization problems?*

To investigate this research question, a distributed FLA approach based on distributed sampling, sample recombination, and local application of FLA techniques is presented below. The resulting metrics are compared to their centrally computed counterparts to assess their meaningfulness. At the same time, the impact of sampling techniques and number of samples is examined to assess the balance between expressiveness and computational cost. In addition, a correlation analysis is performed to investigate whether the metrics capture different aspects of the search space. The usability for parameter tuning and control will be investigated later in part IV.

In the following, first the general approach to the distributed computation of FL metrics is presented in section 3.1. Afterwards, the distributed FLA is further explained for different groups of jointly computed metrics in sections 3.3, 3.4 and 3.5. Finally, the correlation analysis is performed for the complete set of metrics in section 3.7.

3.1 Distributed computation of metrics

Before developing an approach for distributed FLA, general requirements must be identified. The basic requirements are straightforward:

1. **Reasonable effort**: the entire procedure should be carried out in reasonable time and with reasonable communication and computation effort, so it may be performed as preliminary step (if the optimization problem changes significantly).

2. **Exploitable results**: The set of calculated metrics should be suitable for optimizing the parameters for initialization and adaptation of the CT on a problem-specific basis.

The basic idea behind requirement 1 was already touched upon at the beginning of chapter 2. FLA is a preparatory step before the actual optimization. Performing it every time is not reasonable. Instead, the goal is to characterize a problem class and, if appropriate, the difference between different problem instances. For the motivating use case of a distributed energy optimization problem, this would mean that the analysis is performed before the first couple of runs to obtain an initial parameter setting of the search heuristic. Later, it would be repeated when the problem changes significantly, e.g., when a sufficient number of plants are joining or leaving, or when the target values differ greatly from the original ones. Understanding of the problem class would thus grow substantially over time, revealing the overall picture more and more. How large the differences in the problem setup must be to trigger a new execution of the FLA is a question in itself. Nevertheless, the effort for the analysis should be kept within reasonable limits compared to the actual optimization. It consumes both communicative and computational resources and must be carried out during the ongoing operation of a system, whose effectiveness and efficiency it is intended to improve.

Requirement 2 includes two aspects. First, the set of selected FLA metrics should cover enough characteristics of the landscape to provide a sufficient picture of the landscape. This aspect was taken into account when selecting the FLA metrics in section 2.2. On the other hand, the distributed computation of the chosen metrics is inherently accompanied by a loss of information. Clearly, an agent cannot have complete knowledge of the search spaces of all other agents, as would be the case in a fully centralized system. The method should, however, allow to sufficiently approximate the properties of the overall optimization problem. The crucial point is that the derived features are suitable for their intended purpose, namely the optimized parameterization of the search algorithm. This will be evaluated later on in chapter 6.

Approach

The proposed approach for the distributed computation of FLA metrics of a decomposed and distributed optimization problem is straightforward. The steps are shown in fig. 3.1 along with a simple illustrative example. The basic idea is that agents first sample values for their own decision variables (step 1: **initial sampling**) and exchange this information (step 2: **dissemination**). Thereafter, each agent can perform a **local FLA** (step 3) using the samples of the other agents as additional input. The agent either uses these samples to create a complete set of samples, including its own decision variables, and calculates the local metrics on that basis. Or it recombines only the samples of the other agents and uses them as the basis for its random walks. In the latter case, recombination leads to a large number of variations of the values of the decision variables of the other agents. Assuming that the values of the decision variables in each of these combinations are fixed, the agent obtains different versions of its local search space. This corresponds to the concept of dynamic and coupled fitness landscapes (see section 2.3). The exact form of this local computation of FLA metrics and the final **feature composition** (step 4) differs depending on the feature group (see summary table 2.1 for the assignment of metrics to groups). The metrics were classified into these groups because they can be computed in the same distributed manner based on the underlying techniques, especially the sampling methods. Within the SA-FLA, each agent calculates the sensitivity indices for its own decision variables. Only the central controller obtains the actual FLA metrics based on all sensitivity indices. For SE-FLA and FSD-FLA

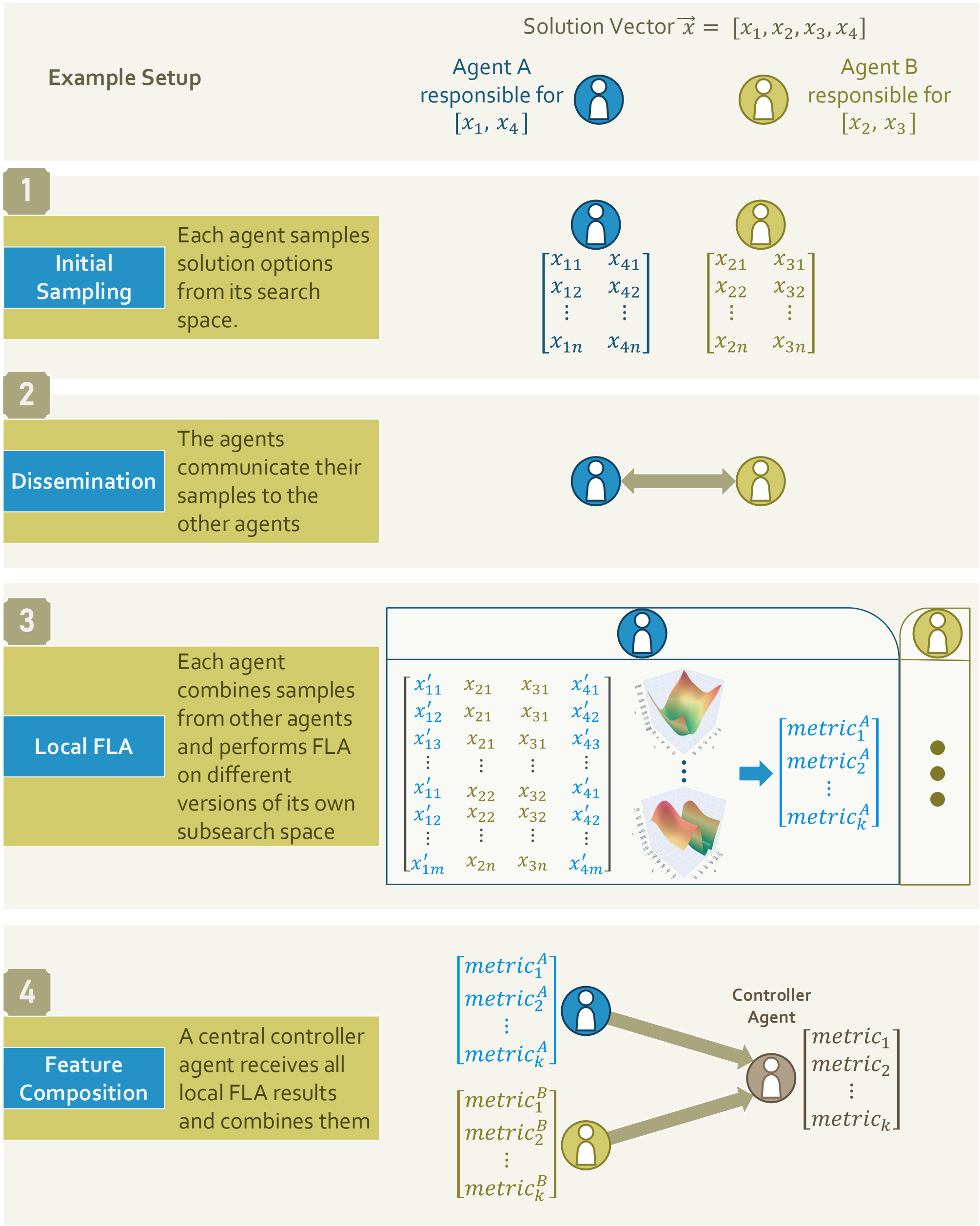

Solution Vector $\vec{x} = [x_1, x_2, x_3, x_4]$
Example Setup
Agent A responsible for $[x_1, x_4]$
Agent B responsible for $[x_2, x_3]$
1
Initial Sampling
Each agent samples solution options from its search space.
2
Dissemination
The agents communicate their samples to the other agents
3
Local FLA
Each agent combines samples from other agents and performs FLA on different versions of its own subsearch space
4
Feature Composition
A central controller agent receives all local FLA results and combines them
Controller Agent
$metric_1^A$, $metric_2^A$, $metric_k^A$
$metric_1^B$, $metric_2^B$, $metric_k^B$
$metric_1$, $metric_2$, $metric_k$

Fig. 3.1.: Distributed FLA approach with simple example for illustration

metrics, each FLA metric is computed for the local subsearch spaces. The central controller then computes the mean and standard deviation for each metric across all subspaces. The consideration of the mean values and the standard deviations of the features allows to estimate the basic properties of the overall search space as well as to quantify the heterogeneity of the subsearch spaces. The difference between SE-FLA and FSD-FLA lies in the way an agent samples its own search space in combination with the samples of the other agents. The exact procedure for the local FLA in the individual feature groups is described in more detail in the respective sections (3.3, 3.4 and 3.5).

The approach has similarities to the approach of Hordijk and Kauffman [HK05] (see section 2.3) as the fitness landscape of one agent is investigated with different preset values for other agents decision variables. However, in the work of Hordijk and Kauffman, the values of the other variables were changed while performing the random walks and then the effect of co-dynamics on the correlation in the agent's search space was studied. In the approach presented in this thesis, the entire computation of the FLA features is performed for each set of variables. The mean values of the FLA metrics computed over all sets therefore give more of an estimate of what kind of landscape an agent encounters on average.

For the complete formulation of the approach several parameters have to be chosen. This concerns mainly the initial sampling and the local FLA. For both phases, a tradeoff between the computational and communication effort (due to the number of samples) and the accuracy of the computed metrics is to be expected. The tradeoff might be reduced by choosing a suitable sampling method.

In addition, the initial sampling must be designed in such a way that it provides a good basis for SA-FLA, FSD-FLA and SE-FLA techniques. Therefore, a method based on LDS is required, as this is a prerequisite for the calculation of SA-FLA and FSD-FLA metrics. For the second sampling for partial feature determination, a LDS-technique naturally has to be performed again for the LDS-based techniques, while it plays a minor role for the SE-FLA techniques. For this reason, only LDS techniques are used for sampling. The equilibrium properties of one of the low-discrepancy sequences employed, the Sobol sequence, require the number of samples generated to be a power of two. Therefore, only powers of two are used. Table 3.1 shows an overview of the varied parameters and the chosen value ranges.

In the following, the experimental setup is described first, specifically how the optimization problems for the evaluation of the approach are constructed. Then, for the three groups of FLA metrics, the peculiarities of distributed computation are presented and the results of parameter tuning are evaluated in each case. With the final selected parameter setting, the FLA metrics are computed for a much larger set

phase	parameter	description /impact	value range
initial sampling	1.1: sampling method	Depending on the sampling technique, the selected points in the search space vary	sobol, halton, latin hypercube (classic and centered)
	1.2: number of samples	The more samples the more accurate the FLA metrics might be. But the computational and communication overhead would increase as well.	32, 64, 128
local FLA	2.1: sampling method	Method for recombination of the samples received by other agents; It involves generating the samples in the agents own search space for LDS-based techniques. In random walk-based techniques, the recombined entries are taken as the basis for random walks.	halton, latin hypercube (classic and centered)
	2.2: number of samples	see 1.2, but without communication costs	256, 512, 1024, 1536 for LDS 32, 64, 128 combined with other local sampling technique
	2.3: number of repetitions	In random walk-based techniques, it is common to repeat the calculation of the metrics several times and finally average them. This is supposed to compensate for random effects. However, it leads to an enormous increase of the calculation effort.	10, 20, 30

Tab. 3.1.: Parameters of distributed FLA

of optimization problems. Based on this, the correlation of the metrics with each other is investigated.

3.2 Composite search spaces

For the design and evaluation of the distributed FLA and also the subsequent optimization runs, simple problem instances implementing dynamic and coupled landscapes are needed. Consequently, well-known benchmark functions for global optimization are used as synthetic problem instances (see appendix A.1)[1]. To investigate different system sizes, instances with 10 and 100[2] decision variables are employed. One agent is always responsible for two of the decision variables, resulting in either 5 or 50 involved agents. The degree to which the individual fitness landscapes of the agents are coupled depends on the specific objective function. Furthermore, for each agent, the boundaries of the search space are varied such that each agent examines a different section of the search space, resulting in a more heterogeneous setup. In order to generate a setup, Latin Hypercube (LHS) is used to determine the lower and upper bounds of each agent's search space. Then, two decision variables are randomly assigned to each agent, both of which may only be varied within the previously defined limits. In this way, for each benchmark function 20 so-called composite spaces were created.

Figure 3.2 shows examples of such composite spaces for different benchmark functions. To plot each agent's 3-D fitness landscape, a random value within the individual thresholds of each of the other agents' decision variables was chosen and treated as fixed when calculating the fitness values. The examples show that the different subspaces exhibit different landscape characteristics. Still, all agents share the same global objective. When the objective function is not separable (as is the case for most of the selected functions), the subspaces are dynamic and coupled since the selection of an agent with respect to a decision value changes the fitness values obtained for other agents.

3.3 Distributed sensitivity analysis

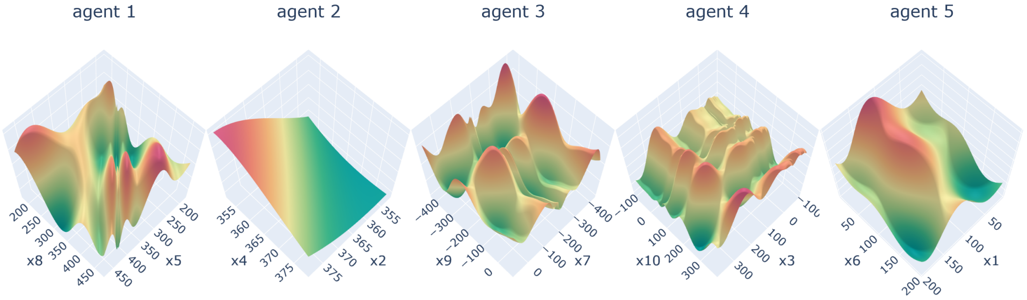
In order to calculate the SA-FLA metrics for the degree of variable separability presented in section 2.2.1, the Sobol sensitivity indices must first be determined.

---

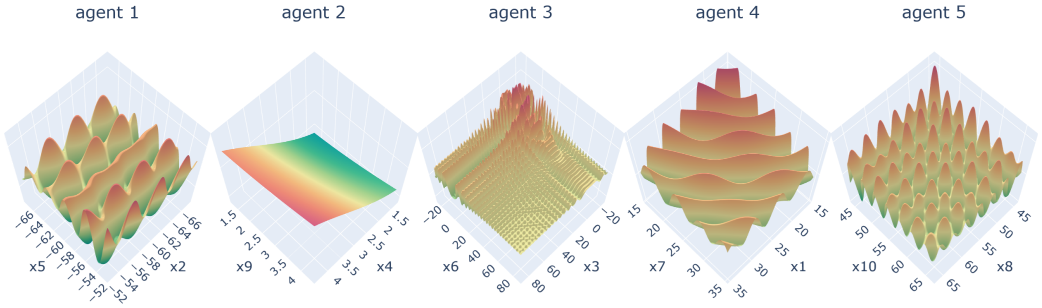
[1] For the rationale behind the selection of benchmark functions, see also appendix A.1.

[2] Problem dimension 100 is also used for subsequent heuristic optimization runs, as it has been shown in preliminary studies to provide a good trade-off between computational time and observability of the effects of CTVs. The smaller 10-D setups are used to determine if there are dimension-related effects, such as larger differences from the centrally computed metrics.

(a) Eggholder composites space



(b) Schaffer F6 composite space



(c) Xin-She Yang 1 composite space

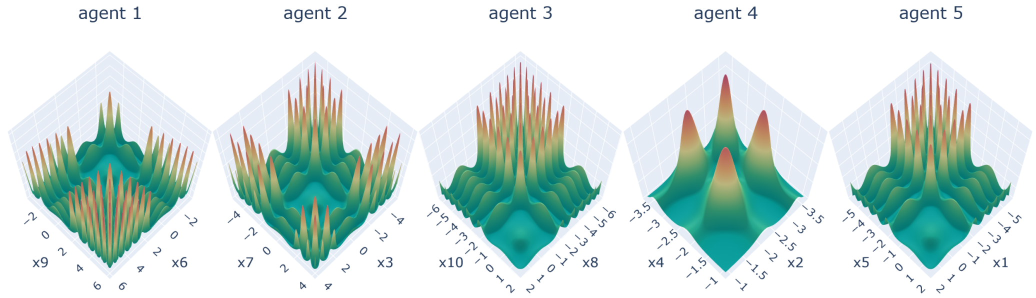


Fig. 3.2.: Examples for composite spaces with 10 dimensional benchmark functions and 5 agents handling 2 decision variables each.

These indices are based on LDS, and therefore require a sample with low discrepancy. During the distributed computation, an agent receives only a certain amount of samples from other agents, i.e., from other dimensions of the overall problem, but can extensively sample its own search space. Thus, the goal of the following evaluation is first to investigate how large the possible bias caused by this approach is and whether the distributedly computed sensitivity indices can be used as a basis for the FLA metrics.

The specific procedure for the distributed computation of the indices involves a joint recombination of the samples from other agents and the sampling of the own search space. For this purpose, the problem dimensions of other agents are considered as categorical axes for which only the transmitted values can be chosen, while the own problem dimensions are sampled in full scope.[3] Figure 3.3 shows two examples that illustrate the procedure. The two functions were developed for benchmarking in sensitivity analysis and are shown as 3-D plots [AR22]:

$$\begin{aligned} F_1 :=& f(x) = 10 * x_1 + 0.2 * x_2^3 \\ G - function :=& f(x) = \prod_{i=1}^{n} \frac{|4x_i - 2| + a_i}{1 + a_i} \quad a_i \neq -1 \end{aligned}$$

For reference, the two decision variables x_1 and x_2 were globally sampled using Saltelli's extension of Sobol' sequence [Sal02] (left figure in each case) and the sensitivity indices were calculated thereon.[4] The other two figures show the samples obtained by the two agents performing the distributed computation procedure. Each of them took 32 initial samples and generated 2048 samples each, for the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ and thus also for $\boldsymbol{A}_B^{(i)}$ (see section 2.2.1 for reference), leading to a total number of 6144 samples, which equals the number of samples in the global approach. The relatively small number of initial samples highlights the different structures of the sample sets, as the lines along each of the agents' own variable axes are clearly visible. Even with this very limited knowledge of each other's capabilities, the calculated sensitivity indices are fairly close to the reference values. Initial case studies have shown that increasing the number of initial samples significantly increases the accuracy of the obtained indices, but a low number was chosen here for illustrative purposes.

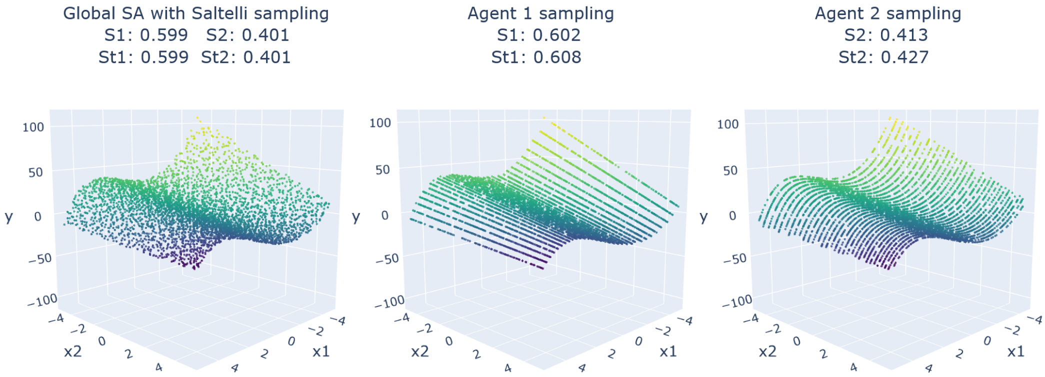
Extending the approach to a higher dimensional setup where an agent is responsible for more than one problem dimension raises a problematic aspect in the way the foreign samples are used as categorical axes. This leads to the loss of possible relationships between several problem dimensions belonging to one foreign agent. An example of such a relationship might be that the value of one variable can

---

[3] For this combined sampling the sampling library of [Hea+21] is used

[4] For reference of global sensitivity analysis the python library *SAlib* is used [IUH22; HU17]

(a) F1 function from [AR22]



(b) G-function from [AR22]

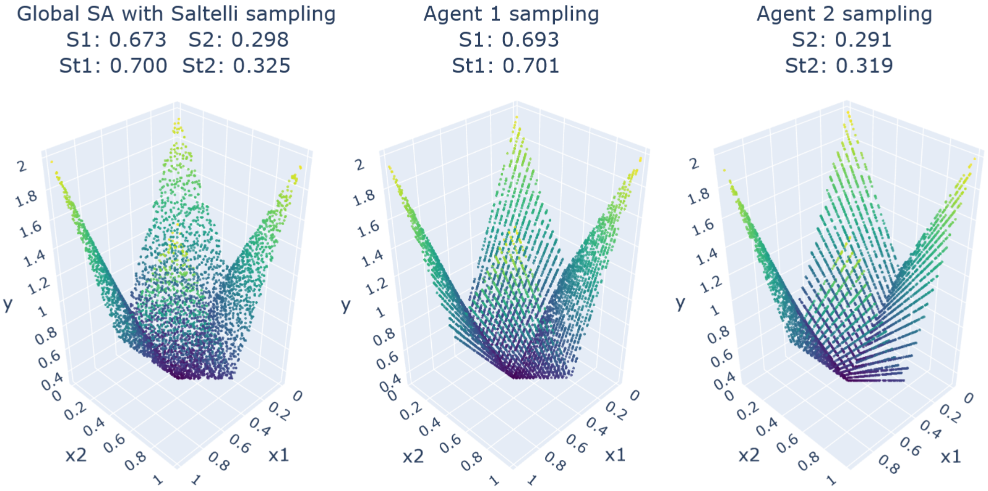


Fig. 3.3.: Examples for distributed computation of sensitivity indices: The number of samples for the global calculation is equal to that of the individual agents (6144).

only be in a certain value range if the value of another variable is below a certain threshold. This might result in using combinations of variables from other agents that may not be realistic. If this is only a small part of the total sample set, it should not significantly affect the quantification of the impact of the agents' own decision variables. Especially when applied to real world problems, this aspect will have to be considered again and the recombination of the samples will have to be modified in order to preserve relations between variables. However, the theoretical problem instances used here are not affected, since there are no such dependencies between an agent's decision variables.

Parameter tuning

The parameter tuning with respect to the SA-FLA metrics concerns the impact of sampling methods and number of samples, both in the initial sampling phase and in the local FLA, as summarized in table 3.1. The detailed analysis is provided in appendix B.1.1. In summary, the parameter evaluation showed that initial sampling has only a minor impact on the accuracy of the indices and the computational effort with the current setup. From a number of 128 samples on, no significant improvement was observed. However, the parameterization for local FLA shows a clear correlation between the number of samples and the quality of the results as well as the computational effort. Especially for higher dimensional problems, there is also a clear difference between different sampling techniques. Table B.1 in the appendix gives an overview of the findings and the chosen values for the parameters used in the further course of this work.

Globally and distributedly computed sensitivity indices

After the selection of a suitable parameterization, the analysis of the discrepancies between the distributedly computed indices and their centrally computed counterparts can be performed. Figure 3.4 shows the indices for all decision variables for one composite space per 10-dimensional benchmark function. The total order effects in fig. 3.4a have very small Root Mean Square Error (RMSE), which is also clearly visible from the centrally and distributedly calculated indices superimposed in the scatter plots. For the first order effects in fig. 3.4b this still holds for most benchmark functions. The distributedly computed first-order effects in fig. 3.4b are also very close to the reference points for most benchmark functions. The exceptions are Ackley, Salomon and Schaffer f6. Considering the 100-dimensional problem instances in fig. 3.5, the same pattern emerges. The total order effects are calculated very accurately, despite distributed computation. The first order effects are again also well captured for most benchmark functions. However, the deviations for Ackley, Salomon and Schaffer f6 increase even further. This observation suggests that these functions have one or more properties that cause this strong divergence.

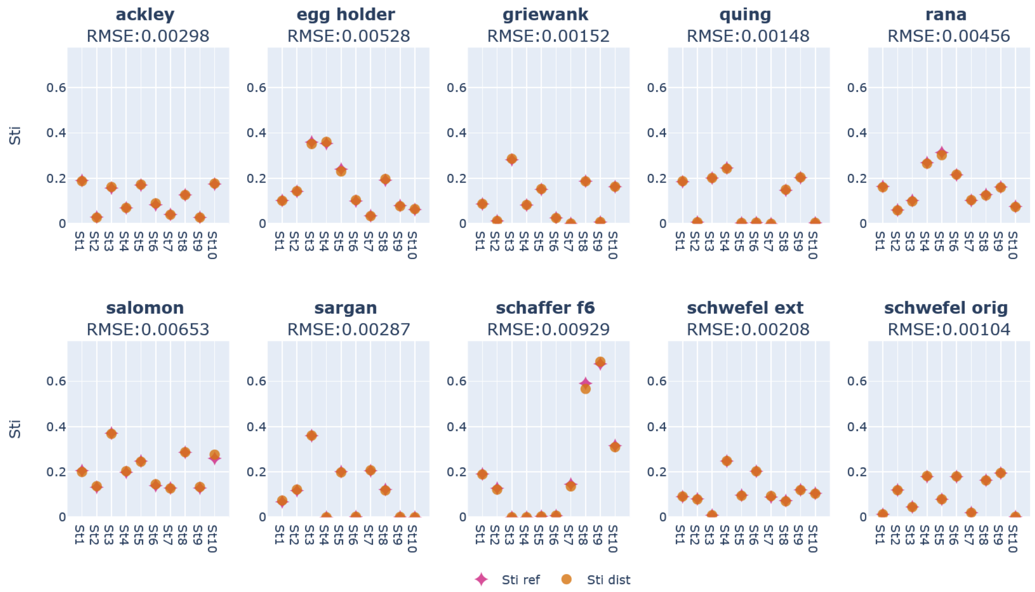
Figure 3.6 shows the $\boldsymbol{S}_{\boldsymbol{i}RMSE}$ plotted against other FLA metrics for two composite spaces per benchmark function. Plots with other FLA metrics are not shown as no correlation is apparent there. The errors seem to correlate with metrics that quantify the landscape structure on micro scale as shown in fig. 3.6a. The error is remarkably higher for landscapes which are highly rugged or show high modality and respectively low smoothness on micro scale (stepsize about 1 % of the domain). The gradient based metrics derived from walks with even smaller step sizes (about 0.1% of the domain) also exhibit a correlation with the error sizes, as shown in fig. 3.6b. This could suggest that the smaller subspaces in the distributed calculation of the first-order effect leads to an overestimation of the effects of the decision variables given that the fitness differences of the landscape at the microscale are large.

Whether the discrepancy of the S_i values is problematic for the later use of the derived metric v_{inter} remains unclear at this point. After all, the metric is largely correct for many problem instances. In other cases it may fail to indicate variable interdependence. However, high values of v_{inter} would then most likely be associated with high and strongly varying gradients and ruggedness or multimodality at the micro level. Later in the thesis, a correlation analysis between the FLA metrics is performed (see section 3.7), where v_{inter} could be discarded if it has no unique information content. Furthermore, the evaluation in section 6.2 examines the importance of the features, which could lead to the exclusion of v_{inter} if it proves to be a misleading metric for predicting algorithm performance.

3.4 Distributed computation of fitness distributions

The distributed computation of the FSD-FLA metrics regarding fitness and state distribution represents a middle ground between the way sensitivity-based FLA metrics and walk-based metrics are obtained. As with the former, a new LDS sample matrix is generated upon the samples of the other agents and the agent's own subsearch spaces. The metrics are then derived from this matrix. In order to achieve synergy, the same sampling matrix should be used as in the calculation of the sensitivity-based metrics. The calculation of the metrics, is done as described in section 2.2.2. As suggested by Waibel et al. [WMZ20], 20 bins are used to categorize the samples. If the mean distance of the points in a bin is calculated and there are not enough points in a bin, the mean distance of the bin is set to 0. Also following

(a) Total order effect S_{ti} - impact of variables x_i **including** effect of variable interaction



(b) First order effect S_i - impact of variables x_i **excluding** effect of variable interaction

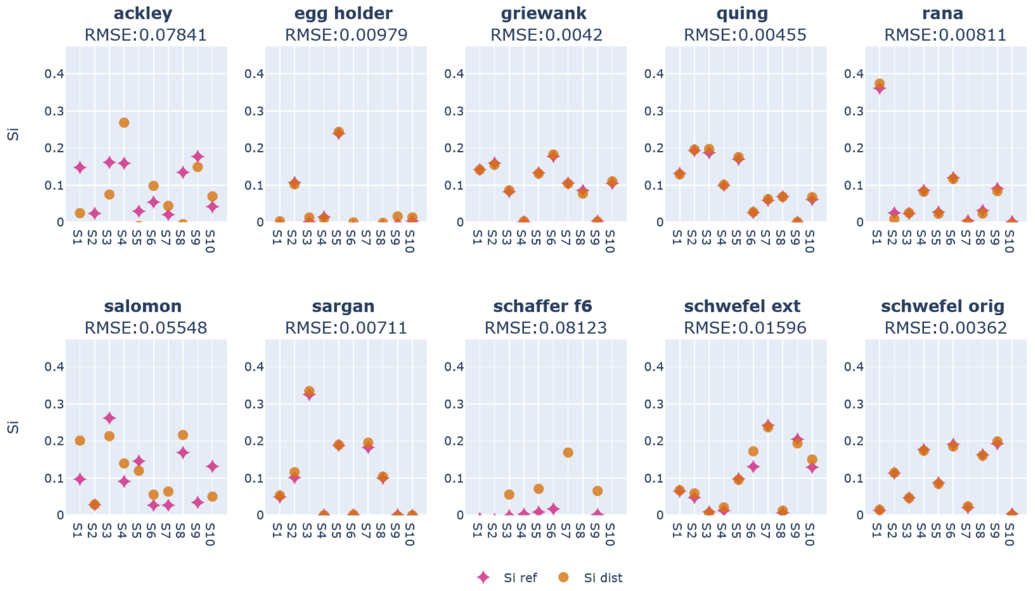
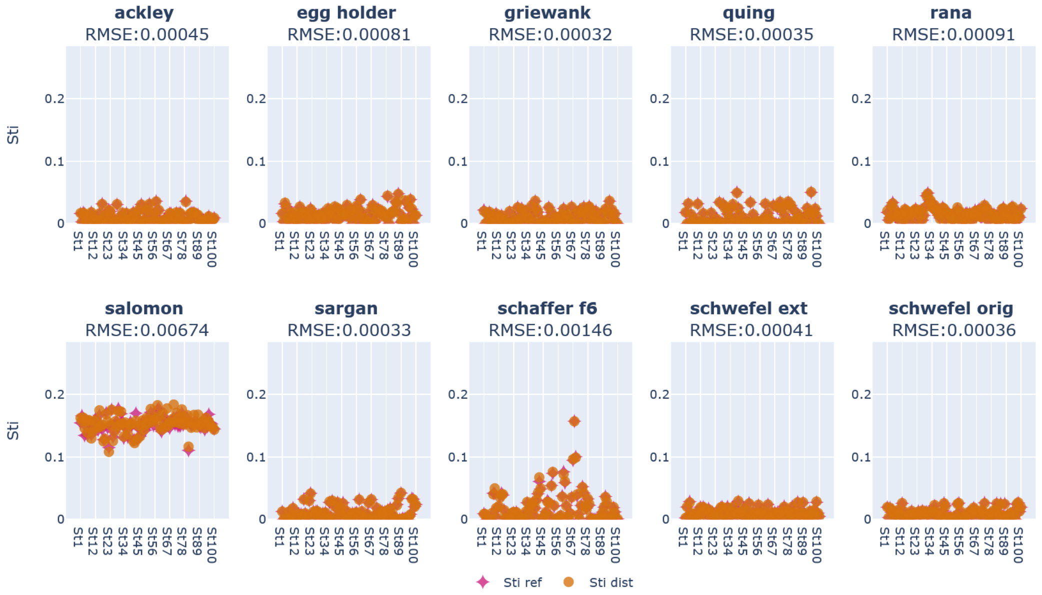


Fig. 3.4.: Comparison of globally and distributedly computed sensitivity indices for 10-dimensional benchmark functions

(a) Total order effect S_{ti} - impact of variables x_i **including** effect of variable interaction



(b) First order effect S_i - impact of variables x_i **excluding** effect of variable interaction

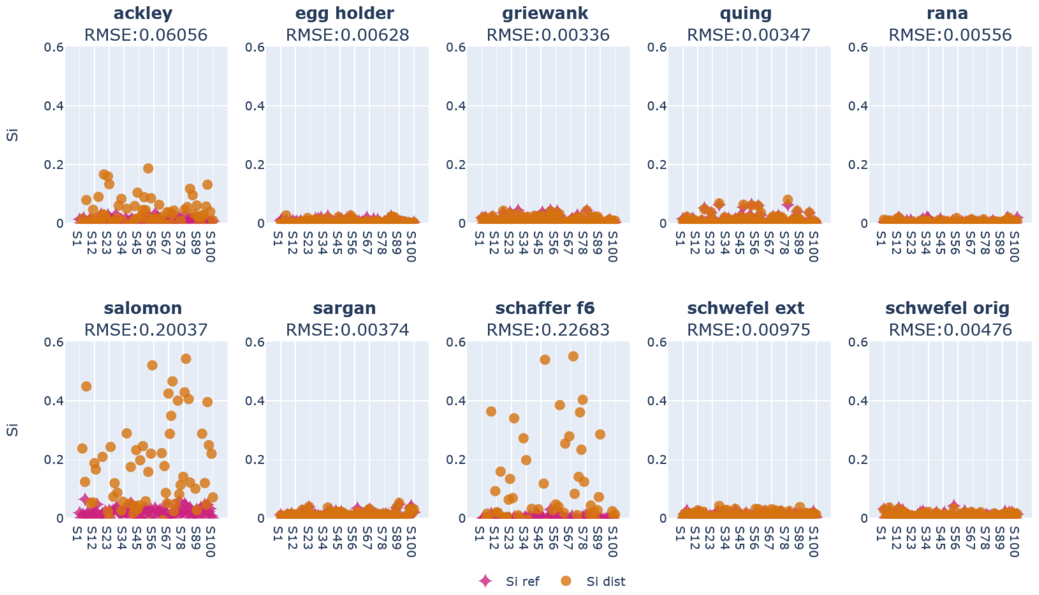
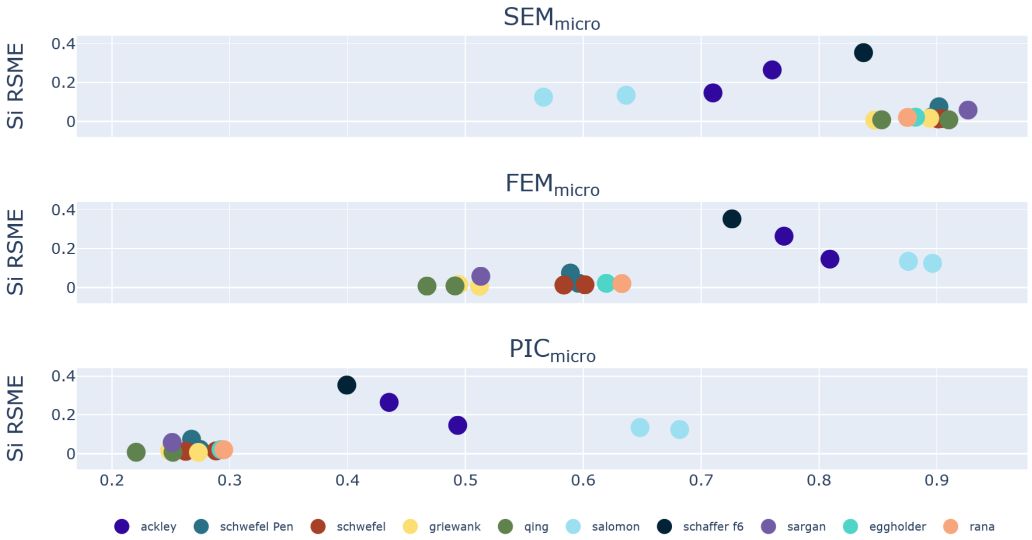


Fig. 3.5.: Comparison of globally and distributedly computed sensitivity indices for 100-dimensional benchmark functions

(a) Correlation of $S_{i_{RMSE}}$ and FLA metrics on micro scale



(b) Correlation of $S_{i_{RMSE}}$ and gradient based FLA metrics

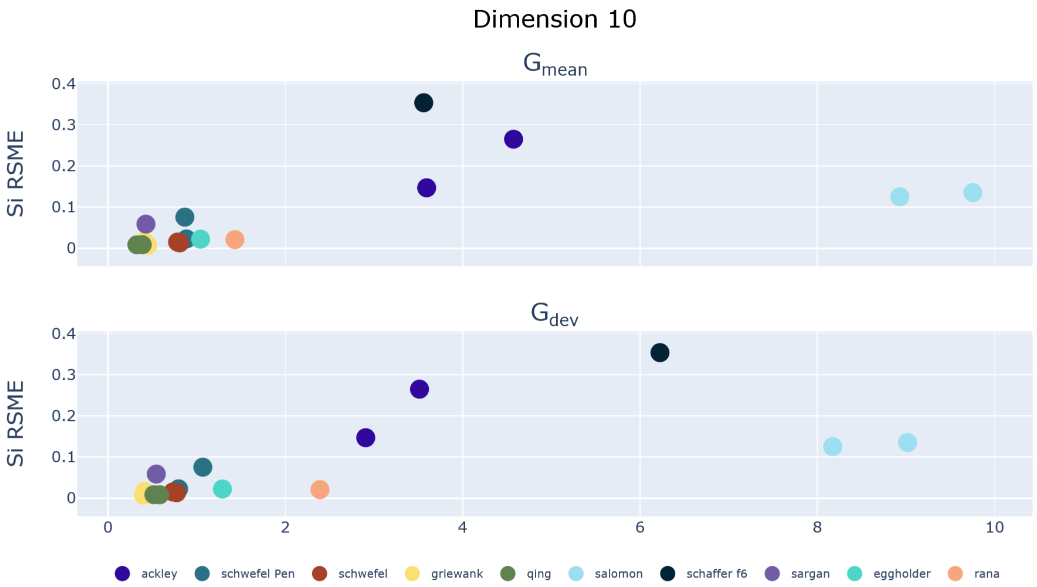


Fig. 3.6.: Correlation of RMSE for first order effects and other FLA metrics in dimension 10

Waibel et al., the threshold for a bin to be considered underpopulated is $1.5 * k$, where k is the number of samples.

During the computation of the SA-FLA metrics, the sensitivity indices are computed for each subspace individually, and only by combining all indices the FLA metrics can be computed later. In contrast, for FSD-FLA metrics, each agent already computes the FLA metric for its subspace. The combined FLA metric for the overall problem is obtained by computing the mean of the metric over all subspaces. The standard deviation is additionally computed as an indicator for heterogeneity of the subspaces. Therein lies the shared approach with the SE-FLA techniques, where the same procedure is used.

Parameter tuning

The parameter tuning with respect to the FSD-FLA metrics concerns the impact of sampling methods and number of samples, both in the initial sampling phase and in the local FLA, as summarized in table 3.1. The detailed analysis is provided in appendix B.1.2. In summary, the parameters for initial sampling showed no major impact on the deviations between distributedly and centrally computed metrics. Thus, parameters can be set according to requirements of other FLA metrics. For local FLA, the number of samples had a positive effect on the deviations of some metrics, but also resulted in a higher computational effort. This increased effort seems do be less relevant when centered LHS is used. In conclusion, centered LHS with 1024 samples appears to be a reasonable compromise that is also consistent with the setting most appropriate for distributed sensitivity analysis. Table B.1 in the appendix summarizes the findings for all feature groups and the final choice of parameters.

3.5 Distributed computation of structure exploring metrics

SE-FLA metrics comprise all FLA metrics that are not based on LDS sampling. This includes all techniques based on random walks and the dispersion metric, which uses uniform random sampling. The difference with SA-FLA and FSD-FLA is that the sampling of the agent's own search space and the recombination of other agents' samples are not performed jointly. Thus, the LDS procedure is used only for the recombination of the foreign samples. For the computation of the local FLA metrics, the matrix of recombined foreign samples is iterated over. Each combination of samples from other agents (i.e., a row in the matrix) is considered fixed, and a random walk is performed or a uniform sample is taken and the corresponding metric is computed. With each row in the recombined samples matrix, a different version of the local subsearch spaces is created and subsequently explored. After all rows have been processed, the mean value is calculated for each SE-FLA metric. This indicates which properties the agent encounters on average in its local subsearch spaces. Since each row in the matrix of recombined foreign samples is combined with either a random walk of 1000 steps or the uniform sample of more than 1000 points, much fewer samples are generated during recombination. For this reason, different numbers of samples are used here than for SA-FLA and FSD-FLA. The whole process is repeated several times, as it is common in FLA. This number of repetitions is therefore an additional tuning parameter for this feature group.

For the selection of the parameters, again the centrally computed counterparts of the metrics are used as reference. Nevertheless, there is a key difference that must be taken into account when evaluating the results. For the central computation of metrics for a n-dimensional problem, n random walks are performed (as suggested in the paragraph on *Progressive random walks*) from different corners of the search space (refer to fig. 2.7 for illustration). In each step, the position in all n dimensions is changed. For the decentralized computation, each agent is responsible for only two decision variables of the overall problem. Thus $\frac{n}{2}$ agents examine 2 dimensions each. For a walk, an agent keeps all other variables fixed. Agents perform 4 walks[5] from each of the starting zones ("00","01","10","11"), resulting in a total of $4 * \frac{n}{2} = 2n$ walks. In each walk, an agent only varies the position of its own two problem dimensions. Given the difference in dimensionality, the agents' local walks cover the examined version of the search space very thoroughly. However, they only investigate some versions of their search space. As a result, the decentrally computed features are based on a completely different coverage of the search space than the centrally computed features. It is inevitable that the derived metrics will differ. But the agents' view of the search spaces in the distributed computation is the same as they have when performing the optimization. Therefore, the distributed metrics might provide a picture that is more relevant to distributed optimization approaches. The comparison to the centrally calculated metrics is nevertheless the most meaningful criterion for the parameter selection. Therefore, the deviation between the centrally calculated reference value and the mean value of the distributedly calculated values is evaluated for each metric (analogous to eq. (B.3)).

---

[5] This is the number of walks suggested by Malan [ME14], depending on the number of problem dimensions

Parameter tuning

The parameter tuning of the SE-FLA metrics involves the same parameters as the previous two feature groups, namely the sampling method and the number of samples, for the initial sampling and the local FLA. In addition, the random walks for the local FLA are repeated a number of times to compensate for random effects, as this is a common approach in the literature (e.g., [ME13b]). The number of repetitions is therefore another considered parameter (see table 3.1). The detailed analysis is provided in appendix B.1.3. The impact of the parameters on the deviation between centrally and decentrally calculated metrics is very small for this category. Thus, for initial sampling, the sampling method and number of samples can be chosen according to other feature group requirements. For local FLA, the recombination of foreign samples is coupled with random walks. Again, the sampling method did not show much influence. Even the smallest number of samples was sufficient. Also, repeating the random walks did not provide any additional benefit. For efficient computation, sampling with only 32 recombinations without repetitions is sufficient for local FLA. Table B.1 in the appendix summarizes the findings for all feature groups and the final choice of parameters.

3.6 Summary of distributed computation of feature groups

The parallel calculation of the metrics for all three feature groups is summarized below. The goal is to clarify which calculation steps are shared for multiple feature groups in order to take advantage of synergies, and where different calculation steps are necessary. Figure 3.7 illustrates these relationships. All metrics are based on the same initial sampling, which is how agents obtain knowledge about the possible assignment of external decision variables in the first place.

The local FLA can be divided into several steps. First, the samples of external agents must be recombined to generate different variants of the assignment of all external decision variables and thus different versions of the local fitness landscape. A second step is to perform the sampling for the own decision variables and assess them in the context of these different local fitness landscapes. For the SA-FLA and FSD-FLA metrics, these two steps can be performed not only together for both feature groups, but also in a single step. For this purpose, a joint LDS sampling is performed, in which the external variables are considered as categorical axes and thus only the values transmitted by the initial sampling are used, while the full range of values is used for the own decision variables. For the SE-FLA metrics, only the external

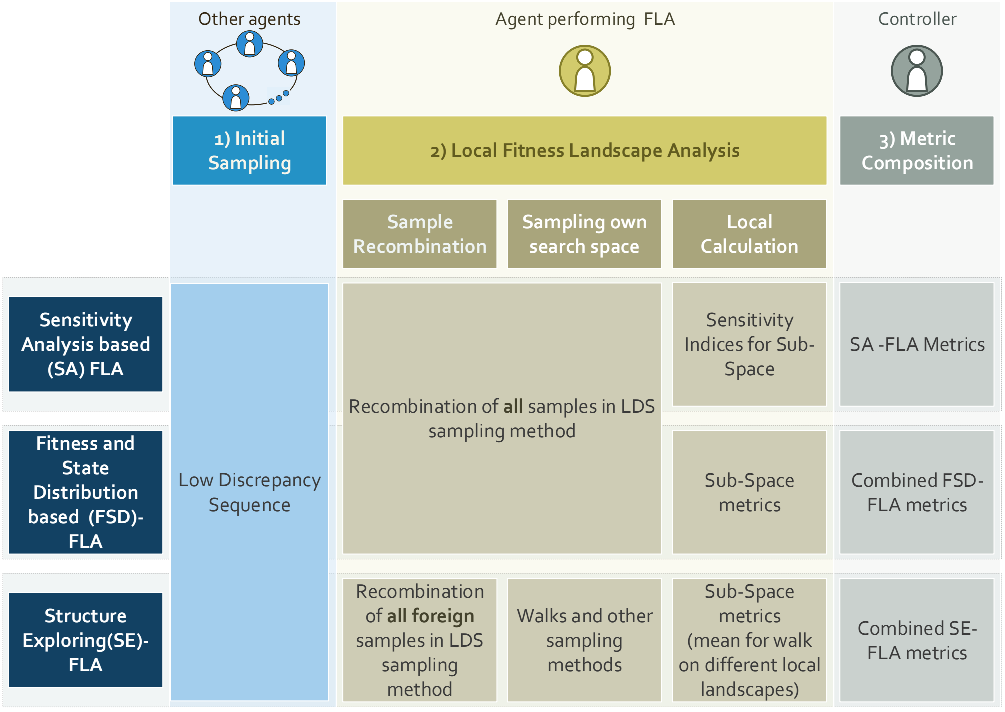

Other agents
Agent performing FLA
Controller
1) Initial Sampling
2) Local Fitness Landscape Analysis
3) Metric Composition
Sample Recombination
Sampling own search space
Local Calculation
Sensitivity Analysis based (SA) FLA
Fitness and State Distribution based (FSD)-FLA
Structure Exploring(SE)-FLA
Low Discrepancy Sequence
Recombination of all samples in LDS sampling method
Sensitivity Indices for Sub-Space
SA -FLA Metrics
Sub-Space metrics
Combined FSD-FLA metrics
Recombination of all foreign samples in LDS sampling method
Walks and other sampling methods
Sub-Space metrics (mean for walk on different local landscapes)
Combined SE-FLA metrics

Fig. 3.7.: Overview of parallel distributed calculation of all feature group metrics

samples are recombined using a LDS method. The resulting different versions of the local fitness landscape then serve as the basis for random walks and other sampling methods for the own decision variables.

The final step of the local FLA is the actual calculation of the metrics or their precursors based on the combined sampling of external and own decision variables in the previous two steps. This is where all three feature groups differ. In SA-FLA, the sensitivity indices for the own decision variables are calculated based on the LDS sample set. In FSD-FLA and SE-FLA, the actual fitness landscape metrics are calculated for the subspace. However, in SE-FLA, they are computed individually for each version of the local fitness landscape and then the mean values are calculated.

The locally calculated values for all three feature groups are then sent to the controller agent. In the final step, the controller agent calculates the mean values and standard deviations across all subsearch spaces for the FLA metrics from the FSD-FLA and SE-FLA. For the SA-FLA metrics, on the other hand, the actual calculation of the FLA metrics takes place here on the basis of the transmitted sensitivity indices.

For all three feature groups, the effects of sampling methods and the number of samples used in the initial sampling and in the local FLA were examined individually and the appropriate parameter selection was determined (see appendix B.1). Thus, for the common steps, such as the initial sampling, the parameterization that is suitable for all feature groups could be selected (see table B.1). In this way, an investigation into parameter sensitivity was carried out while at the same time aiming to achieve the greatest possible synergy effects in the calculation of the entire feature set.

However, by still using different sampling methods, the computational basis of the sample set should also be reasonably diversified. This is the trade-off between computational effort and sufficient coverage of the characteristics outlined in points 4 and 5 of the criteria for selecting the total set of metrics in section 2.2. The consideration of the individual metrics instead of the feature groups shows the continuation of this principle, since e.g. the different local fitness landscapes within SE-FLA are examined with different types of random walks. In the following section, a correlation analysis of all distributed metrics is performed to determine whether the desired variance in the information content of the calculated metrics has been achieved.

3.7 Correlation analysis of metrics

The presented catalog of metrics for the FLA was composed with the intention of covering a wide range of features that, when combined, can give a good overall picture of the landscape. Since the distributed calculation of metrics often considers the mean and standard deviation across all subsearch spaces, the number of all metrics amounts to 26. Therefore, a reasonable next step is to investigate the correlation between these metrics. The goal is to reduce the number of metrics needed for further application, but at the same time to maintain the contained information of the set. The metrics were computed using the final parameter setup displayed in table B.1 for all 100 dimensional composite spaces for all benchmark functions in set number 1 (appendix A.1.1) and set number 2 (appendix A.1.2), corresponding to a total of 460 composite spaces, with 23,000 subsearch spaces. The metrics were all rescaled using min-max normalization.

Figure 3.8 shows the resulting heatmap of the Spearman correlation coefficient along with a dendrogram representing the hierarchical clustering of features, obtained by the UPGMA algorithm [Mül11]. In addition, table 3.2 gives an overview of the determined clusters.

Several metrics, such as the dispersion metric or the coefficient of variable interaction v_{inter}, did not correlate sufficiently with other metrics to be included in a cluster. This suggests that the information provided by these metrics is unique in the set of metrics. Furthermore, there are 4 clusters with two metrics each. Clusters 9 and 10

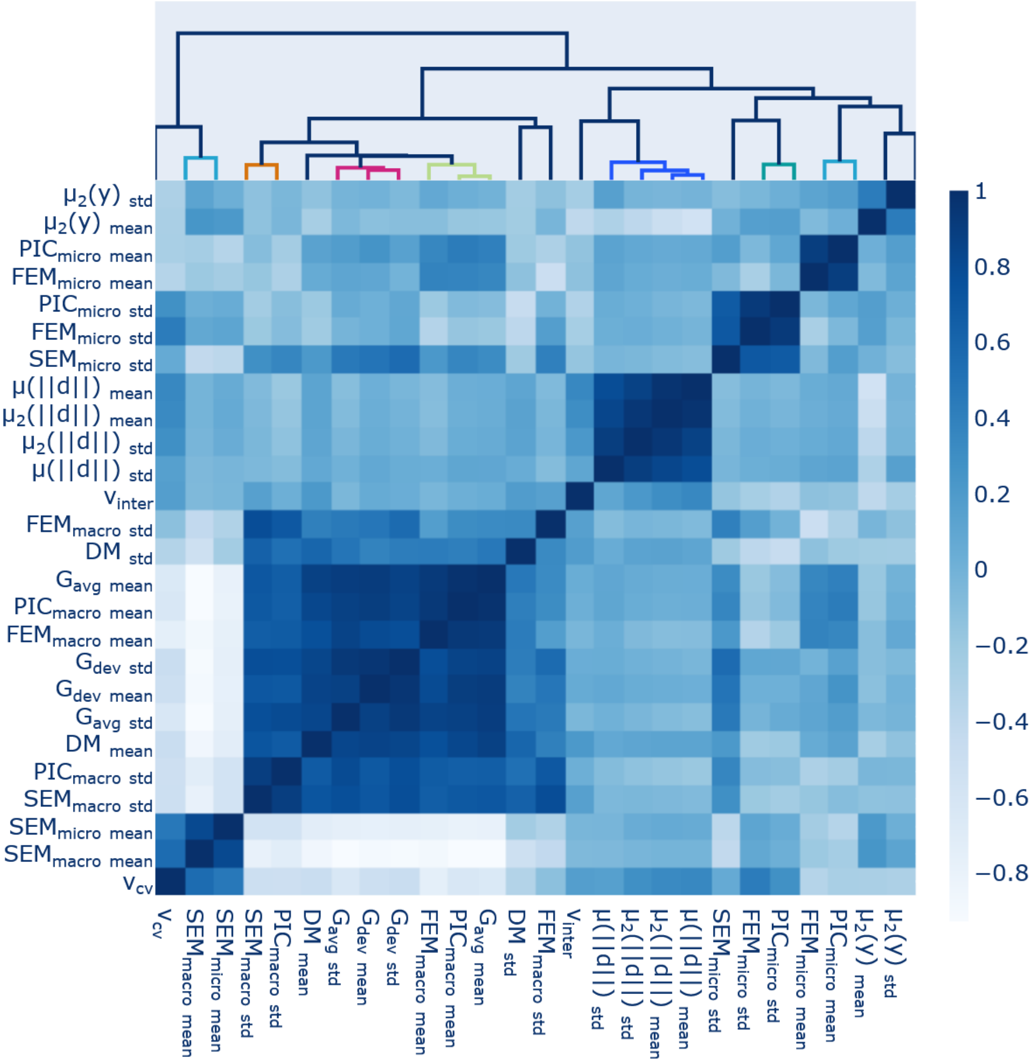


Fig. 3.8.: The Spearman correlation coefficients for all FLA metrics are shown in the heat map. The dendrogram shows the subsequent clustering with the UPGMA algorithm.

size	cluster	metrics	minimal correlation	remarks
single feature clusters	1	v_{cv}	-	Coefficient of variation in variable sensitivity
	2	v_{inter}	-	Degree of variable interaction
	3	$DM\ mean$	-	Dispersion metric mean
	4	$DM\ std$	-	Dispersion metric standard deviation
	5	$FEM_{macro}\ std$	-	Standard deviation of entropy based measure of ruggedness on macro scale
	6	$SEM_{micro}\ std$	-	Standard deviation of entropy based measure of smoothness on micro scale
	7	$\mu_2(y)\ mean$	-	Fitness variance mean and standard deviation
	8	$\mu_2(y)\ std$	-	
2 feature clusters	9	$FEM_{micro}\ mean$ $PIC_{micro}\ mean$	0.900	Mean values over all subsearch spaces of ruggedness and modality on micro scale
	10	$FEM_{micro}\ std$ $PIC_{micro}\ std$	0.928	Standard deviations over all subsearch spaces of ruggedness and modality on micro scale
	11	$SEM_{macro}\ mean$ $SEM_{micro}\ mean$	0.821	Mean values over all subsearch spaces of smoothness on macro and micro scale
	12	$SEM_{macro}\ std$ $PIC_{macro}\ std$	0.896	Standard deviations (as measure for heterogeneity) of smoothness and modality in macro scale
3 feature clusters	13	$G_{dev}\ mean$ $G_{avg}\ std$ $G_{dev}\ std$	0.879	Gradient based metrics involving standard deviations
	14	$FEM_{macro}\ mean$ $PIC_{macro}\ mean$ $G_{avg}\ mean$	0.925	Mean values for ruggedness and modality on macro scale combined with average gradients
4 feature cluster	15	$\mu_2(\|\|d\|\|)\ mean$ $\mu_2(\|\|d\|\|)\ std$ $\mu(\|\|d\|\|)\ mean$ $\mu(\|\|d\|\|)\ std$	0.791	metrics concerning state distance and variance

Tab. 3.2.: Clustered FLA metrics based on Spearman correlation coefficient

each include the metrics for ruggedness FEM and modality PIC on micro scale. The corresponding cluster on macro scale is number 14, which also includes the mean value over all subsearch spaces of the average gradients $G_{avg}\ mean$. That these metrics correlate in both mean and standard deviation seems reasonable, as multi-modality, ruggedness and higher gradients often go hand in hand. However, the correlation does not hold for the standard deviation on the macro scale. The $PIC_{macro}\ std$ is clustered with $SEM_{macro}\ std$ (nr. 12). Since the standard deviation represents a measure of the heterogeneity of the subsearch spaces, it appears sensible that this heterogeneity is similar with respect to modality and smoothness. The same measure of ruggedness, $FEM_{macro}\ std$, was not assigned to any cluster. The correlation coefficient with $PIC_{macro}\ std$ and $SEM_{macro}\ std$ is 0.706 and 0.785, respectively. As such, there is a degree of correlation between the three standard deviation metrics, but the threshold chosen by the UPGMA algorithm was too high to form a joint cluster.

Cluster 13 includes all gradient based metrics that involve some kind of deviation. $G_{dev}\ mean$ is the mean value over all subsearch spaces, for G_{dev} which is the standard deviation of gradients in one subsearch space and thus indicates whether the landscape has large irregularities. Whereas $G_{avg}\ std$ is the standard deviation across all subsearch spaces of the mean of the gradients in one subsearch space. The last cluster, number 15, contains all metrics related to state variance. Similar to cluster 13, all included metrics contain various combinations of means, standard deviations, and variances that summarize the distance of similar fitness values in the subsearch spaces for the entire composite space.

Figure 3.9 shows the values of the included metrics across all problem instances in shared scatter plots for clusters 9 to 11. The correlation of the clustered metrics is clearly visible, as the relation of the values between different problem instances is very similar. See appendix B.2 for additional scatter plots for the remaining clusters.

The closer examination of the clusters illustrates that for the differentiation and classification of different problem instances, one representative metric per cluster is sufficient. Thus, the number of metrics considered can be reduced from 26 to 15. Such a reduced set is used to proceed in chapter 6 when the FLA metrics are used as the basis for the training of machine learning models. First, however, the algorithmic aspects are considered in more detail in the following part III.

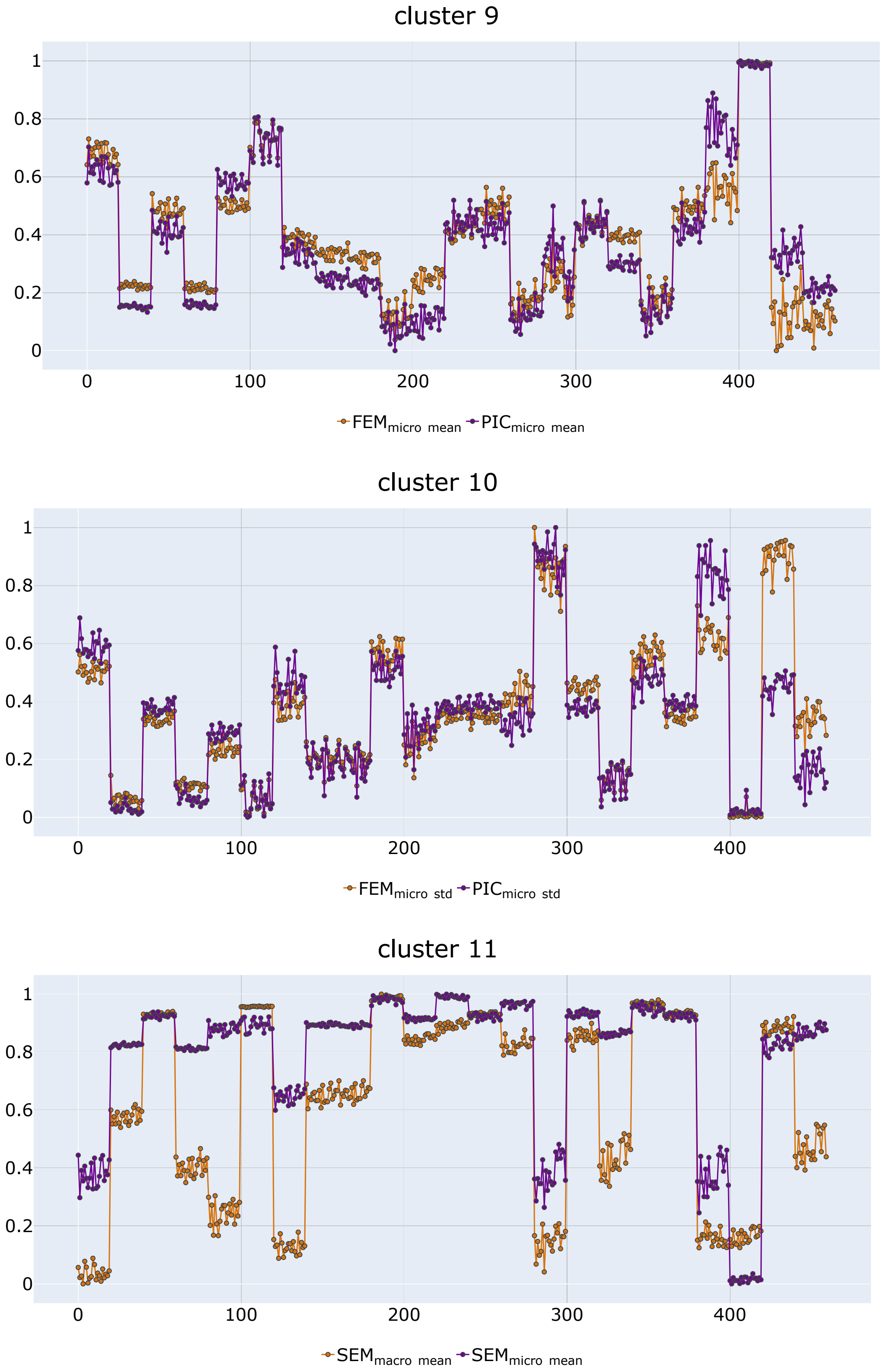


Fig. 3.9.: Clusters 9 - 11: values of clustered metrics across problem instances

Part III

Communication Topologies for distributed Optimization Algorithms

This part addresses the algorithmic aspects of the thesis. First, chapter 4 introduces the fundamentals of distributed optimization algorithms in the form of parallel cooperative metaheuristics. Section 4.1 places particular emphasis on different types of parallelization and on the information sharing cooperation mechanism. This mechanism covers all aspects related to the interaction of distributed solvers, including the communication topology. In section 4.2, two distributed optimization heuristics are presented in detail, which serve as exemplary algorithms in this thesis. Finally, section 4.3 provides an overview of existing research on communication topologies in the context of parallel cooperative metaheuristics and highlights the research gap. Most of the original contributions in this part are presented in chapter 5. It presents the conceptual design of the communication topology variants that allow for different runtime adaptations, and illustrates their impact on algorithmic performance on several problem instances.

4

Distributed Optimization Algorithms

In recent years, a transition to networked systems is emerging in many application areas, such as sensor networks, building automation, smart manufacturing, and especially energy systems. A networked system is composed of a large number of interconnected subsystems that have to cooperate to achieve a global goal [Yan+19]. In particular, future energy systems can be viewed as complex systems of systems, sometimes referred to as multi-energy cyber-physical systems of cyber-physical-energy-systems, linking communication, power, heat, and gas systems [NTS13].

Distributed control and optimization systems represent a way to handle the new scalability requirements arising from the large number of energy sources and the increased complexity caused by distributed information, increasing uncertainties, and real-time requirements. In such distributed control and optimization systems, decentralized components take over control tasks at the local level, while the global system behavior emerges from the interaction of these components [HN20].

The general principle of operation of such distributed control and optimization systems implemented by MAS is as follows: Each asset or subsystem is represented by an intelligent agent. Agents have control over their local asset and knowledge of their environment (control parameters, constraints, local preferences, cost functions, etc.). They exchange information and iteratively adjust decisions regarding their own control parameters to jointly solve a global optimization problem. Direct information exchange in such a peer-to-peer architecture is usually limited, as it would quickly lead to an enormous communication overhead when dealing with a large number of agents. The Communication Topology (CT) determines which agents directly exchange information. It is modeled as a graph $G = (V, E)$, where each agent a_i is represented by a node v_i. An edge $e_{ij} \in E$ indicates that the two agents a_i and a_j communicate directly. Agents usually share the information they receive from their neighbors. In this way, the decisions of all agents can eventually reach all other agents. Thus, different communication topologies lead to different information dissemination in the system. The impact and design of the CT on the optimization process is the core of the thesis. First, however, the principles concerning distributed optimization algorithms that use such topologies must be explained in more detail.

Such distributed systems have several advantages over centralized approaches. First, agents exchange only a limited amount of data, which improves cybersecurity and privacy of data such as measurements, cost functions, and constraints, and reduces communication traffic in comparison to systems where all data is transmitted to a central location. Second, the robustness of the system is increased since there is no single point of failure. Third, parallel computation can lead to an acceleration of problem solving, which of course also depends on the hardware. Finally, the scalability of a distributed system is one of the biggest advantages [Mol+17].

Considerable research on distributed optimization in the power system has been conducted in the field of control theory, where systematic mathematical approaches are used to design distributed controllers [Mol+17]. However, the type of problems that can be solved with such approaches is limited [RAT18]. Distributed energy resources are very heterogeneous regarding forecast precision and flexibility potential. Furthermore, as outlined in section 1.1 and section 1.3, energy optimization problems are often nonlinear, high-dimensional, multimodal, and their decision variables can be highly interdependent. This makes heuristic solution approaches a reasonable choice, and thus heuristic approaches for distributed optimization a central aspect of this thesis. Such approaches correspond to so-called parallel cooperative metaheuristics, which are able to cooperate over a network infrastructure.

Parallel cooperative metaheuristics are the key element of the remainder of the chapter. First, we start with an overview of the design principles of parallel cooperative metaheuristics. In section 4.2, two heuristics that meet the requirements of the motivating use case are presented in detail, as they will be used as exemplary algorithms in the further course of the thesis. The chapter concludes with a survey of related work dealing with the influence and design of communication topologies for parallel cooperative metaheuristics.

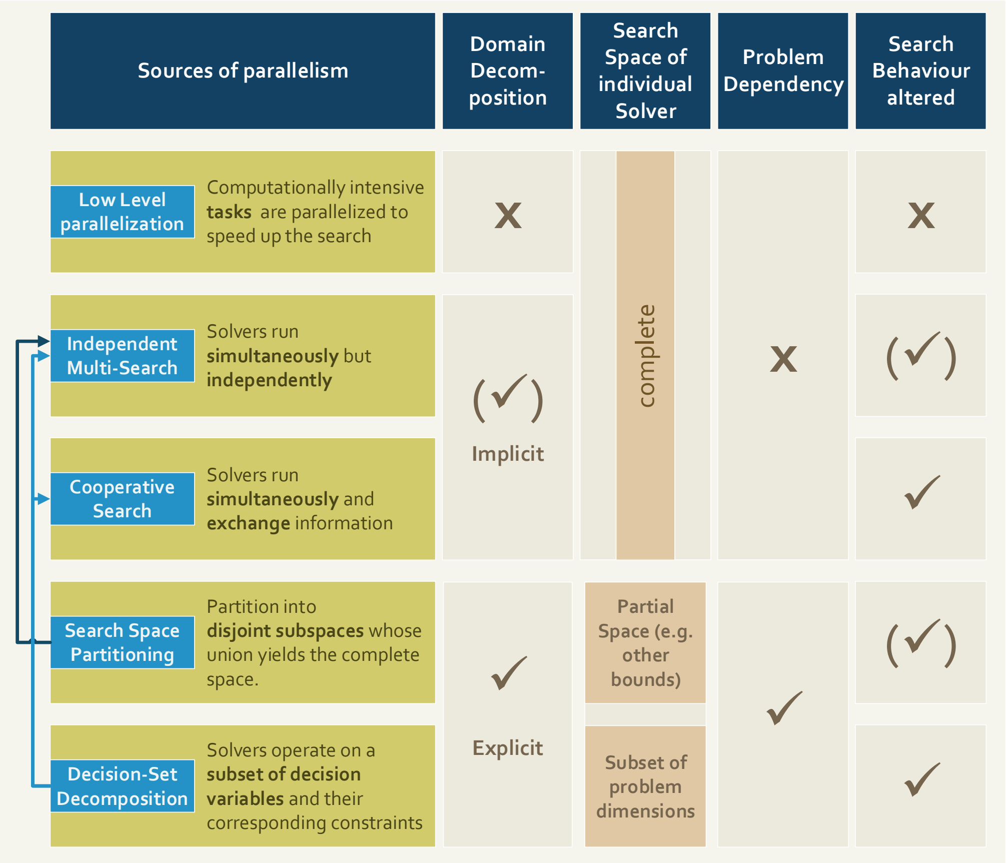
4.1 Parallel cooperative metaheuristics

In parallel metaheuristics, multiple (meta) heuristic solvers run in parallel to solve an optimization problem. If these solvers exchange information during the search, the search is considered cooperative. The parallel design of metaheuristics offers several advantages over the sequential execution. The first goal is usually a **speed up of the search**, as the simultaneously running solvers need less time to explore the search space. At the same time, parallel metaheuristics often yield results of **higher solution quality** compared to their sequential counter parts. This is especially the case in cooperative approaches, where the exchanged information alters the search behavior of the solvers. The **robustness** of the search in terms of the ability to effectively find solutions to different problems and problem instances is increased while the sensitivity to parameters of the metaheuristic is reduced by the parallel design. In addition, parallelization allows many large optimization problems to be solved for the first time [Tal09].

The parallelization of the optimization can be accomplished on different levels. Furthermore, the information exchange in cooperative search is a very important design aspect. Both topics are presented in more detail in the following.

4.1.1 Levels of parallelism

According to Crainic [Cra19] parallelization can be based on a decomposition of the algorithm, the search space or the problem structure. Figure 4.1 summarizes Crainic's classification and adds additional aspects such as problem dependence and impact on search behavior, inspired by the classification of [Tal09].



Sources of parallelism		Domain Decom-position	Search Space of individual Solver	Problem Dependency	Search Behaviour altered
Low Level parallelization	Computationally intensive **tasks** are parallelized to speed up the search	X	complete	X	X
Independent Multi-Search	Solvers run **simultaneously** but **independently**	(✓) Implicit			(✓)
Cooperative Search	Solvers run **simultaneously** and **exchange** information				✓
Search Space Partitioning	Partition into **disjoint subspaces** whose union yields the complete space.	✓ Explicit	Partial Space (e.g. other bounds)	✓	(✓)
Decision-Set Decomposition	Solvers operate on a **subset of decision variables** and their corresponding constraints		Subset of problem dimensions		✓

Fig. 4.1.: Types of parallel metaheuristics

The **low level parallelization** corresponds to decomposition of computationally intensive tasks (during an iteration of the heuristic). Examples include mutation or recombination of large populations in an evolutionary heuristic. This parallelization can therefore be performed to speed up the original heuristic without significant changes. It is still possible to work with only one solver. The procedure is therefore also independent of the optimization problem.

Independent multi-search and **cooperative search** are the other two parallelization approaches that can be performed independently of the problem. In both methods, several solvers run simultaneously to solve the problem. This concurrent exploration of the search space by the (self-contained) solvers implicitly decomposes the problem domain. Nevertheless, the individual solvers are not restricted and can theoretically explore the complete search space. The difference between the two approaches is whether the different solvers cooperate with each other. In **independent multi-search**, the solvers run independently. In the final step, the best overall solution is taken. Thus, in terms of solution quality, the search is equivalent to the sequential execution of solvers. The behavior with respect to a single sequential solver is modified, but not compared to the sequential execution of multiple solvers. In **cooperative search**, on the other hand, the solvers exchange information during the search and can thus benefit from each other. Therefore, the search behavior of a metaheuristic parallelized in this way is quite different and can not only speed up the search, but also often lead to solutions of higher quality [Cra19].

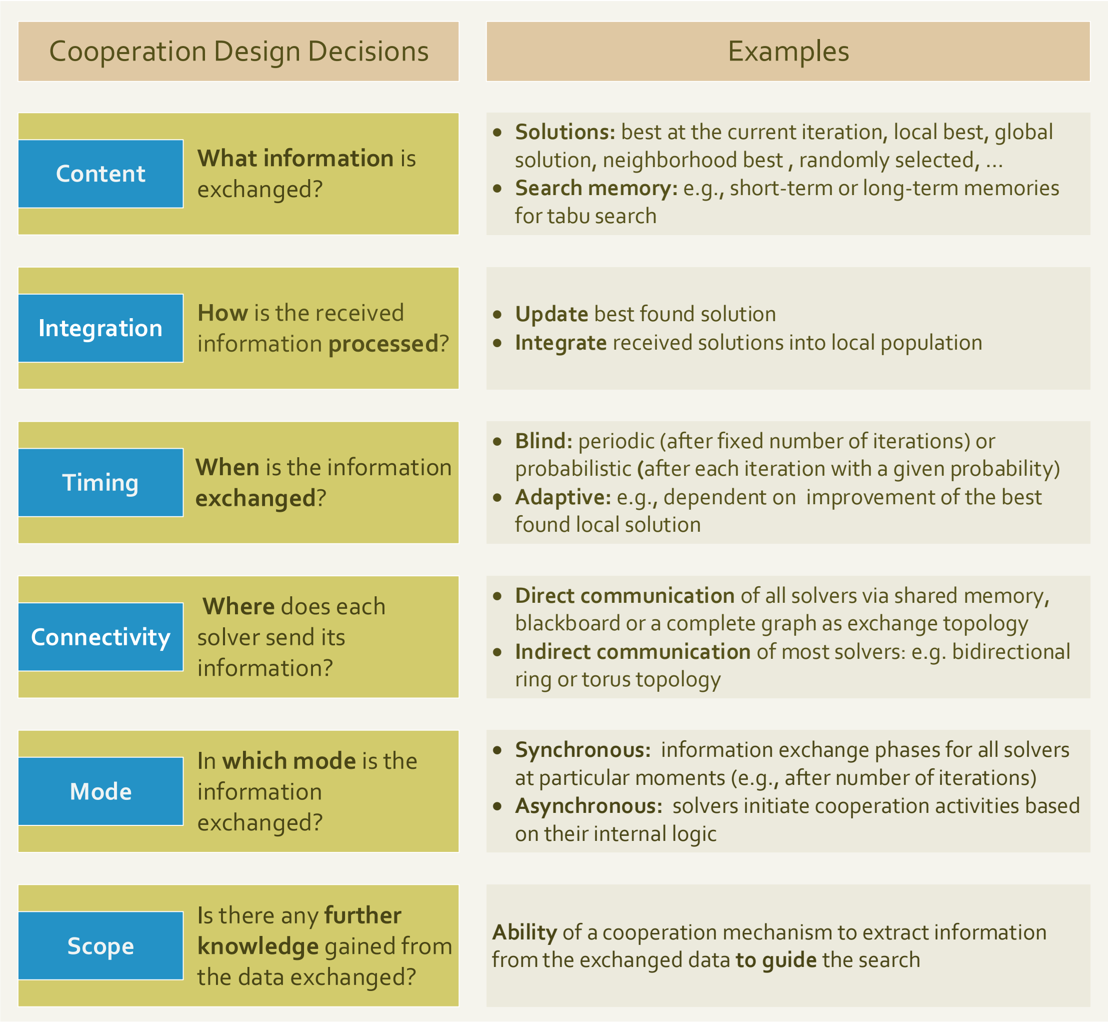
Search space partitioning and **decision-set decomposition** are the two approaches which explicitly decompose the search space. How this decomposition is performed is problem dependant. The simplest scenario for partitioning the search space is to divide the search space into smaller subspaces and solve the resulting subproblems by applying sequential metaheuristics to each subspace, thus performing an independent multi-search. With **decision-set decomposition**, each solver receives a subset of decision variables and the corresponding constraints. When optimizing this subset, other variables are discarded or (temporarily) fixed. The fixed variables of other solvers can be updated during the iterative optimization process. Often, separation is determined by a single control process that re-assembles and evaluates the entire decision set. If **decision-set decomposition** is performed without a central coordination process, a **cooperative search** allows solvers to exchange information about the current candidate solutions for their respective decision variables. Thus, each solver can regularly update the fixed foreign variables and the solvers jointly converge to an overall solution.

The presented parallelization approaches can be combined in various ways. However, the combination of **decision-set decomposition** and **cooperative search** yields the kind of distributed optimization heuristic that is the subject of further investigation in this thesis. The decomposition of decision sets is inherent to spatially distributed optimization problems from the energy domain.

Optimizing these problems by cooperative search allows to simultaneously exploit the aforementioned advantages of distributed computation (see introduction of this chapter) and also to leverage the positive effects of solver interaction. The design of the cooperation mechanism for cooperative search has a strong influence on its performance and is especially important in spatially distributed systems. Therefore, the information-sharing cooperation mechanism is considered in more detail next.

4.1.2 Information-sharing cooperation mechanism

The Information-Sharing Cooperation Mechanism (ISCM) covers all aspects related to information exchange and cooperation between solvers. The global search behaviour of the cooperative search emerges from the interactions that are specified by this mechanism. The ISCM itself can thus be regarded as a new metaheuristic [Cra19].



Cooperation Design Decisions		Examples
Content	**What information** is exchanged?	• **Solutions:** best at the current iteration, local best, global solution, neighborhood best , randomly selected, … • **Search memory:** e.g., short-term or long-term memories for tabu search
Integration	**How** is the received information **processed**?	• **Update** best found solution • **Integrate** received solutions into local population
Timing	**When** is the information **exchanged**?	• **Blind:** periodic (after fixed number of iterations) or probabilistic (after each iteration with a given probability) • **Adaptive:** e.g., dependent on improvement of the best found local solution
Connectivity	**Where** does each solver send its information?	• **Direct communication** of all solvers via shared memory, blackboard or a complete graph as exchange topology • **Indirect communication** of most solvers: e.g. bidirectional ring or torus topology
Mode	In **which mode** is the information exchanged?	• **Synchronous:** information exchange phases for all solvers at particular moments (e.g., after number of iterations) • **Asynchronous:** solvers initiate cooperation activities based on their internal logic
Scope	Is there any **further knowledge** gained from the data exchanged?	**Ability** of a cooperation mechanism to extract information from the exchanged data **to guide** the search

Fig. 4.2.: Design decisions regarding the cooperation mechanism [Tal09; CT07]

Figure 4.2 shows an overview of the design decisions to be considered when developing a cooperation mechanism according to Talbi [Tal09] and Crainic and Toulouse [CT07]. The first question that arises is what information should be exchanged between the solvers. This usually includes elite solutions, such as the overall best solution or the best solution of the current iteration. Furthermore, additional information from the solver's search memory can be provided and used to adapt the search parameters of other solvers. One example are the pheromone trails (or the probability model) in ant colonies.

The next design decision concerns how the received information is processed by the solvers. In population-based metaheuristics, for example, the worst solution candidates in the solver's own population can be replaced by the newly obtained solution candidates. One way to use contextual information from the search memory is to combine it with the local version. For example, in ant colonies, the local and neighboring pheromone matrices can be aggregated linearly.

Another important aspect is timing. According to Talbi, the exchange of information can either follow a *blind* scheme, i.e. periodically or probabilistically, or be intelligently adapted to the runtime behavior. For instance, the sending of messages can be tied to achieving a large enough improvement in the elite solution.

The design decision that is of central interest for this thesis is the connectivity. Crainic and Toulouse differentiate between direct and indirect communication. In the fist case, all solvers can directly share information. Such a direct exchange can be realized via shared memory or blackboard architectures, for example. For spatially distributed systems, it can be implemented by a fully meshed Communication Topology (CT), respectively a complete graph. In indirect communication, the number of solvers that communicate directly with each other is limited. The CT determines which solvers are adjacent and explicitly exchange information. When a solver receives information from a neighbor, it influences its own search and thus the messages it sends to its other neighbors. Therefore, information is shared between cooperating solvers not only explicitly through the direct exchange, but also implicitly through a propagation (or diffusion) process. Crainic and Toulouse identify the dynamic behavior emerging from solver cooperation as an important area of research. They point out that the explicit design of implicit interactions could be leveraged to increase algorithms' performance.

The mode of the information exchange is another important design decision. Synchronous communication requires information exchange phases, in which all solvers engage and stop other activities. The synchronization may be at predefined points, e.g., after a number of iterations, or may be initiated by a specific solver. The aim is a recreation of a state of full knowledge. However, synchronous communication is usually time inefficient since the slowest search process typically sets the pace. In general, asynchronous information sharing outperforms synchronous methods. In asynchronous communication mechanisms solvers initiate cooperation activities based on their internal logic. This can be realized, for instance, by sending a newly found superior solution candidate, or by requesting new information from other solvers when the own optimization process stagnates. This eliminates the time overhead of waiting for slower solvers. Furthermore, the asynchrony of the solvers also allows for increased adaptability of the solvers and thus dynamic adaptation of the exploration of the search space. However, the asynchronous nature also poses challenges, as poorly chosen parameters (e.g., the exchange interval or criterion) can cause undesirable behavior, such as erratic searches in which the corresponding search trajectories resemble random walks [CT07].

The last design decision according to Crainic and Toulouse [CT07] is the so called scope. This describes the ability of a cooperation mechanism to extract new information and knowledge from the exchanged data in order to guide the search. Most examples of this originate from the context of asynchronous cooperative search with centralized memory. For instance, the combination of complementary metaheuristic and exact methods allows intelligent coordination between different solvers specialized in exploration or exploitation of the search space. The data stored by the solvers in the central memory may also be used to learn about the parts of the search space that have already been explored, or the relationships between the values of certain decision variables. Cooperative search strategies that involve decision-set decomposition must include a mechanism for reconstructing complete solutions. This is also considered a form of knowledge creation. Crainic and Toulouse explicitly name the *Integrative Cooperative Search* by Lahrichi et al. [Lah+15] as example for this. Lahrichi et al. combine a number of independent exact or metaheuristic solvers. Some of them work on the subproblems, while the so-called integrators combine the resulting partial solutions into overall solutions, aiming for high solution quality. The solvers collaborate through an adaptive search-guidance mechanism based on the paradigm of cooperative search with centralized memory. Another example are coevolutionary algorithms. The recombination of partial solutions can be performed in various ways, e.g. by different information splicing methods [SS04] (see section 4.2.2 for more detail), and thus influence the further search process significantly.

In the examples mentioned, knowledge creation takes place in a central instance that recombines the partial solutions of the solvers. However, such a recombination and thus knowledge creation must also take place in completely distributed algorithms. Two variants for such a distributed mechanism are presented in the following chapter.

4.2 Spatially distributed optimization of decomposed domains

The motivating use case of optimized operation of spatially distributed energy sources and consumers poses certain challenges. Each asset has its own local requirements, constraints and preferences, that lead to different kinds of flexibility. Nevertheless, the assets need to jointly reach a global target, e.g. to achieve a load reduction in a certain area of the grid to prevent a transformer overload. Due to the scalability and data protection issues mentioned earlier, it is not advantageous to gather all information about energy resource flexibilities in a central location. However, the plants still need to be jointly coordinated. A solution to this is an optimization algorithm that allows distributed agents to optimize their own assets locally, but share information with the other agents to find an overall solution. In addition, a decision set decomposition is inherent to these optimization problems, since the contribution of each asset can be considered as a subset of the decision variables of the overall problem. Thus, the importance of the communication aspects is even more pronounced in this use case compared to a centrally performed cooperative search. On the one hand, there is an actual spatial distribution and, on the other hand, the agents have to be informed regularly about the selection of the other agents, as it may massively impact the evaluation of their own solution options (see section 2.3).

In the work at hand, two optimization heuristics which suit this use case are employed to explore the impact of different communication topologies and how these topologies can be designed in an optimized manner. The first, *COHDA* is a fully distributed optimization heuristic developed for energy resource scheduling [HS17]. The second, named *IDICE*, was developed as part of this work by combining well known principles of the generalized island model, coevolutionary algorithms and differential evolution.

4.2.1 Combinatorial optimization heuristic for distributed agents (COHDA)

COHDA is a combinatorial optimization heuristic for distributed agents, and was developed for the self-organized scheduling of distributed energy resources in virtual power plants [HS17; Hin14]. Since it was developed for the motivating use case, it has exactly the required properties: it is a cooperative search with domain decomposition. In the original setup, each agent has to choose an operation schedule for its power plant, while the combination of all chosen schedules is supposed to reach an overall target schedule. In [BL17], Bremer et al. adapted COHDA to find the global minimum of a real valued objective function. In the work at hand this approach is adopted, but each agent is responsible for two decision variables of the overall problem. This approach is used to combine fitness landscape analysis for distributed search spaces (see chapter 3) in a convenient way with the investigation of algorithm performance in relation to communication topologies. By adapting to this setup, the heuristic is slightly simplified compared to the version in [HS17]. In the following, the heuristic as implemented in this work is presented in more detail.

Each agent knows the global objective function $f(\vec{x}) = f(x_1, x_2, \ldots, x_n)$, which depends partly on its own decision variables, but largely on the decisions of the other agents. In its so-called working memory $\kappa_i = (\Omega_i, \Upsilon_i)$, the agent stores the believed current configuration of the whole system Ω_i and the best known solution candidate Υ_i.[1] The current system state $\Omega_i = [(x_{1\Omega_i}, \lambda_{1i}), (x_{2\Omega_i}, \lambda_{2i}), \ldots, (x_{n\Omega_i}, \lambda_{ni})]$ contains the latest known choice for each decision variable in $\vec{x}$, together with the version counter λ_j of the respective agent a_j which is responsible for the variable. The solution candidate $\Upsilon_i = ([x_{1\Upsilon_i}, x_{2\Upsilon_i}, \ldots, x_{n\Upsilon_i}], a_{\Upsilon_i})$ also contains a choice for each decision variable in $\vec{x}$ and the id of the agent which found the candidate. The working memory κ_i is used for the local operation of the agent, but also sent to its neighboring agents.

The basic workflow of the heuristic consists of each agent performing the following three steps in each iteration:

1. **perceive:** the agent processes all newly received messages. It updates Ω_i when a newer selection of variables becomes known, and it updates Υ_i when it receives a better solution candidate (see section on **information integration** below for more detail).
2. **decide:** the agent performs a local optimization to check if it can find a better solution candidate with given choices of other agents from Ω_i. For this, the Simplicial Homology Global Optimization (SHGO) [ESF18] is employed.[2]. If the newly found solution outperforms Υ_i, the candidate is updated. Otherwise, the agent falls back on its selection in Υ_i for its own decision variables and uses them to update the system state Ω_i.
3. **act:** if any component in the working memory $\kappa_i = (\Omega_i, \Upsilon_i)$ has changed in the previous two steps, the agent sends its updated κ_i to its neighbors.

Abstracting COHDA from its original use case, the sequence of the three steps mentioned above remains, as well as the specification of the inter-agent cooperation.

---

[1] Note that the original working memory also contains the target schedule. This was omitted since the benchmark function $f(\vec{x})$ does not change and is known to the agents from the beginning.

[2] Implementation by python library SciPy [Vir+20]

The local optimization in the "decide" step can be designed arbitrarily depending on the use case and, for example, consider local constraints or additional local objective functions. Therefore, according to Crainic's definition [Cra19], COHDA can be viewed as an *ISCM* that allows arbitrary (meta-)heuristic solvers to cooperatively solve a problem with domain decomposition. The design decisions that determine the cooperation mechanism, as presented in section 4.1.2 with an overview in fig. 4.2, have several aspects that overlap with the phases outlined above. However, for the sake of completeness, they are all described below:

Content: Agents send their working memory $\kappa_i = (\Omega_i, \Upsilon_i)$, including their believed current system state and the best known solution candidate.

Information Integration: To integrate the newly received working memory $\kappa_j = (\Omega_j, \Upsilon_j)$ from agent j, agent i checks if Ω_i or Υ_i have to be updated.

- the **believed current system state** Ω_i is updated according to the following rules:
  - if Ω_j includes decision variables that are not included in Ω_i, they are added to Ω_i[3]
  - if Ω_j contains newer values for a decision variable, i.e. the version counter λ_{kj} for the decision variable x_k is greater than λ_{ki}, then the corresponding decision variable is replaced
- the **solution candidate** Υ_i is updated according to the following rules:
  - if Υ_j contains more decision variables than Υ_i, Υ_i is replaced by Υ_j[3]
  - if Υ_j contains other decision variables than Υ_i, these decision variables are added to Υ_i[3]
  - if Υ_j and Υ_i contain the same number of decision variables, but the objective value of Υ_j is better, then Υ_i is replaced
  - if Υ_j and Υ_i contain the same number of decision variables and the objective values are equal, then the solution that was found by the agent with the lowest id is taken. This requires unique identifiers for the agents that allow a lexicographic ordering. The step is necessary to break ties and allow convergence to a common solution.

Timing/ Exchange Criterion: In the "act" phase of each iteration, the working memory κ is sent to the neighbors if anything has changed in the current iteration, whether by external information or during local optimization.

---

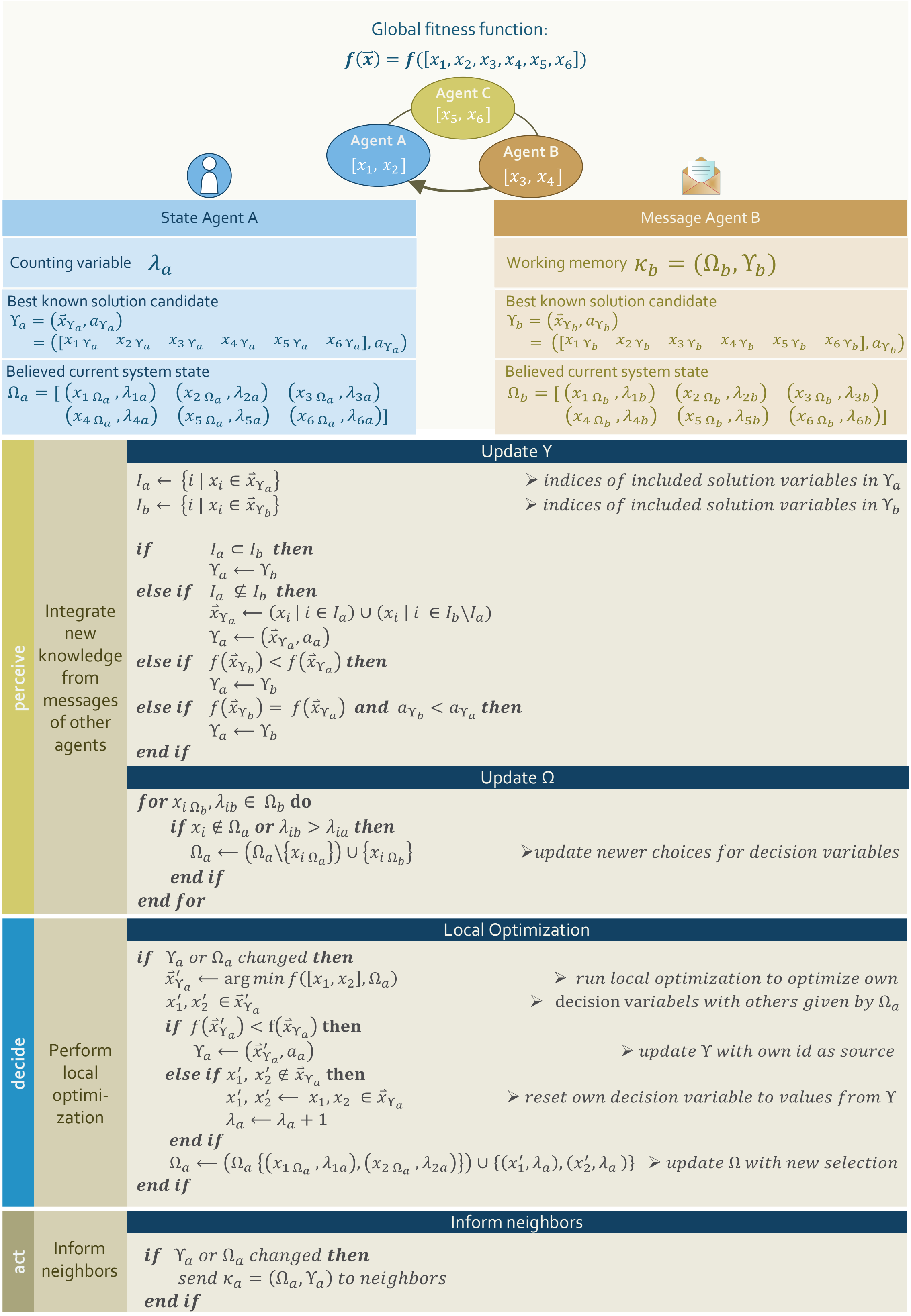
[3] This is the case in the initial phase, when the agent doesn't yet know the initial decisions made by all other agents.

Connectivity: A bidirectional graph defines the Communication Topology (CT). Each agents sends information to all of its neighbors. The CT must be interconnected, irreflexive and symmetrical [Hin14].

Mode: Messages are passed asynchronously between agents. The spatial distribution of the agents results in different transmission times (e.g. with TCP/IP communication on the public internet). In [HLS13b], Hinrichs et al. have shown that these varying delays do not negatively affect the achieved solution quality, and that COHDA even benefits from a slight agent-level variation caused by message delays. This aspect was further explored and exploited by Bremer et al.[BL19] by introducing laziness to agents. The agents probabilistically delayed their reaction to received messages, which often led to an increase in diversity in the solution population of the overall heuristic, preventing premature convergence and thus achieving better solution quality. To leverage the positive effects of inter-agent variation, the simulation studies in the work at hand included a short sleeping period with a random value between 0.05 and 0.15 seconds in the beginning of each iteration of an agent.

Scope: Unlike most cooperative search algorithms based on decision set decomposition, COHDA does not recombine the different solution fragments at a central point. Instead, a variety of complete solutions are present in the system at any given time in the form of perceived system states Ω. These system states are updated asynchronously, causing their diversity to fluctuate during runtime. They provide the basis for the local optimization runs, which include the full execution of the local solver, not just a few iterations as in most cooperative search algorithms.

For a better understanding, fig. 4.3 shows a small example. The setup involves only three agents, each of which has two decision variables. Agent A receives a message from Agent B and performs one iteration, which includes the three phases of updating its knowledge, performing local optimization, and informing its neighbors.


Global fitness function:
$f(\vec{x}) = f([x_1, x_2, x_3, x_4, x_5, x_6])$
Agent C
$[x_5, x_6]$
Agent A
$[x_1, x_2]$
Agent B
$[x_3, x_4]$
perceive
Integrate new knowledge from messages of other agents
decide
Perform local optimi-zation
act
Inform neighbors

State Agent A

Counting variable λ_a

Best known solution candidate
$\Upsilon_a = (\vec{x}_{\Upsilon_a}, a_{\Upsilon_a}) = ([x_{1\,\Upsilon_a}\ \ x_{2\,\Upsilon_a}\ \ x_{3\,\Upsilon_a}\ \ x_{4\,\Upsilon_a}\ \ x_{5\,\Upsilon_a}\ \ x_{6\,\Upsilon_a}], a_{\Upsilon_a})$

Believed current system state
$\Omega_a = [\,(x_{1\,\Omega_a}, \lambda_{1a})\ \ (x_{2\,\Omega_a}, \lambda_{2a})\ \ (x_{3\,\Omega_a}, \lambda_{3a})\ \ (x_{4\,\Omega_a}, \lambda_{4a})\ \ (x_{5\,\Omega_a}, \lambda_{5a})\ \ (x_{6\,\Omega_a}, \lambda_{6a})]$

Message Agent B

Working memory $\kappa_b = (\Omega_b, \Upsilon_b)$

Best known solution candidate
$\Upsilon_b = (\vec{x}_{\Upsilon_b}, a_{\Upsilon_b}) = ([x_{1\,\Upsilon_b}\ \ x_{2\,\Upsilon_b}\ \ x_{3\,\Upsilon_b}\ \ x_{4\,\Upsilon_b}\ \ x_{5\,\Upsilon_b}\ \ x_{6\,\Upsilon_b}], a_{\Upsilon_b})$

Believed current system state
$\Omega_b = [\,(x_{1\,\Omega_b}, \lambda_{1b})\ \ (x_{2\,\Omega_b}, \lambda_{2b})\ \ (x_{3\,\Omega_b}, \lambda_{3b})\ \ (x_{4\,\Omega_b}, \lambda_{4b})\ \ (x_{5\,\Omega_b}, \lambda_{5b})\ \ (x_{6\,\Omega_b}, \lambda_{6b})]$

Update Υ

```
$I_a \leftarrow \{i \mid x_i \in \vec{x}_{\Upsilon_a}\}$        ➢ indices of included solution variables in $\Upsilon_a$
$I_b \leftarrow \{i \mid x_i \in \vec{x}_{\Upsilon_b}\}$        ➢ indices of included solution variables in $\Upsilon_b$

if $I_a \subset I_b$ then
    $\Upsilon_a \longleftarrow \Upsilon_b$
else if $I_a \not\subseteq I_b$ then
    $\vec{x}_{\Upsilon_a} \longleftarrow (x_i \mid i \in I_a) \cup (x_i \mid i \in I_b \backslash I_a)$
    $\Upsilon_a \longleftarrow (\vec{x}_{\Upsilon_a}, a_a)$
else if $f(\vec{x}_{\Upsilon_b}) < f(\vec{x}_{\Upsilon_a})$ then
    $\Upsilon_a \longleftarrow \Upsilon_b$
else if $f(\vec{x}_{\Upsilon_b}) = f(\vec{x}_{\Upsilon_a})$ and $a_{\Upsilon_b} < a_{\Upsilon_a}$ then
    $\Upsilon_a \longleftarrow \Upsilon_b$
end if
```

Update Ω

```
for $x_{i\,\Omega_b}, \lambda_{ib} \in \Omega_b$ do
    if $x_i \notin \Omega_a$ or $\lambda_{ib} > \lambda_{ia}$ then
        $\Omega_a \longleftarrow (\Omega_a \backslash \{x_{i\,\Omega_a}\}) \cup \{x_{i\,\Omega_b}\}$        ➢update newer choices for decision variables
    end if
end for
```

Local Optimization

```
if $\Upsilon_a$ or $\Omega_a$ changed then
    $\vec{x}'_{\Upsilon_a} \longleftarrow \arg min\, f([x_1, x_2], \Omega_a)$        ➢ run local optimization to optimize own
    $x'_1, x'_2 \in \vec{x}'_{\Upsilon_a}$        ➢ decision variabels with others given by $\Omega_a$
    if $f(\vec{x}'_{\Upsilon_a}) < \mathrm{f}(\vec{x}_{\Upsilon_a})$ then
        $\Upsilon_a \longleftarrow (\vec{x}'_{\Upsilon_a}, a_a)$        ➢ update $\Upsilon$ with own id as source
    else if $x'_1, x'_2 \notin \vec{x}_{\Upsilon_a}$ then
        $x'_1, x'_2 \longleftarrow x_1, x_2 \in \vec{x}_{\Upsilon_a}$        ➢ reset own decision variable to values from $\Upsilon$
        $\lambda_a \longleftarrow \lambda_a + 1$
    end if
    $\Omega_a \longleftarrow (\Omega_a \{(x_{1\,\Omega_a}, \lambda_{1a}), (x_{2\,\Omega_a}, \lambda_{2a})\}) \cup \{(x'_1, \lambda_a), (x'_2, \lambda_a)\}$        ➢ update $\Omega$ with new selection
end if
```

Inform neighbors

```
if $\Upsilon_a$ or $\Omega_a$ changed then
    send $\kappa_a = (\Omega_a, \Upsilon_a)$ to neighbors
end if
```

Fig. 4.3.: Exemplary behavior in COHDA of agent A in one iteration after receiving a message from agent B

4.2.2 Island model differential coevolution (IDICE)

IDICE is a combination of the generalized island model [IRB12] with the concept of coevolution [SS04]. Both models are approaches for distributed EAs. In [Gon+15], Gong et al. distinguish between population-distributed and dimension-distributed models. Population distributed EAs, like the island-model, distribute the individuals of the population (or subpopulations) across multiple processors or computational nodes. Thus, they are by their design suitable for a spatially distributed computation. Dimension-distributed models, like coevolution, distribute partitions of the problem dimensions (or subspaces), i.e., they perform a decision set decomposition. However, in the majority of works in literature, the individual solvers are coordinated by a central mechanism (master slave setup) that also repeatedly assembles complete solutions and passes them to all solvers as a new global iteration step (e.g. [SS04]). Instead of this central coordination mechanism, the island model is used here. Both models combined meet the requirements of the motivating use case, since the island model allows to handle the distributed nature of the system and the inherent decision set decomposition can be addressed by coevolution.

The generalized island model [IRB12] by Izzo et al. is a generic framework for the combination of different metaheuristics by the island model. Each metaheuristic is executed by a solver on an "island". The islands exchange information via a migration topology to mutually enrich their searches. Izzo et al. define the archipelago $\mathbb{A}$, that is the overall setup for optimization, by the set of islands $\mathbb{I}$ and their migration topology $\mathcal{T}$, hence $\mathbb{A} = (\mathbb{I}, \mathcal{T})$. The set of islands of course corresponds to the set of agents and the migration topology to the CT for the presented use case.

Every island I_i is defined by a quadruple:

$$I_i = (\mathcal{A}_i, G_i, \mathcal{M}_i, \mathcal{R}_i) \tag{4.1}$$

where $\mathcal{A}_i$ is the optimization algorithm and G_i the population, respectively the current generation (which may only contain one individual for trajectory-based heuristics). $\mathcal{M}_i$ is the migration selection policy, which is used to select the migration pool P containing individuals from G_i that will be sent to neighboring islands. $\mathcal{R}_i$ is the migration replacement policy, which determines how individuals from a received migration pool P are integrated into the existing population. $\mathcal{M}_i$ and $\mathcal{R}_i$ correspond to the exchanged content and the integration of information regarding the design decisions of the cooperation mechanism shown in fig. 4.2.

Combining this setup with the concept of coevolution, metaphorically speaking, a different species lives on each island. The species evolve on their islands. For the evaluation of the fitness of a species, however, all other species are relevant.

Migration between islands regularly brings new individuals from other species to the islands. They are not integrated into the local population, but they change how the current population is evaluated in terms of fitness and thus how it must evolve to improve its fitness. Thus, similar to COHDA, in IDICE each agent requires a perceived current system state Ω. This contains the currently assumed values for the properties of the other species, respectively for the decision variables of the other agents. The agents again share the global objective function $f(\vec{x}) = f(x_1, x_2, \ldots, x_n)$. To evaluate the fitness of its population, which includes only its own decision variables, an agent uses the values from Ω for the other decision variables.

The islands also store the best found solution candidate $\Upsilon_i = ([x_{1\Upsilon_i}, x_{2\Upsilon_i}, \ldots, x_{n\Upsilon_i}], a_{\Upsilon_i})$ which contains a value for each decision variable in $\vec{x}$ and the id of the agent which found the candidate. To ensure that all islands eventually converge to a common solution despite the distributed computation without a central controller, Υ is sent to the neighboring islands together with the migration pool P. In analogy to COHDA, agents send their working memory $\kappa = (P, \Upsilon)$, In addition, migration does not change the population on the islands. Therefore, no migration-replacement policy $\mathcal{R}_i$ is needed, but rather an update policy $\mathcal{U}_i$ for Ω_i and Υ_i. Therefore, in IDICE the islands are defined by a six-tuple:

$$I_i = (\mathcal{A}_i, G_i, \Omega_i, \Upsilon_i, \mathcal{M}_i, \mathcal{U}_i) \tag{4.2}$$

Izzo et al. specify the optimization algorithm $\mathcal{A}$ as any optimization process that supports an evolution operator $G' = \mathcal{A}(G, \mu)$, where μ refers to the migration interval, i.e., the number of iterations performed between migrations. For IDICE, Differential Evolution (DE)[SP97] is employed as optimization algorithm by each agent. The implementations of mutation, crossover and selection are taken from [Vir+20], but adapted to the special setup in the island model. Moreover, G_i includes only the own decision variables of I_i. Thus, Ω_i is always necessary for the evaluation and further development of G_i by an algorithm. The evolution operator is hence specified as $G' = \mathcal{DE}(G, \Omega, \mu)$.

In analogy to COHDA, each island agent also must iteratively perform the three basic steps:

- "perceive": process messages from neighbors
- "decide": continue local optimization process
- "act": send messages to neighbors

The order of the steps in IDICE is therefore different from that in the generalized island model of Izzo et al, namely: decide - act - perceive. Because of the fully distributed optimization new solution candidates (Υ) should always be forwarded. Therefore, in this setting, it is most reasonable to perform the "act" step last. Furthermore, a local optimization is most useful after Ω has been updated. Due to these considerations, the same order is used for IDICE as for COHDA, i.e. perceive- decide - act. Algorithm 2 shows the general flow of the optimization algorithm for an island agent as pseudocode.

Algorithm 2 Process of optimization on the island I_i respectively for the agent a_i

```
initialize $G_i$
$P^* \leftarrow \mathcal{M}_i(G_i)$                    ▷ select initial choices as migration pool
/* Send initial choices to islands adjacent to $I_i$ in $\mathcal{T}$ */

while !stop_criteria do
    /* Let $\mathbb{K}$ be a set of working memories $\kappa = (P, \Upsilon)$ from neighboring islands */
    $\Omega_i, \Upsilon_i \leftarrow \mathcal{U}_i(G, \mathbb{K})$                    ▷ Update $\Omega$ and $\Upsilon$
    $G_i \leftarrow \mathcal{DE}(G_i, \Omega_i, \mu)$    ▷ Perform $\mu$ iterations of $\mathcal{DE}$ to evolve $G_i$ with given $\Omega$
    $P^* \leftarrow \mathcal{M}_i(G_i)$                    ▷ select migration pool
    /* Send $\kappa^* = (P^*, \Upsilon_i)$ to islands adjacent to $I_i$ in $\mathcal{T}$ */
end while
```

After outlining the general workflow of the heuristic, the design aspects of the ISCM can be examined in more detail in order to gain a deeper insight into the algorithm. The final parameterization of IDICE with all mentioned parameters can be found in the appendix in appendix A.2.2.

Content: Agents send their working memory $\kappa_i = (P_i, \Upsilon_i)$, including the migration pool and the best known solution candidate. The migration pool P_i contains the k-best individuals from the current generation of the population G_i. The current values for foreign decision variables are completed from Ω_i, which was also used to evaluate the individuals from G_i. (See message of agent B in fig. 4.4 as example)

Information Integration: In the "perceive" phase an island agent processes all messages received since its last iteration. The messages contain a set of working memories $\mathbb{K}$ from neighboring islands. Based on this, the the best found solution candidate Υ_i and the perceived system state Ω_i are updated.

The island agent examines all received solution candidates Υ and updates its own Υ_i (similar to COHDA) by the foreign solution candidate Υ_j if necessary using the following rules:

- if the objective value of Υ_j is better, then Υ_i is replaced
- if the objective values of Υ_j and Υ_i are equal, then the solution that was found by the agent with the lowest id is taken. This requires unique identifiers for the agents that allow a lexicographic ordering. The step is necessary to break ties and allow convergence to a common solution.

In order to update Ω_i, the island agent first collects all received migration pools from other agents and adds its own k-best individuals from the current generation to form the combined pool P'. Based on P', the splicing pool P'' is generated. The goal of the information splicing operation is to combine information from agents stochastically. For IDICE, the unified splicing operation proposed by Subbu and Sanderson in [SS04] is used. Here, the splicing pool P'' is generated by randomly taking one value for each decision variable from the values contained in P' to construct ρ new solutions. This procedure is valid for the selected benchmark functions. However, for real-world problems, dependencies between decision variables would have to be taken into account. Finally, P' and P'' are combined into the full solution pool P^*. The best individual in P^* is selected as new Ω_i. This coordination scheme corresponds to the "pooling" strategy of Subbu and Sanderson, which achieved the best results in their work compared to other schemes.

Note that the agent only receives messages from its direct neighbors. Therefore, in the obtained migration pools, the decision variables of the direct neighbors vary most. However, due to different Ω of the neighbors, the pool P' also contains many different values for decision variables of more distant agents. This indirect propagation of possible values for all decision variables is largely responsible for the system's information dissemination process, especially in sparse communication topologies.

Another noteworthy aspect concerns population evaluation. During the optimization process, the objective function remains the same, but Ω changes frequently after messages from other agents arrive. Thus, the evaluation of the existing population changes after an update of Ω, which leads to the fact that the population can deteriorate repeatedly. However, this does not apply to the solution candidate Υ. It contains a complete solution and only ever improves over the entire course of the optimization.

Timing/ Exchange Criterion: In the "act" phase of each iteration, the working memory κ is sent to the neighbors if at least one of two conditions holds: either the agent has so far performed fewer iterations of its DE solver than minimally required ($iter \leq iter_{min}$), or a new best solution candidate Υ_i has been found. The first condition aims to ensure that at the beginning of the heuristic, agents communicate their migration pools after each iteration to drive the propagation of different valid values for all decision variables. The second condition is necessary to enable convergence on a common solution. In order to accelerate convergence, the threshold for the required improvement of the self-discovered Υ increases with time (determined by the parameter cf). Solution candidates, which originate from other agents, must nevertheless be forwarded even in the case of smaller improvements in order to allow convergence to a common solution. Following Talbi's classification of the exchange criterion [Tal09], IDICE transitions from a "blind" scheme to an adaptive one. In the initial phase, each agent communicates blindly after each μ iteration of the DE solver. While later, communication is only triggered depending on solution candidate improvements.

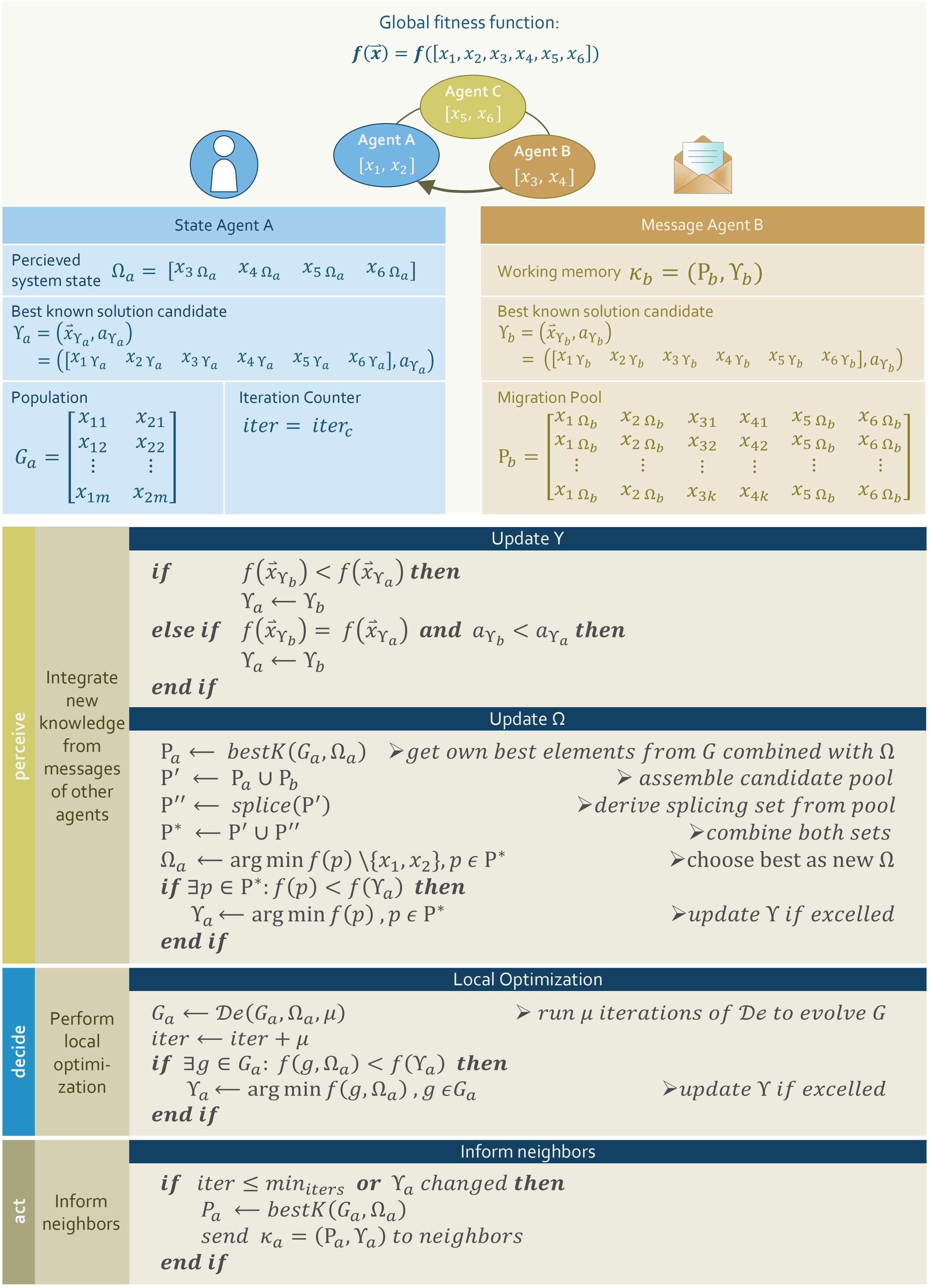
Connectivity: A bidirectional graph defines the communication topology. Each agents sends information to all of its neighbors. The CT must be interconnected, irreflexive and symmetrical.

Mode: Messages are passed asynchronously between agents. The spatial distribution of the agents results in different transmission times (e.g. with TCP/IP communication on the public internet). Like COHDA, IDICE also involves a short sleeping period with a random value between 0.05 and 0.15 seconds in the beginning of each iteration of an agent. This is intended to leverage potentially positive effects of the inter-agent variation.

Scope: The information splicing operation described for the information integration aspect is the primary mechanism for recombining information from the overall system. Depending on the CT that defines the information flow in the system, very different combinations of partial solutions are created in this step. This allows the island agents to reach completely different areas in the high-dimensional search space and to further explore the most promising one after updating Ω.

For a better understanding, the same example as for COHDA is shown in fig. 4.4. The setup involves only three agents, each of which has two decision variables. Agent A receives a message from Agent B and performs one iteration, which includes the three phases of updating its knowledge, performing local optimization, and informing its neighbors.

After presenting the two heuristics used in this thesis, the communication aspects need to be considered in more detail. Therefore, the following section presents research that investigates the impact of communication topologies on distributed optimization heuristics.


Global fitness function:
$\boldsymbol{f}(\vec{\boldsymbol{x}}) = \boldsymbol{f}([x_1, x_2, x_3, x_4, x_5, x_6])$
Agent C $[x_5, x_6]$
Agent A $[x_1, x_2]$
Agent B $[x_3, x_4]$
State Agent A
Percieved system state $\Omega_a = [x_{3\,\Omega_a} \quad x_{4\,\Omega_a} \quad x_{5\,\Omega_a} \quad x_{6\,\Omega_a}]$
Best known solution candidate
$\Upsilon_a = (\vec{x}_{\Upsilon_a}, a_{\Upsilon_a})$
$= ([x_{1\,\Upsilon_a} \quad x_{2\,\Upsilon_a} \quad x_{3\,\Upsilon_a} \quad x_{4\,\Upsilon_a} \quad x_{5\,\Upsilon_a} \quad x_{6\,\Upsilon_a}], a_{\Upsilon_a})$
Population
$G_a = \begin{bmatrix} x_{11} & x_{21} \\ x_{12} & x_{22} \\ \vdots & \vdots \\ x_{1m} & x_{2m} \end{bmatrix}$
Iteration Counter
$iter = iter_c$
Message Agent B
Working memory $\kappa_b = (\mathrm{P}_b, \Upsilon_b)$
Best known solution candidate
$\Upsilon_b = (\vec{x}_{\Upsilon_b}, a_{\Upsilon_b})$
$= ([x_{1\,\Upsilon_b} \quad x_{2\,\Upsilon_b} \quad x_{3\,\Upsilon_b} \quad x_{4\,\Upsilon_b} \quad x_{5\,\Upsilon_b} \quad x_{6\,\Upsilon_b}], a_{\Upsilon_b})$
Migration Pool
$\mathrm{P}_b = \begin{bmatrix} x_{1\,\Omega_b} & x_{2\,\Omega_b} & x_{31} & x_{41} & x_{5\,\Omega_b} & x_{6\,\Omega_b} \\ x_{1\,\Omega_b} & x_{2\,\Omega_b} & x_{32} & x_{42} & x_{5\,\Omega_b} & x_{6\,\Omega_b} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{1\,\Omega_b} & x_{2\,\Omega_b} & x_{3k} & x_{4k} & x_{5\,\Omega_b} & x_{6\,\Omega_b} \end{bmatrix}$
perceive
Integrate new knowledge from messages of other agents
decide
Perform local optimi-zation
act
Inform neighbors
Update Υ
Update Ω
Local Optimization
Inform neighbors

```
if        $f(\vec{x}_{\Upsilon_b}) < f(\vec{x}_{\Upsilon_a})$ then
          $\Upsilon_a \longleftarrow \Upsilon_b$
else if   $f(\vec{x}_{\Upsilon_b}) = f(\vec{x}_{\Upsilon_a})$ and $a_{\Upsilon_b} < a_{\Upsilon_a}$ then
          $\Upsilon_a \longleftarrow \Upsilon_b$
end if
```

```
$\mathrm{P}_a \longleftarrow bestK(G_a, \Omega_a)$   ➢get own best elements from G combined with Ω
$\mathrm{P}' \longleftarrow \mathrm{P}_a \cup \mathrm{P}_b$   ➢ assemble candidate pool
$\mathrm{P}'' \longleftarrow splice(\mathrm{P}')$   ➢derive splicing set from pool
$\mathrm{P}^* \longleftarrow \mathrm{P}' \cup \mathrm{P}''$   ➢combine both sets
$\Omega_a \longleftarrow \arg\min f(p) \setminus \{x_1, x_2\}, p \in \mathrm{P}^*$   ➢choose best as new Ω
if $\exists p \in \mathrm{P}^*: f(p) < f(\Upsilon_a)$ then
    $\Upsilon_a \longleftarrow \arg\min f(p), p \in \mathrm{P}^*$   ➢update Υ if excelled
end if
```

```
$G_a \longleftarrow \mathcal{De}(G_a, \Omega_a, \mu)$   ➢ run μ iterations of De to evolve G
$iter \longleftarrow iter + \mu$
if $\exists g \in G_a$: $f(g, \Omega_a) < f(\Upsilon_a)$ then
    $\Upsilon_a \longleftarrow \arg\min f(g, \Omega_a), g \in G_a$   ➢update Υ if excelled
end if
```

```
if $iter \leq min_{iters}$ or $\Upsilon_a$ changed then
    $\mathrm{P}_a \longleftarrow bestK(G_a, \Omega_a)$
    send $\kappa_a = (\mathrm{P}_a, \Upsilon_a)$ to neighbors
end if
```

Fig. 4.4.: Exemplary behavior in IDICE of agent A in one iteration after receiving a message from agent B

4.3 Impact of communication topologies

So far, this chapter has introduced the basic principles of parallelization in metaheuristics, given special attention to the Information-Sharing Cooperation Mechanism (ISCM), and presented two heuristics as exemplary algorithms and placed them in the framework of the ISCM. The CT is an important component of ISCM. The impact of different exchange topologies on algorithm performance, and in particular the design of dynamic topology adaptation at runtime, is a main subject of this thesis. The following literature review is intended to provide an overview of existing work on the influence and design of exchange topologies in the context of parallel cooperative metaheuristics and, in the process, to highlight the research gap that the approach presented in this thesis aims to fill.

The first studies of the CT for the COHDA algorithm were conducted during its development. In [Hin14], Hinrichs investigated the impact of several topologies, namely path graph, ring, small world [Str01], complete graph and a grid topology, on the performance of COHDA. In his experiments, the topology had little effect on solution quality. However, they did show a correlation between a high degree of topology connectivity and faster termination and higher communication overhead. The lack of impact on solution quality in Hinrichs' work may be due to the optimization problem examined, as there is certainly an impact on solution quality when using COHDA to solve continuous global optimization problems [HN20; HN21b; HN21a]. Nevertheless, Hinrichs identified the topology as a tuning parameter for the heuristic that can be customized according to a prioritization of the performance dimensions.

The effect of communication topologies on the performance of parallel cooperative metaheuristics has been studied mainly for the island model. In most works, different topologies for a given parallel heuristic are evaluated on several benchmark problems, and the topologies are ranked according to the achieved performance, which may involve different aspects [RIB10], [SZ18], [HC09], [HC11], [SJ15], [Wan+19].

Ruciński et al. investigated the effect of different migration topologies, including ring, cartwheel and hypercube topologies, on the performance of two different parallel global optimization algorithms cooperating via the island model [RIB10]. As a measure of performance, they used objective function values of the final solution or the evolution of the objective function values after migration periods. Both island-model based heuristics were designed with homogeneous solvers, i.e., either only instances of solvers using differential evolution or only instances of solvers performing simulated annealing with adaptive neighborhoods were used. They evaluated different topologies for both heuristics according to the performance obtained. Since the results varied widely, Ruciński et al. suggested that such studies should be conducted in the future for other heuristics and with more problem instances.

In [SZ18], Shia and Zhang presented a parallel elite biased framework (PEB framework) for parallel trajectory-based metaheuristics. They used the framework to implement a parallel variant of guided local search (PEBGLS) and conducted experiments to solve the symmetric traveling salesman problem (TSP) to demonstrate the competitiveness of PEBGLS and thus the applicability of the framework. In these experiments two types of topologies were applied: the bidirectional ring topology and the torus topology. They used different numbers of parallel processes, hence instances of guided local search, to solve the TSP with the two different topologies (adjusted to number of processes). They also varied the communication frequency and compared the results in terms of speed up an efficiency. For the problem instances considered, the torus topology led to a higher speedup in most cases and was also more efficient.

Hijaze and Corne [HC09] analyzed how different topologies affect the performance of an asynchronous distributed evolutionary algorithm (EA). They examined three different topologies. A master node is informed by the subpopulations about newly found best values. The master then forwards this best value to the population with the worst progress. The topologies differ in whether and how subpopulations are divided into subgroups that communicate directly. In this case, only one of the subpopulations communicates directly with the master node. They evaluated the effect of the topologies on the performance of the algorithm using 30-dimensional target functions (Sphere, Rosenbrock, Schwefel, Rastrigin, Griewank, Ackley [MY13][Li+13]). The success rate in finding the optimum was similar for all topologies, but better than for the standard single population EA (with equal total population size). The speedup was consistently highest for a topology variant that divides all subpolulations into groups of two. Based on their results, Hijaze and Corne hypothesized that the different performance was due to a different balance between exploration and exploitation.

In their follow-up work in [HC11], they introduced an online adaptation of the migration scheme in which the migration probability was adjusted based on the progress of subpopulations on islands. They performed the experiments with the same topologies and same objective functions, but higher dimensions (50d and 100d). The adaptive migration scheme is based on the progress of the subpopulations. As long as this is high, a low migration probability is used. If the progress decreases, the migration probability is increased. With the adaptive scheme, optimal solutions were regularly found in less time and with a higher success rate, suggesting that a balance between exploration and exploitation can be achieved by dynamically adjusting the migration mechanisms.

In [SJ15], Sanu and Jeyakumar conducted an empirical analysis on the performance of distributed Differential Evolution (DE) for several migration topologies. They used various topologies (basic ring and ring variants, star, cartwheel, torus and mesh) and multiple benchmark functions (e.g. Sphere, Schwefel (1,2,3), Rosenbrock, Rastrigin) to investigate the impact of the topologies on the performance of an island model DE. They considered not only the convergence speed and solution quality based metrics, but also the computational effort, i.e., the number of function evaluations. They concluded that no single topology is suitable for all optimization problems and took a first step towards linking characteristics of the search spaces to the performance of the topologies by roughly categorizing the functions (modality and separability) and assigning the best performing topologies in each case.

Wang et al. [Wan+19] studied the effects of various island model parameters, including migration topology, on solution quality, convergence speed, and diversity for large-scale optimization functions with 1,000 dimensions. The study considers a ring, lattice and a fully connected topology and ranks them with a non-parametric Mann-Whitney U-test. In this study the fully connected graph obtained the best rank. However, in that approach the migration rate is used to limit the number of neighboring islands migrated to at the same time. With 100 islands, this migration rate is at most 5. Therefore, not too much migration takes place in the fully meshed topology, but between which populations the exchange takes place is much more variable than in the other two topologies.

In [Arn+13], Arnaldo et al. performed a systematic analysis of the correlation between island topologies and problem structure. The problem structure in this sense is the structure in which the decision variables are interconnected. Arnaldo et al. analyzed both idealized and real-world problems as well as a real-world scenario and found that different problem structures require different island topologies. They also emphasized that topologies should be designed in a problem-dependent manner, but that the relationship between island topologies and problem structures is highly complex. In their work, however, they were able to construct the problem structure in the synthetic problem instances themselves. For the real-world scenario, no analysis of the problem structure was performed, but meta-optimization was used to select an optimal topology. Furthermore, the notion of problem structure covers only a small part of the actual problem characteristics as expressed by Fitness Landscape Analysis (FLA). Therefore, no systematic link between problem characteristics and favorable topologies has been established.

The presented research can be summarized as follows: First, different communication topologies affect the performance of the various parallel metaheuristics. Second, the notion of performance is mainly limited to the achieved solution quality and convergence speed. Third, the design of communication topologies leads to different balances between exploration and exploitation of the search space. In some cases, it is investigated how this balance can be improved by adjusting parameters such as migration frequency or migration rate. Finally, the effect of communication topologies depends on the characteristics of the underlying problem.

This leaves a research gap, as the following aspects have either not been studied thoroughly or not in combination:

- **No spatial distribution**: Usually no spatial distribution of the parallel cooperative metaheuristic is assumed. Thus aspects such as the *costs of collaboration*, especially the resulting message traffic are not considered.
- **No dynamic adaptation**: So far, no *systematic modeling* of different dynamic approaches for tuning exploration and exploitation in the search process and thus improving optimization performance has been presented.
- **No problem-dependant selection**: To date, there has been no systematic study of the correlation between problem characteristics and the influence of topologies, or how to select a topology tailored to the problem based on the problem characteristics determined by a comprehensive FLA.

This work aims to fill this research gap. In chapter 5 a systematic approach for the design of communication topologies and their dynamic adaptation at runtime is presented. The problem-dependent selection of the topology and its adaptation strategy based on a comprehensive set of FLA metrics characterizing the problem is the goal of the Topology Variant Optimization (ToVarO) in chapter 6. Thereby, several performance dimensions, namely the solution quality, the computational effort and also the communication overhead are considered.

5

Dynamic Communication Topology Adaptation

As described in section 4.3, the communication topologies of distributed optimization heuristics affect the degree of exploration and exploitation of the search space. Strongly meshed topologies lead to a high amount of information exchange between units. This leads to a fast convergence but bears the risk of a premature convergence into local optima. In contrast, with sparsely meshed topologies, the individual agents can evolve more independently. This leads to an intensified search in some areas of the search space. But in the worst case, these areas can be far away from the global optimum.

Several of the aforementioned papers in section 4.3 have pointed out that the CT could be used deliberately to control the flow of information in the system, thus supporting the search for good solutions to the problem at hand. As such, it could play an important role in enhancing the performance of cooperative search procedures [CT07; Cra19; RIB10]. This is the gist of hypothesis 1, which was presented in section 1.3.3 and will be examined in this chapter. Let us recall hypothesis 1:

Hypothesis 1
An adaptation of the communication topology at runtime of a distributed optimization heuristic solving an optimization problem with inherent decision set decomposition affects the search behavior of the algorithm and can improve its performance in terms of solution quality, computational effort and communication overhead.

The corresponding research question to the hypothesis is research question 1:

Research Question 1

How can dynamic ***communication topology variants*** *composed of an* ***initial topology*** *and a* ***topology adaptation strategy*** *be modeled, and how do these variants affect the various performance dimensions of distributed heuristics?*

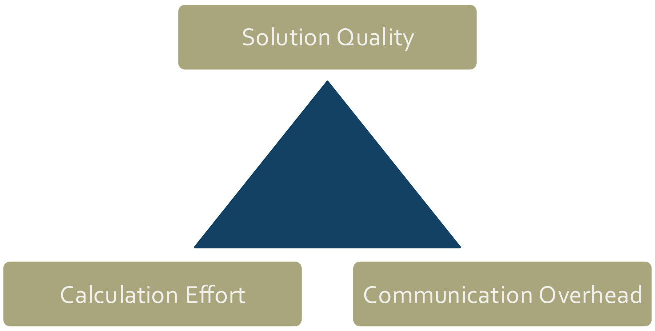


Fig. 5.1.: Performance dimensions of cooperative search algorithms

The goal of this chapter is to present the modeling approach for the topology variants and show its effects in a preliminary study (before the actual evaluation takes place in part IV). Since the performance of distributed algorithms is multifaceted, this chapter first discusses several performance dimensions that are affected by the CT. Then, in section 5.2, the modeling approach for topology variants is presented, including both the initial topology construction and the possible runtime adaptation. Finally, in section 5.3, the effects of the topology variants on the performance dimensions are demonstrated with several examples.

5.1 Dimensions of algorithm performance

The first measure of the performance of optimization heuristics is the quality of the solution. The time and computational effort required for optimization are also evident measures.[1] For cooperative optimization heuristics that are in fact spatially distributed, the resulting communication traffic is also important. Between these performance indicators there is naturally a certain trade-off. For instance, a higher solution quality is usually accompanied by longer computation times and may require more communication between the distributed entities. Figure 5.1 visualizes the tension between the three dimensions of performance.

These performance dimensions are also reflected in the requirements from the motivating use cases. The scheduling of energy sources leads to direct control commands for the assets. The quality of the solution therefore influences the utilization of the power grid. Depending on the specific use case (i.e. congestion management or voltage control), there are also different requirements for the duration of the

---

[1] In evaluations in this work, the calculation effort is used as a criterion, rather than the required convergence time. The effort is determined by the sum of required iterations or local optimization runs of all agents. In contrast to the convergence time, this results in an indicator that is much more independent of the hardware used and any other parallel utilization.

solution finding process. Sufficient solution quality achieved in an acceptable time can thus be directly linked to the reliable operation of power grids. The resulting message traffic as well as the computational effort are also not to be neglected. For example, if there is a high traffic load on the communication network due to external influences, optimization with low message traffic is advantageous. Similarly, the computation is performed on edge devices distributed in the field, which have limited computational capabilities. There are other aspects that could be included, such as robustness to different types of impairments. However, the goal of this work is to extend the commonly used performance definitions of solution quality and computational effort to include the dimension of collaboration costs by taking into account the communication effort, which is particularly relevant in spatially distributed systems. An extension of the performance dimensions considered would be particularly appropriate when transferring the approach to real-world applications.

The CT affects all three considered performance dimensions. A high degree of connectivity in the CT results in higher message traffic. However, in many cases, the convergence time is also reduced, so that the increased message volume decreases more quickly. The topology also influences the degree of exploration and exploitation during the search. The balance between exploration and exploitation in turn affects the quality of the solution found, the speed at which the heuristic converges, and consequently the overall computational effort.

Figure 5.2 uses a small example with COHDA (see section 4.2.1) to illustrate how the degree of exploration and exploitation is affected by the topology. In the case of a strongly meshed topology, like the fully connected graph displayed in the top part of the figure, an agent is likely to receive messages from all or most of the other agents in each iteration. Therefore, after the agent's "perceive" phase during which it processes all these messages, its perceived system state Ω changes fundamentally, as all foreign decision variables may have been updated. Since Ω is the basis for the agent's local optimization, this means that in each iteration the agent will find a completely different starting point for this optimization in the high-dimensional overall search space, in other words, a completely different local fitness landscape. This corresponds to the principle of coupled and dynamic fitness landscapes introduced in section 2.3. Thus, in strongly meshed topologies, many different variants of the local search space are explored from the point of view of the individual agents. However, the search behavior of the heuristic as a whole tends to be exploitative, since all agents remain relatively "close" to each other in the high-dimensional overall search space. In contrast, in sparsely meshed topologies, such as the ring topology shown second, fewer messages from other agents reach an agent within an iteration. In the perceived system state Ω, therefore, few foreign decision variables change. Thus, the starting points for local optimization in successive iterations are not too far apart. This means that the local fitness landscape of the agents changes less from iteration to iteration than in strongly meshed topologies. From this point of view, an exploitation of the space of possible fitness landscapes for individual agents takes place. However, if all agents are considered as a whole, and thus the search in all problem dimensions, this topology tends to diversify the search.

In IDICE (see section 4.2.2), the second heuristic considered in this thesis, the degree of connectivity has similar effects. In highly meshed topologies, an agent receives migration pools P from many other agents. In a migration pool P_i, the decision variables of the sending agent i vary in each entry, while the values for all other variables are the same. The combined solution pool P^*, which contains all the migration pools received plus the recombinations created by splicing, is therefore very large and highly heterogeneous. Since the best solution from P^* becomes the new Ω, the probability of major changes in Ω is very high with such a large and diverse pool. With a weakly meshed topology, an agent receives much fewer migration pools. Accordingly, P^* is much smaller and less heterogeneous. The probability of an Ω being closer to its predecessor is therefore higher.

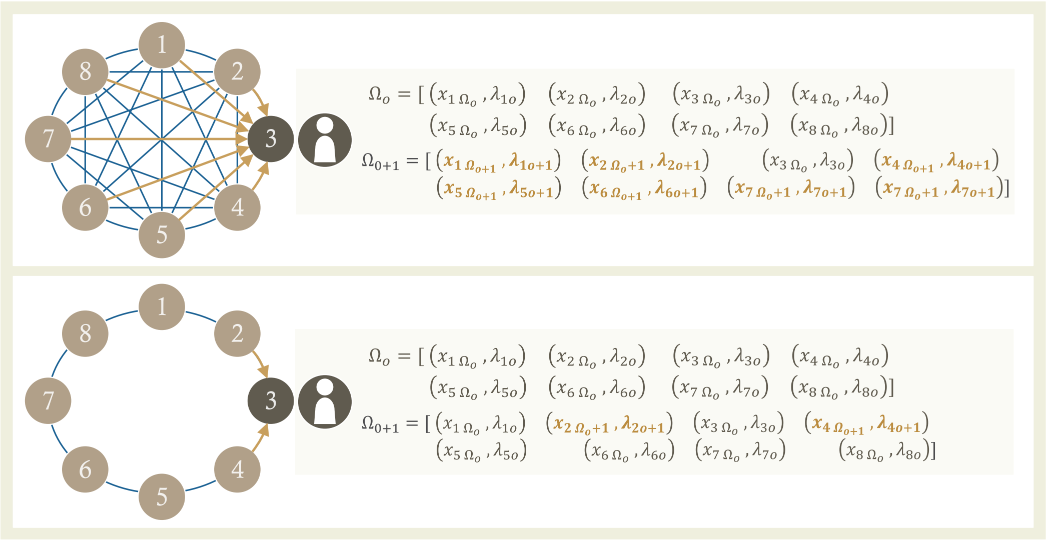

1
2
3
4
5
6
7
8
$\Omega_o = [\,(x_{1\,\Omega_o}, \lambda_{1o})\ (x_{2\,\Omega_o}, \lambda_{2o})\ (x_{3\,\Omega_o}, \lambda_{3o})\ (x_{4\,\Omega_o}, \lambda_{4o})$
$(x_{5\,\Omega_o}, \lambda_{5o})\ (x_{6\,\Omega_o}, \lambda_{6o})\ (x_{7\,\Omega_o}, \lambda_{7o})\ (x_{8\,\Omega_o}, \lambda_{8o})]$
$\Omega_{0+1} = [\,(x_{1\,\Omega_{o+1}}, \lambda_{1o+1})\ (x_{2\,\Omega_o+1}, \lambda_{2o+1})\ (x_{3\,\Omega_o}, \lambda_{3o})\ (x_{4\,\Omega_{o+1}}, \lambda_{4o+1})$
$(x_{5\,\Omega_{o+1}}, \lambda_{5o+1})\ (x_{6\,\Omega_{o+1}}, \lambda_{6o+1})\ (x_{7\,\Omega_o+1}, \lambda_{7o+1})\ (x_{7\,\Omega_o+1}, \lambda_{7o+1})]$
1
2
3
4
5
6
7
8
$\Omega_o = [\,(x_{1\,\Omega_o}, \lambda_{1o})\ (x_{2\,\Omega_o}, \lambda_{2o})\ (x_{3\,\Omega_o}, \lambda_{3o})\ (x_{4\,\Omega_o}, \lambda_{4o})$
$(x_{5\,\Omega_o}, \lambda_{5o})\ (x_{6\,\Omega_o}, \lambda_{6o})\ (x_{7\,\Omega_o}, \lambda_{7o})\ (x_{8\,\Omega_o}, \lambda_{8o})]$
$\Omega_{0+1} = [\,(x_{1\,\Omega_o}, \lambda_{1o})\ (x_{2\,\Omega_o+1}, \lambda_{2o+1})\ (x_{3\,\Omega_o}, \lambda_{3o})\ (x_{4\,\Omega_{o+1}}, \lambda_{4o+1})$
$(x_{5\,\Omega_o}, \lambda_{5o})\ (x_{6\,\Omega_o}, \lambda_{6o})\ (x_{7\,\Omega_o}, \lambda_{7o})\ (x_{8\,\Omega_o}, \lambda_{8o})]$

Fig. 5.2.: Updating the perceived current system state Ω of agent 3 in COHDA; In strongly meshed topologies, it is likely that (almost) all decision variables are updated in each iteration. In weakly meshed topologies, it is likely that only a few decision variables are updated. (Note that decision variables from agents that are not directly connected may also be updated. This has been omitted here to emphasize the different behavior.)

In the following, an approach is presented that aims to use this relationship as a guide for the optimization process in order to improve the performance of the algorithms by targeting its different dimensions.

5.2 Topology variants

The goal of the presented approach is to achieve an advantageous balance between exploration and exploitation that in the best case leads to a high solution quality while keeping the required time and the costs of collaboration at acceptable levels. Of course, the actual meaning of "acceptable" depends on the specific application of the optimization heuristic. In addition the approach should allow to prioritize the performance dimensions in different order and adapt the optimization process accordingly. Which topology achieves the desired effect depends on the optimization problem (i.e., the fitness landscape), the optimization heuristic, and the prioritization of the performance dimensions.

A CT variant involves two design issues, which are presented in more detail in the next sections:

1. **(Initial) Topology**: The topology, i.e. especially its degree of meshing, must be chosen in an appropriate way to achieve the best possible results for a chosen prioritization of the performance dimensions. Let $G = (V, E)$ denote the bidirectional graph that represents the communication topology. V is the set of n nodes, where each node is assigned to an agent and thus to one part of the distributed solver. E is the set of edges. An edge between two nodes indicates direct communication between the two agents assigned to the nodes.

2. **Topology adaptation strategy**: A variation of the degree of exploration and exploitation during the search of a heuristic often leads to better optimization results [ČLM13]. Therefore, it is necessary to decide whether such an adaptation is beneficial, and if so, in which way it should be performed.

5.2.1 Starting in a small world

In [WS98], Watts and Strogatz presented their random graph generation model for creating small-world topologies. They coined the term "small-world" networks in reference to the small-world phenomenon popularized by Milgram [Mil67]). Milgram conducted a well-known experiment in which letters were to be delivered between strangers in different regions of the United States. Each participant was asked to forward the letter to a close acquaintance who he or she thought was more likely to have a connection with the target. The letters that reached the target had changed hands only about five to six times. Watts later investigated the small-world hypothesis further, e.g. in [Wat04; WS98], and also studied the properties of other real-world networks that are often highly clustered but have small characteristic path lengths. Since the constructed small-world networks should reflect the properties of real-world networks, the goal was to interpolate between regular and random networks to achieve both high clustering and small characteristic path lengths. Examples of such small-world networks include the neural network of the worm Caenorhabditis elegans, the power grid of the western United States, and the collaboration graph of movie actors [WS98]. Besides resembling real-world networks, models of dynamical systems with small-world coupling also possess properties, such as increased signal propagation speed, computational power, and synchronizability. Small world networks can be highly clustered, such as regular lattices, but still have small characteristic path lengths, such as random graphs. Watts and Strogatz quantify the structural properties of the graphs by

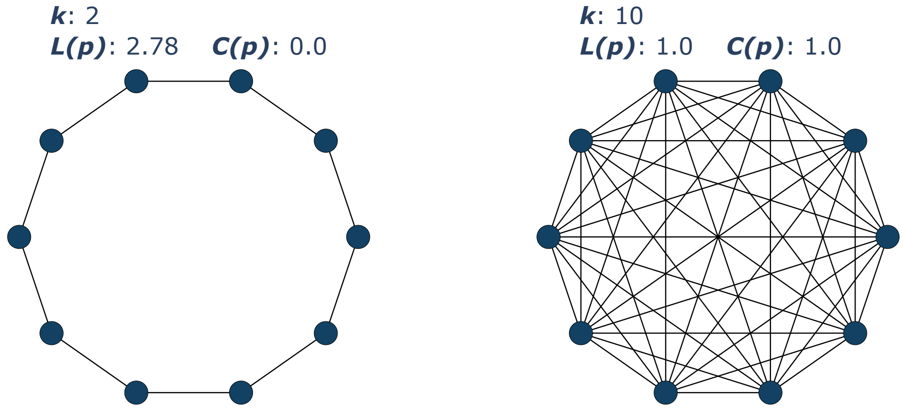
- the **characteristic path length** $L(p)$: measures the typical separation between two vertices in the graph (a global property)

- the **clustering coefficient** $C(p)$: measures the cliquishness of a typical neighbourhood (a local property).

Since small world networks are intended to combine the properties of regular and fully random networks, they are also generated by a combination of these. For a graph with n vertices, a ring lattice with k edges per vertex is generated first. Thereon, each edge is rewired with probability p. Thus, by the two parameters k and p it is possible to control at which level the structural properties $L(p)$ and $C(p)$ emerge. A graph with sufficiently large k ($n \gg k \gg ln(n) \gg 1$) and $p = 0$ will be a regular lattice which is a highly clustered and $L(p)$ grows linearly with n. If $p = 1$ the graph will be a random network which is a poorly clustered small world where $L(p)$ grows only logarithmically with n.

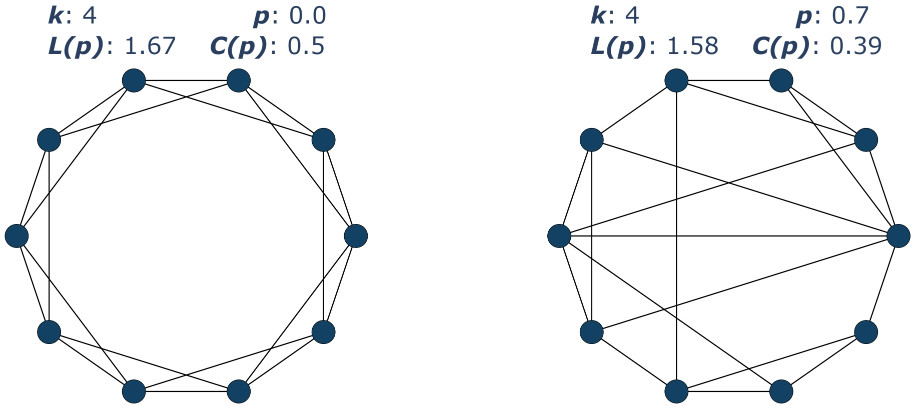
In the present work, the Watts-Strogatz small-world graph was slightly adapted. Before the rewiring step, a subset of edges forming a ring topology is selected as non-removable edges. This design decision was made with runtime topology adaptation in mind (see next section). The reasoning behind this is as follows: In order for a distributed optimization heuristic to work, the CT must be interconnected. The topology type with the fewest edges that satisfies this criterion is the path graph. The ring topology possesses only one more edge, but showed much better results than the path graph in preliminary work [HN20]. Therefore, the ring topology is set as the minimum layout.

Figure 5.3 shows example topologies that were generated by the adapted approach. For the two extreme cases, the ring topology and the fully connected graph, p has no effect. The other graphs show various examples of the graphs generated by the interplay of k and p, resulting in different characteristic path lengths and clustering



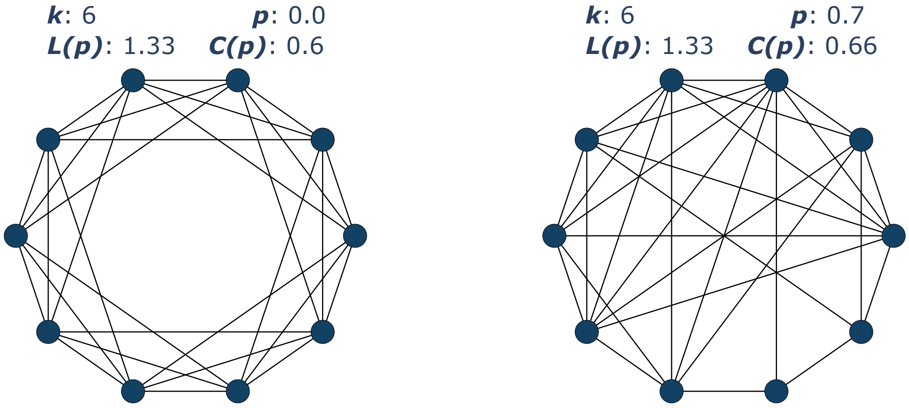
(a) $k = 2$ leads to a ring topology, while p has no effect

(b) $k = n$, leads to a fully meshed topology, while p has no effect



(c) regular ring lattice with $k = 4$ and no rewiring

(d) ring lattice from (c) after rewiring with $p = 0.7$



(e) regular ring lattice with $k = 6$ and no rewiring

(f) ring lattice from (e) after rewiring with $p = 0.7$

Fig. 5.3.: Examples of the modified Watts–Strogatz small-world graph for 10 nodes displaying the effects of different values for k and p

coefficients. This illustrates that using only the two parameters k and p, a wide variety of topologies, from rings to fully meshed graphs, can be generated. At the same time, despite the rewiring by p, the number of edges remains the same (unlike other small world graph generation methods like the Newman–Watts–Strogatz small-world graph [NW99]) and makes this property easily controllable, which is necessary for the adaptation strategies in the next section. Thus, the adapted Watts-Strogatz small-world graphs provide an appropriate and easy-to-parameterize model for generating a wide range of (initial) topologies.

5.2.2 Adaptation strategies

Many metaheuristics adjust parameters at runtime (parameter control) to allow a transition from exploration to exploitation [EHM99; ČLM13]. An example of this is the adjustment of the temperature parameter T for simulated annealing (SA) [BR03]. SA is a trajectory-based metaheuristic and is based on the annealing process of metals and glass. At each iteration a solution from the current neighborhood is randomly chosen. If the fitness of this new sample is better than the existing one, it is accepted as the new solution. If the fitness is worse, the new sample is still accepted with a probability that depends on the difference between the fitness values and the temperature parameter. The algorithm starts with a high temperature. This leads to an increased probability of selecting inferior solutions and thus to an erratic exploration of the search space. Over time, the temperature cools down continuously, resulting in a transition to exploitative behavior [BR03].

As discussed, the CT for cooperative search heuristics also influences the degree of exploration and exploitation. Therefore, the CT can be adapted during runtime to achieve the desired transition. However, due to the domain decomposition, a topology has different effects on the local optimization of one agent and the collaborative global optimization of all agents. Therefore, it is difficult to assess in advance which type of adaptation is beneficial for a specific problem. Figure 5.4 illustrates the adaptation process from the global and local point of view.

Analogous to the cooling process in SA, a high temperature may correspond to a strongly or sparsely meshed topology, while a low temperature corresponds to the respective counterpart. The shift from exploration to exploitation could thus be approached either from the perspective of individual agents or from that of the entire multi-agent system. The former means starting with a highly meshed topology for an initial exploration phase and then moving to exploitative behavior by removing edges. The second involves starting with a sparsely meshed topology for the initial exploration phase and proceeding to exploitative behavior by adding edges. The inclusion and variable weighting of the performance dimensions leads to a further

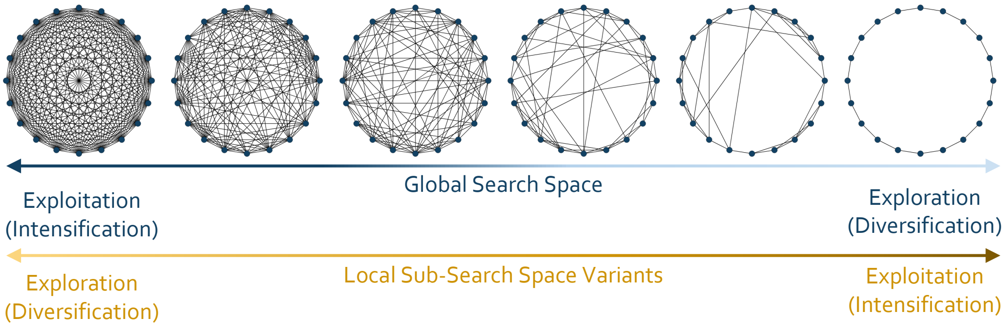

Global Search Space
Exploitation (Intensification)
Exploration (Diversification)
Local Sub-Search Space Variants
Exploration (Diversification)
Exploitation (Intensification)

Fig. 5.4.: Dynamic topology adaptation from the perspective of global optimization involving all agents and from the perspective of individual agents. Individual agents explore or exploit the space of their own subsearch space variants resp. the space of different variants of their coupled and dynamic fitness landscapes.

differentiation of which adaptation is most beneficial under which circumstances. In addition, for some combinations of problem characteristics and heuristics, topology adaptation is not necessary, since it only adds unnecessary complexity. Hence, the presented approach implements different modes that either omit adaptation or frame the topology adaptation by removing or adding edges.

These modes include:

- **static**: no adaptation is performed and the initial topology is used throughout the optimization process
- **decrease**: starts with a rather strongly meshed topology, transitions to small world intermediate stages by removing edges, and ends with a ring to exploit the most promising regions in the solution space
- **increase**: starts with a rather weakly meshed topology, transitions over small world intermediate stages by adding edges, and ends with a complete graph

The modes **decrease** and **increase** are both a form of *deterministic* parameter control, as the value of a parameter, i.e. the number of edges, is modified by a deterministic policy without regard to the current state of the search.

If topology adaptation is performed, the general procedure is inspired by the cooling process in SA. To model the cooling process, SA uses a so-called cooling schedule to determine the temperature at different stages of the algorithm. Similarly, a *topology schedule* is used that specifies the number of edges in the CT for different stages of the distributed algorithm. A cooling schedule for SA is determined by

- the **initialization parameter**: the initial temperature T_0
- the **cooling**, i.e. the **adaptation**: the rate at which the temperature decreases, respectively the temperature steps in the schedule

- the **equilibrium state**: the criterion that controls when the transition to the next temperature level occurs

Exactly these aspects must also be determined for the *topology schedule*, whereby the special requirements of the different modes must also be taken into account. These aspects will therefore be discussed in more detail below.

Initialization parameter: The initialization parameter for the *topology schedule* are the number of edges in the initial topology $G_0 = (V, E_0)$. The initial topology must therefore be chosen appropriately for the mode, i.e. strongly meshed for *decrease* and weakly meshed for *increase*. The initial number of edges $|E_0|$ thus lies in the range between a complete graph $|E_0| = \frac{n \cdot (n-1)}{2}$ and a ring topology $|E_0| = n$.

Adaptation: There are several approaches to modeling cooling process of SA in the literature. Geometric functions are particularly popular to model cooling, whereas logarithmic functions are considered too slow for practical application, although they theoretically converge to a global optimum [Tal09]. Considering that the initial number of edges is much smaller than usual starting temperatures, a slower reduction, respectively increase, seems appropriate. A combination of linear, geometric and logarithmic reduction functions was chosen. The equations 5.1 and 5.2 display the functions that determine the number of edges δ at each schedule step for the modes *decrease*, and *increase*:

$$\textbf{decrease}: \quad \delta_{i+1} = |E_i| - \frac{|E_0|}{ln(i+2)} \cdot \alpha, \quad with \ \ \alpha \in]0,1] \tag{5.1}$$

$$\textbf{increase}: \quad \delta_{i+1} = |E_i| + \frac{|E_0|}{ln(i+2)} \cdot \alpha, \quad with \ \ \alpha \in]0,1] \tag{5.2}$$

The index i represents the index of the step in the topology schedule. δ_i determines the number of edges that should be part of the graph G_i in step i.[2] Hence, upon transition to the next schedule step, a new CT graph is constructed such that

$$G_{i+1} = (V, E_{i+1}), \quad with \quad |E_{i+1}| = \delta_{i+1} \tag{5.3}$$

The parameter α controls how many steps the topology schedule includes. If it is close to 0, only a few edges are removed or added in each step, leading to a slow change in connectivity. If it is equal to 1, the number of edges is reduced or increased in large steps, resulting in a rapid shift in connectivity. The edges to be removed or added, are selected randomly. In the *decrease* mode, the final ring topology is preserved.

---

[2] Note that when calculating the schedule steps, an offset of 2 is added for the calculation of $ln(i)$ to prevent division by undefined values or 0

The figures 5.5 and 5.6 show schedules for the two adaptive modes, which are based on the same initial topology and use the same α. The initial topology for $n = 100$ was constructed with $k = 50$, thus leading to an initial number of edges of $|E_0| = \frac{n \cdot k}{2} = 2500$. The two schedules for the *decrease* and *increase* mode are processed from left to right during optimization. The figures 5.5 and 5.6 also show the impact of the parameter α on the granularity of the schedules. Depending on n and k, a maximum α can be calculated where the schedule consists of only a single step. This is the case, for example, when more edges are to be removed in the first step than can be removed, i.e. $\delta_1 = |E_0| - \frac{|E_0|}{ln(2)} \cdot \alpha \leq n$. This is the case when $\alpha \geq ln(2) \cdot (1 - \frac{2}{k})$. Therefore the maximum α for $n = 100$ when starting from a complete graph is about 0.68. Thus, the chosen value of $\alpha = 0.5$ still shows coarse-grained steps, while the smaller value of $\alpha = 0.1$ yields a much more fine-grained resolution.

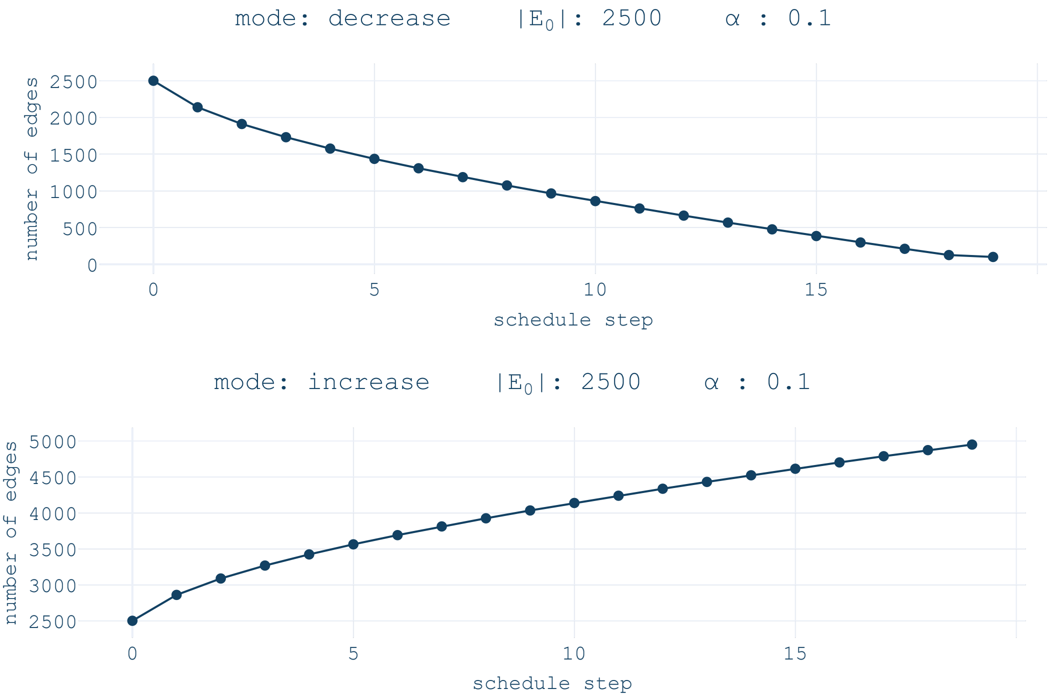


Fig. 5.5.: Exemplary *topology schedules* for $n = 100$ displaying the three modes. Both start from the same initial topology with $k = 50$ and thus $|E_0| = 2500$. Both use the same $\alpha = 0.1$

equilibrium state: The equilibrium state aka. the transition criterion defines when a transition from one schedule step to the next occurs. Common conditions for a temperature adaptation in SA are counting of iterations, acceptances, rejections or a combination of these [Tal09]. For the *topology schedule* in the *decrease* and *increase* mode, a simple approach is used, where the number of local searches (equivalent to the number of iterations) since the last transition is counted. As soon as this number becomes larger than n, the topology is adjusted. This rapid transition was chosen since it showed the best results in preliminary experiments.

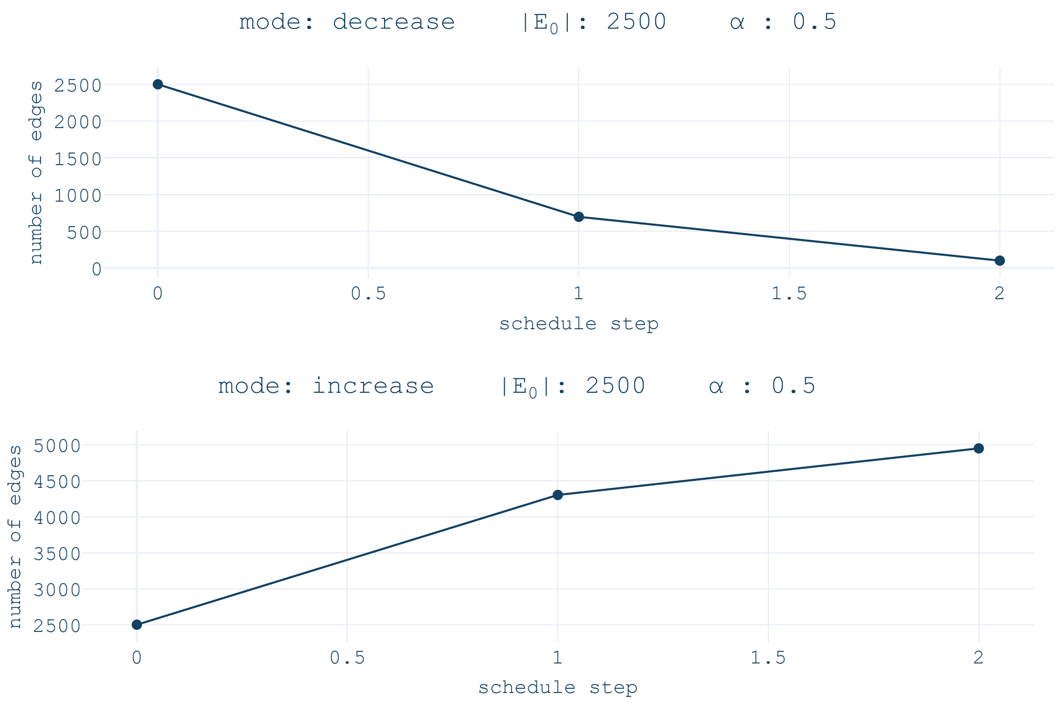


Fig. 5.6.: Exemplary *topology schedules* for $n = 100$ displaying the three modes. Both start from the same initial topology with $k = 50$ and thus $|E_0| = 2500$. Both use the same $\alpha = 0.5$

The next section is intended to give an impression of the impact that different topology adaptation strategies can have on the performance of the optimization heuristics, before the more detailed evaluation in part IV.

5.3 Observed effects of topology variants

The claim of hypothesis 1 is that the presented topology variants, consisting of the initial topology and the adaptation strategy, lead to a different information dissemination in the course of the distributed optimization process. This information dissemination affects the performance dimensions, i.e., the solution quality, the required runtime, and the communication overhead. In this section, a brief preliminary study is conducted to support this claim with examples, before it is evaluated in part IV in combination with the results of the distributed FLA. For this purpose, several optimization runs were performed with COHDA and IDICE. For comparison, each optimization for each problem instance was performed with the topology variants presented in table 5.1. In the spectrum of possible variants that emerges from the presented methodology for creating the initial topology and its adaptation, the selected variants represent extreme points that differ greatly from each other. The variant *static* equals a moderately meshed small world topology and serves as reference. Both variants with mode *decrease* start from a complete graph, but decrease connectivity at different rates. The same holds for the two variants with the mode *increase*, which proceed from a ring topology.

Topology Variant	**Initial Topology**		**Topology Adaptation**	
	k	p	mode	α
static	24	0.5	static	-
decrease slowly	50	0.5	decrease	0.05
decrease fast	50	0.5	decrease	0.7
increase slowly	2	0.5	increase	0.05
increase fast	2	0.5	increase	0.7

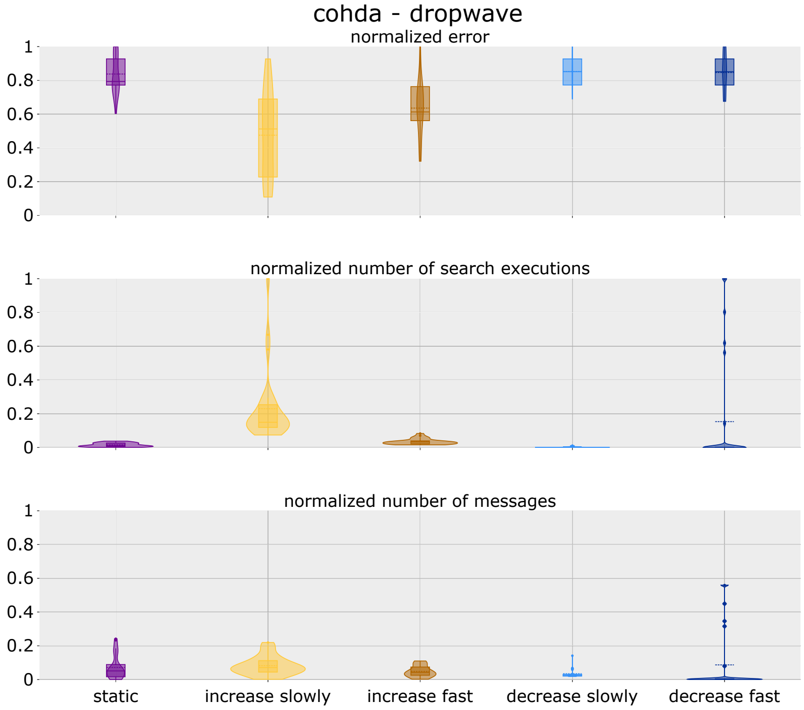
Tab. 5.1.: Employed topology variants for 100-D benchmark problems

Moreover, *Drop Wave* and *Sargan* were chosen as the basis for the problem instances. The global structure of the two basis functions is very different, e.g., multi-funnel vs. single-funnel shape, which results in composite spaces based on *Drop Wave* being very heterogeneous, while composite spaces based on *Sargan* are rather homogeneous. The first three composite spaces of both functions were used. The optimization runs were performed for each problem instance with each topology variant with 3 different random seeds each for the topology variant and the optimization, i.e. 9 times in total with the same setting.

Figure 5.7 and fig. 5.8 show the results for several performance dimensions for COHDA and IDICE. All results were jointly normalized by min-max scaling. The results for the Drop Wave-based composite spaces for COHDA in fig. 5.7a show that the *increase slowly* variant outperforms the others in terms of solution quality. However, this comes at the cost of computational time and effort, as well as a higher communication overhead. Thus, depending on the weight of the performance dimensions, the *increase slowly* or *increase fast* variant would be preferred. The results for the Sargan-based composite spaces in fig. 5.7b display a different picture. The two variants that scored best in terms of solution quality for *Drop Wave* perform worst in this case. The two *decrease* variants and the static one perform similarly well in terms of solution quality. Considering the other performance dimensions, the static topology would be preferable here.

The results for IDICE in fig. 5.8 show different effects of the topology variants. For the Drop Wave-based problem instances in fig. 5.8a, the two variants with the mode *increase* likewise perform best with respect to the solution quality. With IDICE, however, these variants also have an advantage regarding the other two performance dimensions. In addition, the *increase fast* variant is slightly better in terms of solution quality, whereas in COHDA the slow increase performed significantly better. For

(a) Drop Wave



(b) Sargan

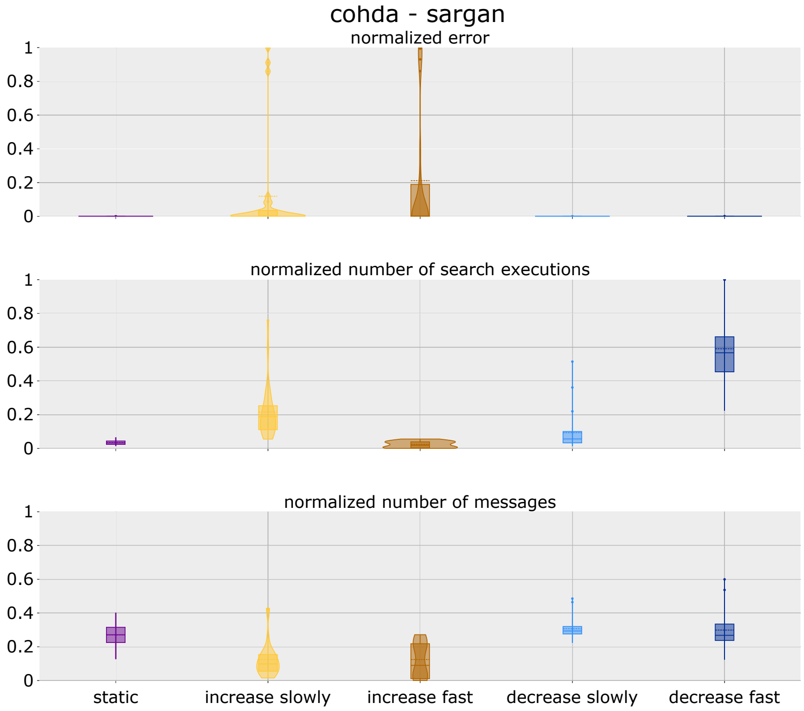
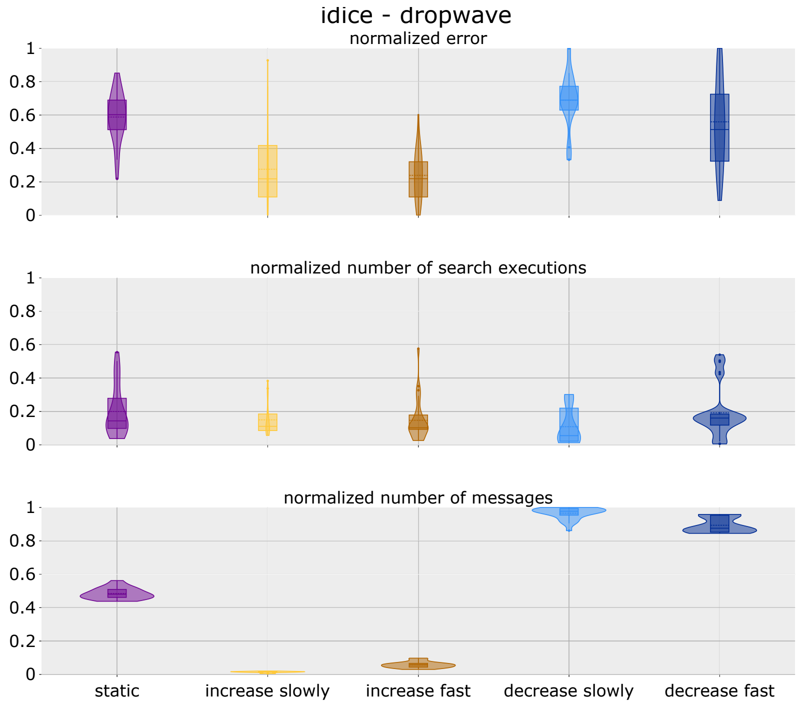


Fig. 5.7.: Results for optimization runs with COHDA

(a) Drop Wave



(b) Sargan

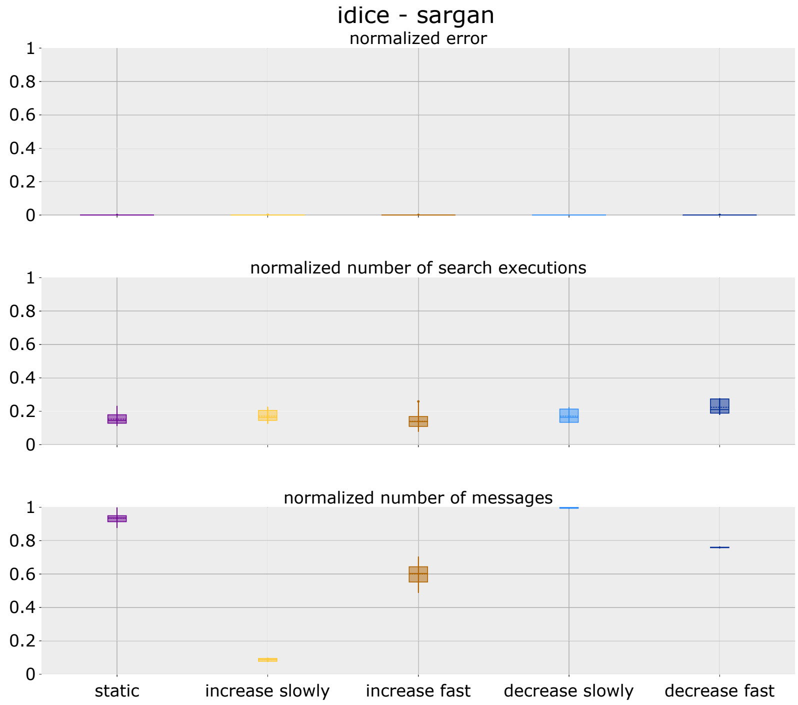


Fig. 5.8.: Results for optimization runs with IDICE

the Sargan-based problem instances in fig. 5.8b, optimization with IDICE leads to excellent solution quality with all topology variants. Since the variants are equivalent concerning the solution quality, either *increase fast* or *increase slowly* would be preferred, depending on whether the computation time or the number of messages are weighted more strongly.

These examples show on the one hand that the topology variants influence the course of the optimization and thus also affect the performance dimensions. On the other hand, it becomes clear that the choice of a suitable, or preferably the best, topology variant must be made with respect to both - the applied heuristic and the problem at hand. The following chapter therefore investigates whether systematic relationships exist between the properties of optimization problems - quantified by Fitness Landscape Analysis (FLA) - and the effect of topology variants on performance dimensions. The goal is to learn these relationships and thus be able to make an optimized topology variant selection for new problems.

Part IV

Learning Optimal Topology Variants

This part of the thesis is dedicated to the investigation of the causal relationship, postulated in research hypothesis 3, between the properties of distributed optimization problems and the effects of the CTVs. The hypothesis should be strengthened by training machine learning models that allow selecting an appropriate topology variant for a given problem instance and a given prioritization of performance dimensions. Therefore, chapter 6 first examines the relevance of FLA and CTparameters for predicting performance dimensions. Then, a machine learning model is trained to learn the hypothesized causal relationship. The predictive capabilities of the model and its ability to select the CTV for a given problem are examined. This will also serve to evaluate the two artifacts obtained from the investigation of the research questions 1 and 2, i.e. the modeling of the topology variants and the distributed fitness landscape analysis.

6

Topology variant optimization (ToVarO)

The Topology Variant Optimization (ToVarO) is the process of determining the best topology variant(s) based on the distributed fitness landscape metrics and a given prioritization of performance dimensions. The development objective of this thesis was stated in section 1.4. In the presented system architecture, the behavior of the productive system, i.e. the optimization heuristic, is monitored and, if necessary, subject to control intervention by a controller/observer structure following the paradigm of controlled self-organization from organic computing. ToVarO belongs to the tasks of the controller, which can select the initial topology and also modify it at runtime based on observed information. Figure 6.1 shows the system architecture for the development objective as presented in section 1.4.

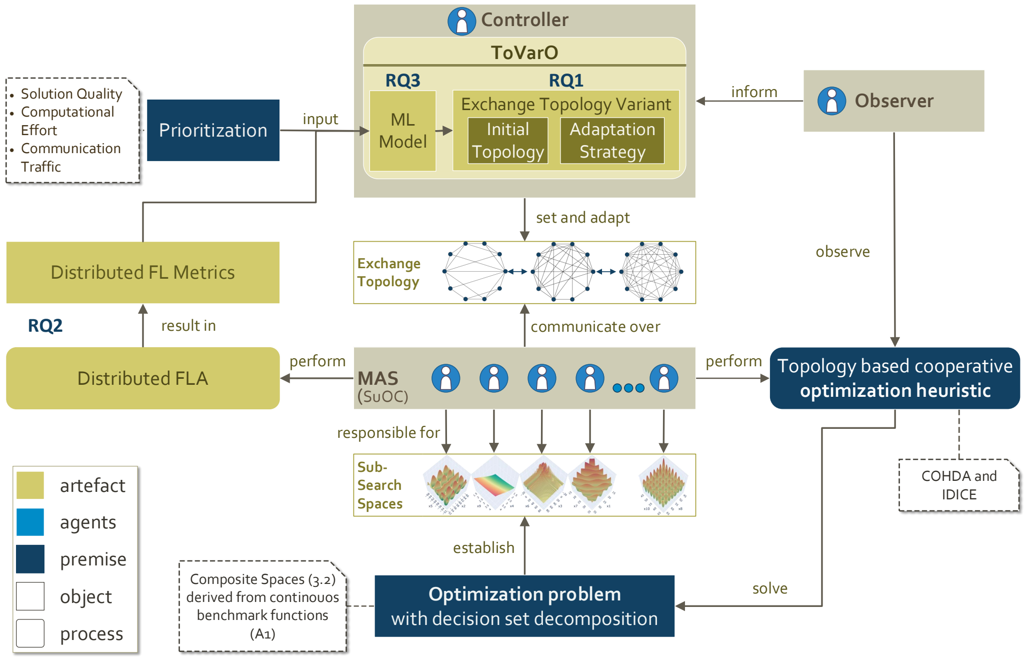


Fig. 6.1.: Development Objective a presented in section 1.4: given an optimization problem, an optimization heuristic, and a prioritization of performance dimensions as premise, the agents first perform a distributed FLA for their decomposed search spaces, based on which the controller can select an initial CT and later adjust the topology at runtime if necessary (ToVarO). This serves to improve the performance of the algorithm in a problem-specific way.

Of course, this problem-specific selection is only possible if the three hypotheses postulated in section 1.3.3 hold, i.e.

1. The topology variants have an effect on the performance of the optimization heuristics in the performance dimensions (as exemplified in section 5.3).
2. The distributed fitness landscape analysis can determine meaningful features that can serve as a basis for parameter tuning and control (preliminary investigation by comparison with centrally calculated metrics in appendix B.1 and correlation analysis in section 3.7).
3. There is a relationship between the problem characteristics and the effects of the topology variants.

The full-length hypothesis 3 states:

Hypothesis 3
There is a causal relationship between the characteristics of the distributed optimization problems and the effects of the communication topology variants. Thus, the communication topology variant of a distributed optimization heuristic can be selected in a problem-dependent manner to increase the optimization performance according to a predefined prioritization of performance dimensions based on a set of distributedly computed fitness landscape metrics.

The goal of the topology variant optimization (ToVarO) is to combine these three aspects of the three hypotheses by developing a machine learning model that can determine a suitable and, in the best case, optimal topology variant for a problem instance based on the distributedly computed FLA metrics. If successful, this will further support the idea that the distributed FLA metrics contain sufficient information to be used for parameter tuning, and that the effect of topology variants is problem specific and can be used to systematically improve algorithm performance. The question of how machine learning models can be used for this purpose is the subject of the third research question:

Research Question 3

How can machine learning models be used to ***show*** *the* ***causal relationship*** *between the characteristics of distributed optimization problems and the impact of communication topology variants, thus enabling the problem-specific selection of communication topology variants?*

The following qualities of a Machine Learning (ML) model can corroborate this relationship:

- **Good prediction results**: This means that the model has learned patterns that allow it to predict output parameters based on input parameters with low error rates.
- **Visible differences between different problem instances and CTVs**: If a model shows clear and comprehensible differences between different topologies for different problem instances, then this confirms that different CTVs have different effects depending on the problem characteristics. Furthermore, it shows that the model can distinguish the problem instances using the FLA metrics and thus affirms the suitability of the FLA metrics for problem-specific parameter optimization.

Optimally, such a ML model would take the metrics of the fitness landscape analysis as input and return the best variant of the CT as output. Such an approach was taken in previous publications [HN21b; HN21a]. In both works, optimization runs for global optimization on continuous benchmark functions were performed with COHDA, once on the full problem domain [HN21b] and once on composite spaces constructed in the same way as presented in this thesis [HN21a]. The optimization runs were performed using several variants of static exchange topologies, such as path graph, ring topology, 2-D lattice topologies, small world, and fully connected graphs. In [HN21b] a reduced version of the *decrease* topology variant was also employed. Several runs were performed for each problem instance with each configuration, and the best CT was selected either by solution quality alone or by a combined consideration of solution quality and execution time. Decision trees were trained based on fitness landscape metrics computed in a centralized [HN21b] or distributed manner [HN21a] to select an appropriate topology based on the problem characteristics. In both works, a correlation was found between certain FLA metrics and the degree of connectivity of the best-fit topologies (with accuracies of the trained models ranging from 74% to 88%). However, only a few topologies were used in these works, and their properties varied considerably. The extended design of CTVs as presented in this thesis has increased the complexity of the learning problem. Instead of a few very different static topology variants (and simple dynamic approaches), there are now a large number of topology variants with different initial conditions and different behavior at runtime. Also, more problem instances are considered, which improves the data base and facilitates other learning methods.

In the following section 6.1, the experimental setup is presented. Section 6.2 analyzes the relevance of FLA metrics and CTVs for predicting performance metrics. The predictive power of the ML models used is then examined in section 6.3. This includes evaluating a targeted selection of topology variants based on the predictions and analyzing the impact of noise on system performance.

6.1 Experimental setup

The details of the experimental setup for the evaluation of the concepts developed in the previous parts are presented below. Section 6.1.1 provides an overview of the problem instances, the FLA metrics, and the parameter settings of the optimization heuristics, including the CTVs. In addition, section 6.1.2 presents the metrics used to quantify the performance dimensions. The challenges of labeling the resulting experimental data are then discussed in section section 6.1.3.

6.1.1 Problem instances, FLA metrics, and optimization parameters

To investigate the hypothesized causal relationship between the properties of optimization problems and the effects of CTVs, 20 composite spaces were created based on each of the 23 benchmark functions presented in appendix A.1. Thus, 460 different problem instances are considered. The objective functions are 100-dimensional and each agent is responsible for 2 decision variables. This system size was chosen because preliminary studies showed that the effects of the CT increase with system size. In 10D experiments, only minor effects were observed, while in 100D they were much more pronounced. At the same time, the experiments can still be run in an acceptable time with this system size and the available hardware. All problems are minimization problems.

For each problem instance, the distributed fitness landscape analysis was conducted. One representative from each of the clusters determined by the correlation analysis in section 3.7 was selected as the basis for further computations, reducing the number of FLA metrics from 26 to 15. Table 6.1 shows the clusters from table 3.2 and the associated representatives. Since an arbitrary metric may be selected, the top metric has been chosen in each case.

Optimization runs were performed on all problem instances using COHDA and IDICE with the different CTVs. The parameterizations described in appendix A.2 were applied to the heuristics (*config 2* for IDICE). Table 6.2 shows the configurations for the topology variants. The setup is such that for the two adaptive modes *decrease* and *increase* a low, medium and high value has been set for each parameter. For the initial topology, $k = 50$ results in a fully meshed topology, while $k = 2$ results in a ring topology. Thus, $k = 24$ produces a moderately meshed topology. In variants with the *decrease* mode, edges are removed. Thus, only initial topologies with a sufficient number of edges and thus edge removal potential are appropriate. The same is true for the *increase* mode. In this mode, only initial topologies with sufficient potential

size	no.	metrics	remarks	representative
single feature clusters	1	v_{cv}	Coefficient of variation in variable sensitivity	v_{cv}
	2	v_{inter}	Degree of variable interaction	v_{inter}
	3	$DM\ mean$	Dispersion metric mean	$DM\ mean$
	4	$DM\ std$	Dispersion metric standard deviation	$DM\ std$
	5	$FEM_{macro}\ std$	Standard deviation of entropy based measure of ruggedness on macro scale	$FEM_{macro}\ std$
	6	$SEM_{micro}\ std$	Standard deviation of entropy based measure of smoothness on micro scale	$SEM_{micro}\ std$
	7	$\mu_2(y)\ mean$	Fitness variance mean	$\mu_2(y)\ mean$
	8	$\mu_2(y)\ std$	Fitness variance standard deviation	$\mu_2(y)\ std$
2 feature clusters	9	$FEM_{micro}\ mean$ $PIC_{micro}\ mean$	Mean values over all subsearch spaces of ruggedness and modality on micro scale	$FEM_{micro}\ mean$
	10	$FEM_{micro}\ std$ $PIC_{micro}\ std$	Standard deviations over all subsearch spaces of ruggedness and modality on micro scale	$FEM_{micro}\ std$
	11	$SEM_{macro}\ mean$ $SEM_{micro}\ mean$	Mean values over all subsearch spaces of smoothness on macro and micro scale	$SEM_{macro}\ mean$
	12	$SEM_{macro}\ std$ $PIC_{macro}\ std$	Standard deviations (as measure for heterogeneity) of smoothness and modality in macro scale	$SEM_{macro}\ std$
3 feature clusters	13	$G_{dev}\ mean$ $G_{avg}\ std$ $G_{dev}\ std$	Gradient based metrics involving standard deviations	$G_{dev}\ mean$
	14	$FEM_{macro}\ mean$ $PIC_{macro}\ mean$ $G_{avg}\ mean$	Mean values for ruggedness and modality on macro scale combined with average gradients	$FEM_{macro}\ mean$
4 feature cluster	15	$\mu_2(\|\|d\|\|)\ mean$ $\mu_2(\|\|d\|\|)\ std$ $\mu(\|\|d\|\|)\ mean$ $\mu(\|\|d\|\|)\ std$	metrics concerning state distance and variance	$\mu_2(\|\|d\|\|)\ mean$

Tab. 6.1.: Clustered FLA metrics based on Spearman correlation coefficient and the chosen representatives for each cluster

for edge addition should be used. The *static* mode does not change the topology. Here, all values for k are used that occur in the other two modes. Thus, for each adaptive setup there is a static counterpart that runs with the same initial topology parameters. The parameter p determines the number of edges that are rewired when the initial topology is created. The topology largely resembles a regular lattice for small values of p. With a large p, the topology corresponds more to a random graph. For all 3 modes the same 3 values were tested, again one low, one medium and one high. The parameter α is only relevant for the two adaptive modes. A value of $\alpha = 0.7$ is the largest reasonable value, since this results in a "one-step-schedule", meaning a schedule that contains only one step from the initial topology to the respective final state, i.e. the fully meshed or the ring topology (see section 5.2.2 for the calculation). Depending on the mode and the value for k, the topology schedule contains up to 1408 steps with $\alpha = 0.1$.

mode **parameters**	**static**	**decrease**	**increase**
k	2	-	2
	12	-	12
	24	24	24
	36	36	-
	50	50	-
p	0.1	0.1	0.1
	0.5	0.5	0.5
	0.9	0.9	0.9
α	-	0.1	0.1
	-	0.3	0.3
	-	0.7	0.7

Tab. 6.2.: Setups for CTVs

With these setups, optimization runs with 53 different CTVs were performed for each composite space (when starting with $k = 2$ or $k = 50$, the parameter p has no effect and is therefore not varied). Two different seeds for random operations were implemented, one responsible for all operations concerning the topology and one relevant for the actual optimization. In each case, 3 seed values are used, which translates to 9 optimization runs with different seed combinations for each configuration. Thus, 477 optimization runs were performed per problem instance, resulting in a total number of 219,420 runs for all problem instances. All numbers given here refer to each heuristic. Thus, 219,420 runs were performed for both COHDA and IDICE, for a total of 438,840 runs[1].

6.1.2 Measures for performance dimensions

The considered dimensions of algorithm performance are (as described in section 5.1) the achieved solution quality, the required computational effort, and the resulting communication overhead. In order to evaluate these three dimensions, the appropriate data must be collected from the optimization runs:

- **Solution quality**: value of the objective function $f(x)$ for the final x
- **Communication overhead**: total number of messages $|M|$ sent by all agents during the optimization run
- **Computational effort**: total number of local optimizations (*decide* steps) $|SE|$ of all agents during the optimization run

For the two heuristics, the local optimization effort and the number of messages are very different. For instance, IDICE has larger message sizes. On the other hand, a local optimization run is less costly than for COHDA, since only a few DE iterations are performed, whereas COHDA performs a complete run of the local SHGO optimizer. However, since the goal is not to compare the two heuristics, but only the CTVs within the heuristics, these differences are not relevant for this evaluation.

To allow comparability of performance on different problem instances, the metrics are normalized per problem instance. This is achieved by applying min-max scaling to the performance metrics for each problem instance, i.e., to the results for each composite space. This scales the values to the range [0,1]. All three performance dimensions favor smaller values, making 0 the optimum. In addition, before scaling, outliers in solution quality, number of messages, and number of local searches are detected using isolation forests [LTZ12] and then removed from the dataset[2].

This approach makes it easier to see the differences between topology variants. However, there is another important aspect that is lost when using this procedure. For some problem instances, many or even all topology variants produce very good results. In other words, compared to the total range of values, all found results are very close to each other. This aspect is eliminated by the regular min-max scaling of the results of the optimization runs. A second possibility for scaling, at least for the

---

[1] For COHDA, most of the optimization runs were even performed with 5 different seeds, i.e. 25 repetitions, but this could not be performed in all cases due to time constraints. Therefore, the data set for COHDA actually contains 523,118 data points.

[2] implementation by [Ped+11]

solution quality, is therefore to perform a maximization for each problem instance in order to determine a new upper bound of the value range. This new upper bound can then also be used for the min-max scaling. The thus calculated metric $f(x)_{norm-max}$ displays the ratio of the value range obtained in the optimization runs to the total value range.

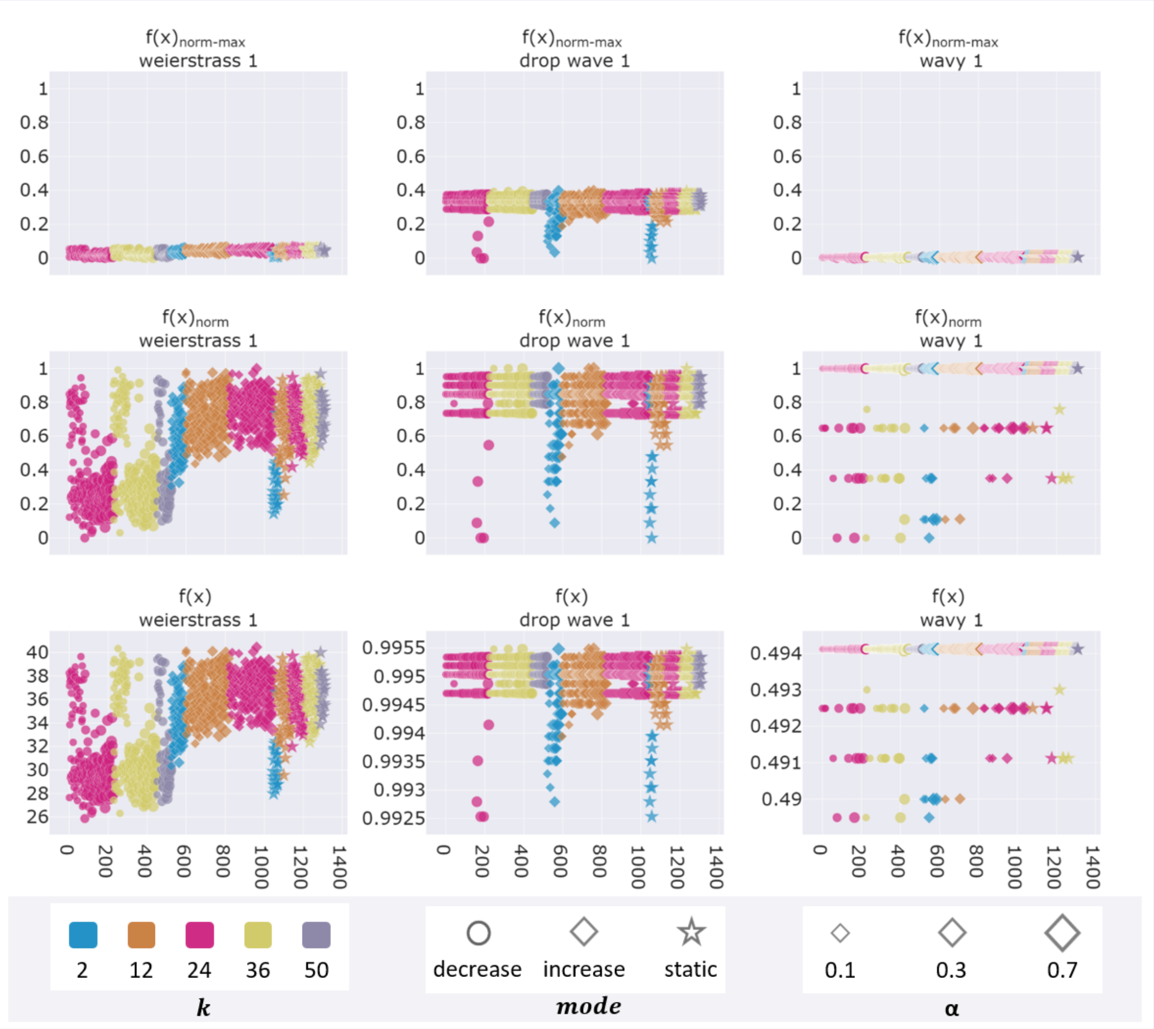


Fig. 6.2.: Results for three different problem instances (columns) with different normalization approaches (rows): $f(x)$ represents the raw result after the optimization run, $f(x)_{norm}$ is normalized with a min-max scaling based on the values obtained in the optimization runs, and $f(x)_{norm-max}$ is normalized with a larger maximum value determined by an additional maximization run.
In each figure, the CTVs are plotted along the x-axis (sorted by their label in the form $mode\_k\_p\_\alpha$, e.g. $decrease\_24\_0.1\_0.1$ on the far left and $static\_50\_0\_0$ on the far right).

For illustration purposes, fig. 6.2 shows the different normalization approaches using three examples. The bottom row shows the raw result values of $f(x)$ after the optimization runs with COHDA. The columns show the results of the first composite space based on *Weierstrass*, *Drop Wave*, and the *Wavy* function. The colors, shapes, and sizes of the markers in the scatterplots represent the different CTVs, although this figure is intended to show only a general tendency. In the middle row, the results for the three problem instances are normalized with regular min-max scaling. In the top row they are scaled with the upper bound obtained by maximization.

The *Weierstrass* problem instance in the first column exemplifies problems wherein CTVs differ in effect but yield fairly similar results relative to the total value range. Considering the original value range of $f(x)$ and the regular normalized value range $f(x)_{norm}$, it is noticeable that variants with large values for $k \in 24, 36, 50$ and the mode *decrease* perform better on average. Values with $f(x) \in [25.8682, 40.4535]$ were obtained in the optimization runs. The maximization for this problem instance resulted in $f(x)_{max} = 253.9390$. Thus, compared to the known total range of values $[25.8682, 253.9390]$, the range of values $([25.8682, 40.45355])$ found in the optimization is only about $6.4\,\%$. When normalized by $f(x)_{max}$, $f(x)_{norm-max}$ is in the range $[0.0000, 0.064]$ with a mean of 0.0363.

The problem instance based on the *Drop Wave* in the middle column shows a different picture. Here the original values lie in a seemingly small value range $f(x) \in [0.9925, 0.9955]$. Both $f(x)$ and $f(x)_{norm}$ show that especially variants with $k = 2$ and the modes *increase* and *static* lead to better results. The upper bound found by maximization was 0.999. Thus, the range of values obtained by optimization has a share of about $45.59\,\%$ of the known total value range. This is noticeable by the larger values at $f(x)_{norm-max}$ compared to *Weierstrass 1*. The mean value here is 0.3263. In combination, $f(x)_{norm-max}$ and $f(x)_{norm}$ show that the choice of CT is a relevant design decision for this problem instance, and also indicate which choices are likely to be advantageous.

As a last example, the third column shows a problem instance that is based on the *wavy* function. Using $f(x)$ and $f(x)_{norm}$, we can see that the majority of all optimization runs lead to the same local optimum. A few runs have achieved better results, although there is no tendency for any particular CTV. The original value range is similarly small as for *Drop Wave 1*: $f(x) \in [0.4895, 0.4941]$. The maximization yielded an upper bound of 1.1956. The percentage of the range of values occurring in the optimization to the total range of values is $0.66\,\%$. The mean at $f(x)_{norm-max}$ is correspondingly low at 0.0064. For this problem, the effect of exchange topologies is relatively small. In addition, there is no clear preference for any topology variant.

The presented examples illustrate that the two metrics $f(x)_{norm}$ and $f(x)_{norm-max}$ emphasize two different aspects. The metric $f(x)_{norm-max}$ indicates the extent to which topology variants influence the result. While the metric $f(x)_{norm}$ highlights differences between variants.

In the following, both metrics are used to evaluate the solution quality. Additionally, the number of messages $|M|$ is used to consider the communication overhead and the number of local searches $|SE|$ is used to consider the computational effort. Since it is not possible to determine maximum values otherwise, both are normalized only by the regular min-max scaling. Figure 6.3 shows all used performance parameters for *Weierstrass 1* and *Drop Wave 1*.

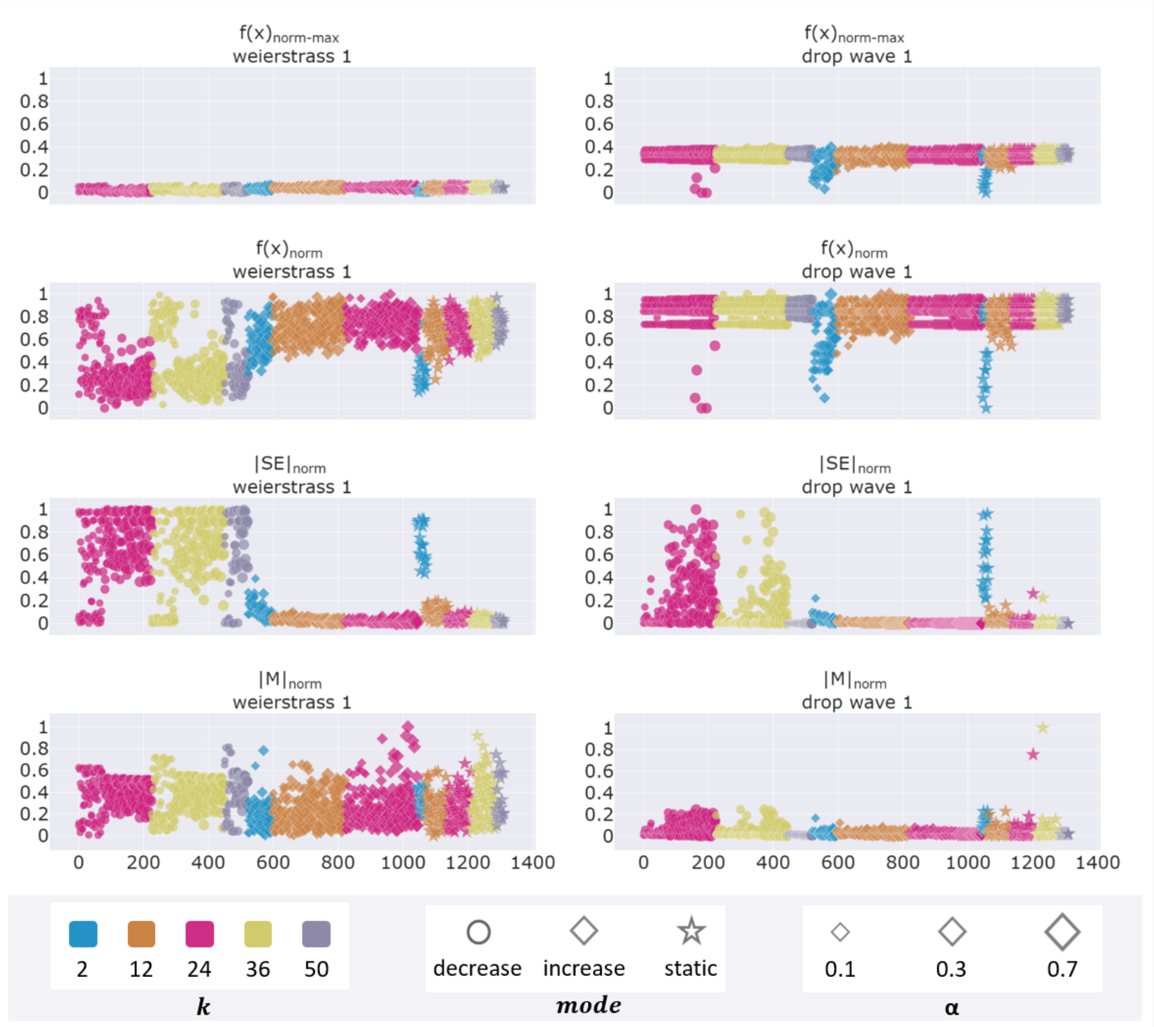


Fig. 6.3.: Examples of applied parameters for performance dimensions (rows) on two problem instances (columns); In each figure, the CTVs are plotted along the x-axis (sorted by their label in the form $mode\_k\_p\_\alpha$, e.g. $decrease\_24\_0.1\_0.1$ on the far left and $static\_50\_0\_0$ on the far right).

As noted above, $f(x)_{norm-max}$ generally indicates a greater influence of the CTV for *Drop Wave 1* than for *Weierstrass 1*. However, for both problems a clear difference between the variants can be seen in $f(x)_{norm}$. For *Weierstrass 1* the variants with large k and the mode *decrease* perform well, but also some runs with static ring topologies. For *Drop Wave 1*, on the other hand, variants with $k = 2$, i.e. also ring topologies with the modes *increase* and *static*, are advantageous. In terms of the computational effort $|SE|_{norm}$, the variants with large k as well as the static ring topologies tend to perform poorly for both problem instances. For *Weierstrass 1* these are exactly the configurations that also yielded the best solution quality. In contrast, for *Drop Wave 1*, the configurations with mode *increase* and $k = 2$ show both very low computational overhead and advantages in solution quality. Regarding the communication overhead $|M|_{norm}$, there are no clear differences between the CTVs for both problem instances.

Considering these exemplary results, it becomes clear that choosing the best topology variant for a problem instance is not trivial. For the example *Drop Wave 1*, the choice might still be relatively clear, since runs with the mode *increase* and $k = 2$ perform well in all performance dimensions. For *Weierstrass 1*, the choice would be much harder, since a tradeoff between different performance dimensions is needed. Furthermore, random factors introduce variance into the results and make it hard to make an unambiguous selection. The object of investigation in the following chapter is therefore the problem of labeling, i.e. the presumed selection of the best topology variant for each problem instance.

6.1.3 The labeling problem

As outlined above, in previous work [HN21b; HN21a], machine learning models have been trained to predict the best CTV for optimization with COHDA based on a set of FLA metrics. However, these works only compared different static topologies and some simple dynamic approaches. Therefore, the performance differences were usually large enough to clearly identify the best topology variant. In [HN21b], supergroups of topology types (e.g. highly meshed) were manually assigned when several topology types performed comparably well. Since then, the modeling of topology variants has become much more sophisticated. Therefore, it is likely that similar variants, e.g., those that differ by only one parameter, will perform similarly. A "winner-takes-all" labeling, i.e., considering only one topology as the best for a problem instance and using it as a label, is not appropriate. Consider two similar problem instances, i.e., with similar characteristics of the FLA metrics. It is likely that topology variants that perform well on one problem instance will also perform well on the other problem instance. However, the same variant may not perform best on both problem instances.

Figure 6.4 shows an example of such a case. The top two rows show the results for $f(x)_{norm}$ for the 53 different CTVs for the two problem instances *Weierstrass 1* and *Weierstrass 17* as box plots. The CTVs are encoded on the x-axis in the form $mode\_k\_p\_\alpha$. The bottom row shows the values for the 15 Fitness Landscape metrics for the two problem instances. From the FLA metrics, it is easy to see that the two problem instances are very similar. Looking at the box plots, we see that for both

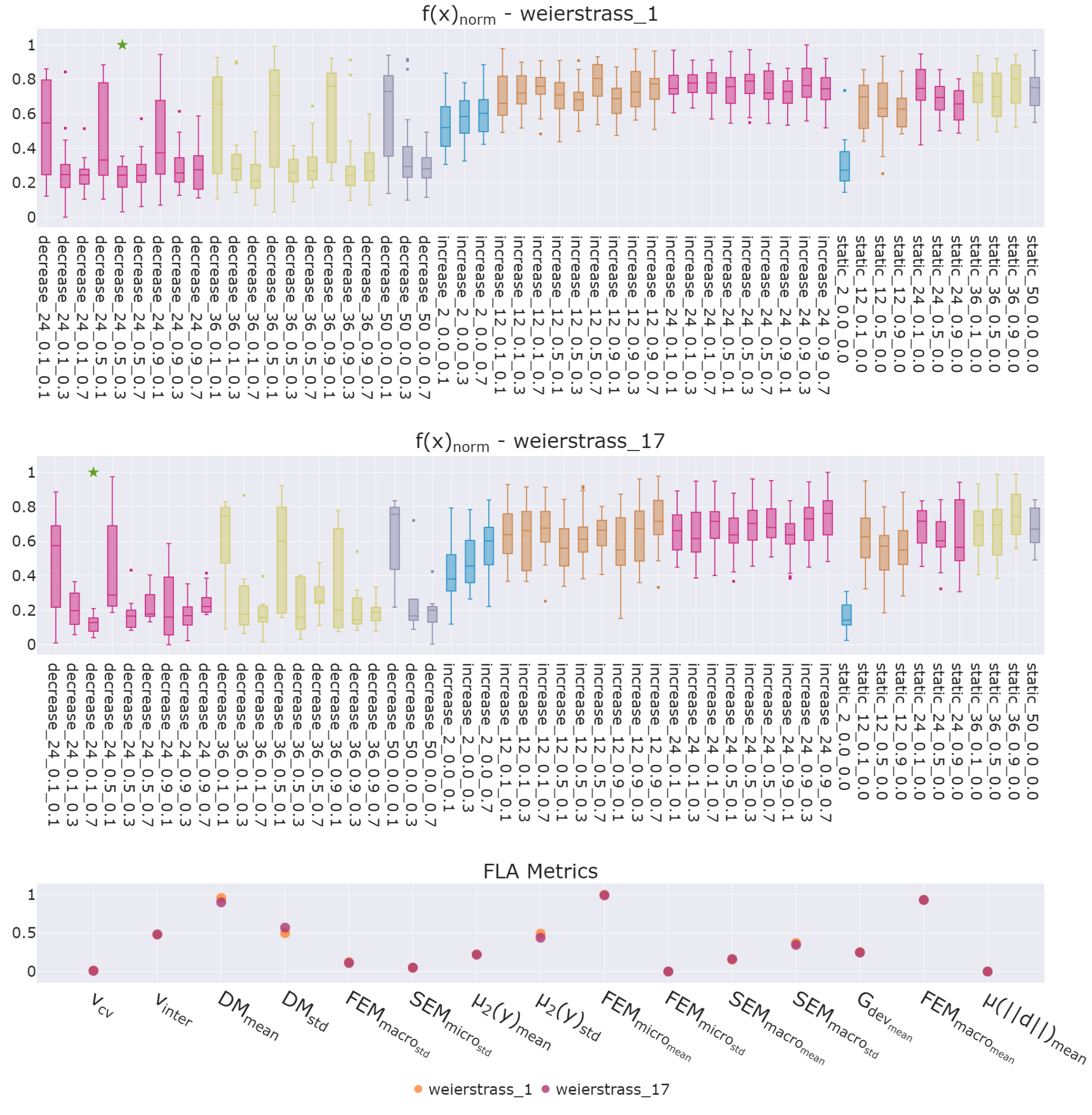


Fig. 6.4.: Comparison of the performance in terms of solution quality of all CTVs for two similar problem instances. Topology variants are encoded in the form $mode\_k\_p\_\alpha$.

problem instances, the topology variants with the mode *decrease* and $\alpha \in [0.3, 0.7]$ as well as the static ring topology ($static\_2\_0.0\_0.0$) perform well. The best run with respect to the mean value of $f(x)_{norm}$ is marked with a green star. If only these variants were set as the label for the best topology variant with respect to $f(x)_{norm}$ for the given problem instance, all other variants that also performed very well would be ignored.

Such a labeling approach would be a poor basis for ML models (such as the decision trees used in [HN21b; HN21a]), since the labels are independent classes and the models cannot generalize or interconnect them. In the example shown, manual labeling and thus defining and assigning superclasses for groups of topology variants would be an option. Clustering algorithms could potentially be used for this purpose. However, since it would still require human control, such an approach is not practical. Furthermore, the selection of the appropriate topology variant shouldn't be based solely on the solution quality. Other performance dimensions should also be taken into account with varying priorities. A weighted sum (as in [HN21a]) can be used for this purpose. However, this would require re-labeling and training a new ML model for each new weight distribution.

Given these challenges, an alternative approach seems more reasonable. Instead of using a CTV as a label and trying to learn the best variant for a given prioritization of performance dimensions, we add the topology variant parameters as input to the ML model and aim to learn the performance metrics.

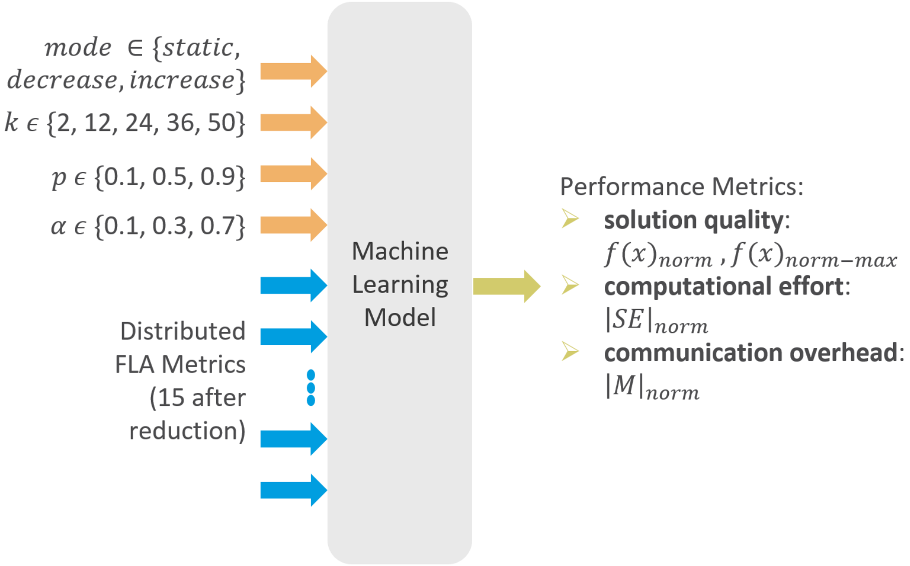

$mode \in \{static, decrease, increase\}$
$k \in \{2, 12, 24, 36, 50\}$
$p \in \{0.1, 0.5, 0.9\}$
$\alpha \in \{0.1, 0.3, 0.7\}$
Distributed FLA Metrics (15 after reduction)
Machine Learning Model
Performance Metrics:
solution quality: $f(x)_{norm}$, $f(x)_{norm-max}$
computational effort: $|SE|_{norm}$
communication overhead: $|M|_{norm}$

Fig. 6.5.: General structure of the ML model that will serve as the core of ToVarO. The FLA metrics and the CTV parameters are used as input, while the values of the performance metrics are predicted. A separate model for each performance metric or a combined model for all metrics can be trained.

In other words, the model receives as input the FLA metrics, i.e., the problem characteristics, and the topology variant parameters, i.e., $mode$, k, p, and α. The model then has to learn how to predict the performance of the optimization heuristic based on the characterization of the problem and a given topology variant. Figure 6.5 illustrates this general structure. This implies that for the envisaged automatic selection of topology variants by the controller, the trained model must be executed once for each topology variant in question. Since running trained models is fast and not computationally demanding, this should not be an issue. On the other hand, many of the above problems are solved. It is no longer necessary to select a single topology variant as a label. Thus, good results of other variants are no longer neglected. Furthermore, the predicted values for the performance parameters can be arbitrarily combined. Thus, at most one model per performance dimension is needed instead of one model per different prioritization of the performance dimensions.

After presenting the experimental setup and the basics of the envisioned ML model setting, the next step is to investigate the effects of the intended input factors on the performance dimensions. This serves to consolidate or adapt the input-output setup on a solid information base and thus to start the training with a suitable model.

6.2 Evaluation of feature importance

The importance of features reflects their relevance to the prediction of the target values. It can be used to gain insight into the data itself or into a trained model. In this thesis, the analysis of the feature importance is the first step in the investigation of the experimental results. This serves as an initial exploration of the relationship between the selected input and output parameters, as well as the first examination of the influence of the parameters of the CTVs.

A number of different techniques can be used to explore the importance of features. First, section 6.2.1 examines the general suitability of the feature set for predicting the performance metrics. The focus then shifts to the effects of individual features. In section section 6.2.2, the model-independent technique of correlation analysis is used for this purpose. Finally, in section 6.2.3, the permutation importance is analyzed. The effect of a features can be examined in combination with its interactions with other features, although this is model-dependent. Furthermore, the combined effect of features can also be analyzed, which is particularly interesting for the parameters of the CTVs.

6.2.1 Coefficients of determination

In statistics, the coefficient of determination (R^2) is a metric that can be used to assess the quality of the fit of a regression. In a regression model, it represents the proportion of the variance in the output variable that can be predicted using only the input variables, i.e., how well the output can be predicted based on the input. The error between the predicted targets $\hat{y}$ and actual target values y is calculated and compared to the error that would result from a continuous prediction of the average target value [Ped+11]:

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

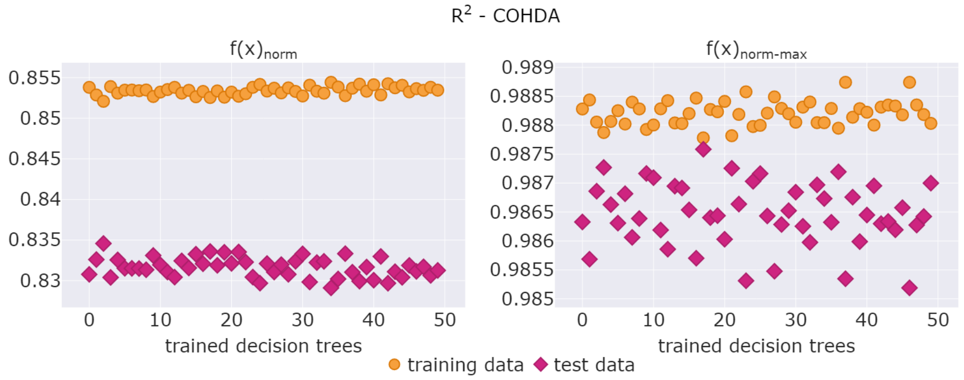
where $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$.

The best possible score is 1.0, which would indicate perfect prediction. A R^2 score of 0.0 would indicate that the prediction is as good as always predicting the mean $\bar{y}$, and a negative score is worse than always predicting $\bar{y}$. In this section, the R^2 score is used to investigate the general suitability of the setup of input and output variables presented in fig. 6.5. For each performance metric and both optimization heuristics, 50 different splits between training (70 %) and test data (30%) were performed and a decision tree regressor[3] subsequently trained. An equal distribution of problem instances was ensured when dividing the data into training and test data. The figures 6.6 and 6.7 show the resulting R^2 scores on the training and test data. Here, fig. 6.6 shows the scores for the two metrics of solution quality, once for COHDA (fig. 6.6a) and once for IDICE (fig. 6.6a). As expected, the scores are higher for the training data. For $f(x)_{norm}$ and COHDA, the scores range from 0.852 to 0.854 for the training data and from 0.829 to 0.835 for the test data. For $f(x)_{norm-max}$ the values are even higher, ranging from 0.9878 to 0.9887 for the training data and from 0.985 to 0.988 for the test data. The higher scores for $f(x)_{norm-max}$ can be explained by the fact that the values, although scaled to the range between $[0, 1]$, tend to be very small, but include a few prominent exceptions. The mean of the training and test data is about 0.068 for $f(x)_{norm-max}$, while it is about 0.34 for $f(x)_{norm}$. Thus, the prediction significantly outperforms $\bar{y}$ if a model is able to distinguish between the problem instances that have very small values and the few that have much larger values, presumably using the FLA metrics. Figure 6.6 shows a similar picture for IDICE, but the scores, especially for $f(x)_{norm}$ are considerably lower, ranging from 0.661 to 0.673 for the test data. In general, both the mean and the standard deviation for both metrics are smaller in the optimization runs with IDICE, even though the problem instances and CTVs are the same. Accordingly, IDICE seems to have less variance in solution quality. The results for both heuristics

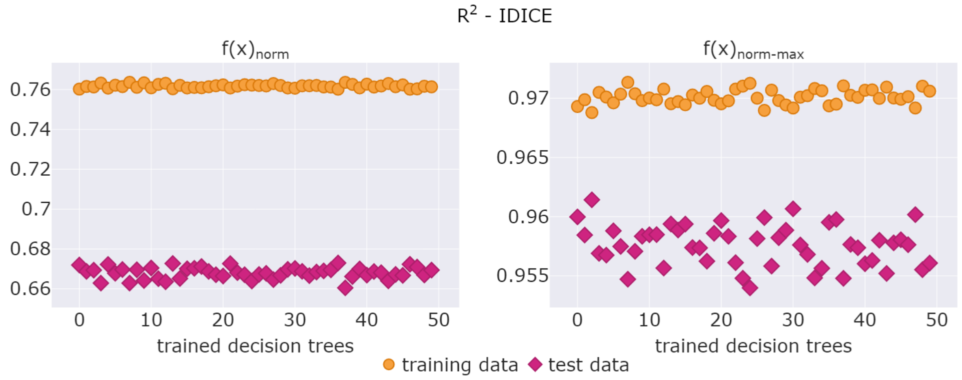
---

[3] Implementation by [Ped+11]

were normalized separately to examine differences between the CTVs within the heuristics, not differences between the two heuristics. Therefore, no conclusion can be drawn as to whether the results of IDICE are worse or better on average (although joint normalization shows that IDICE runs are better on average). However, they are comparatively consistent, and therefore the coupling between the variance of the input and output data seems to be lower with IDICE.



(a) R^2 for COHDA

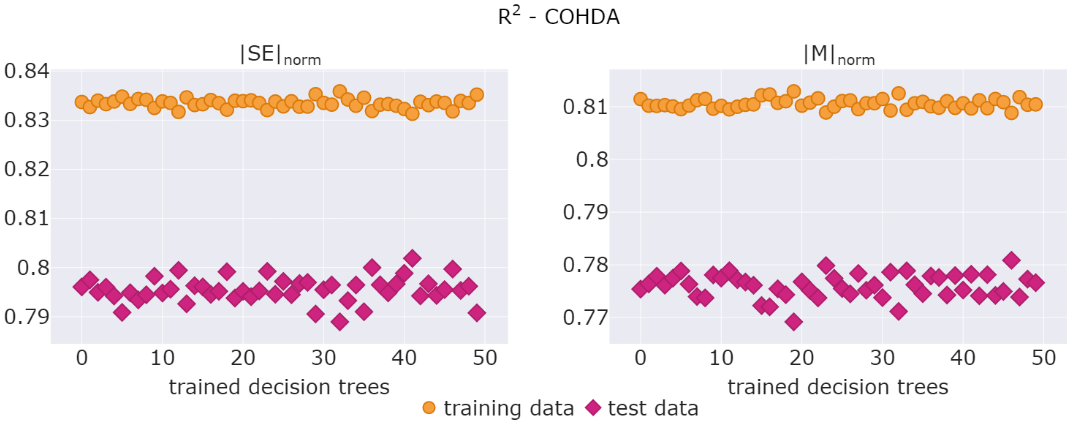


(b) R^2 for IDICE

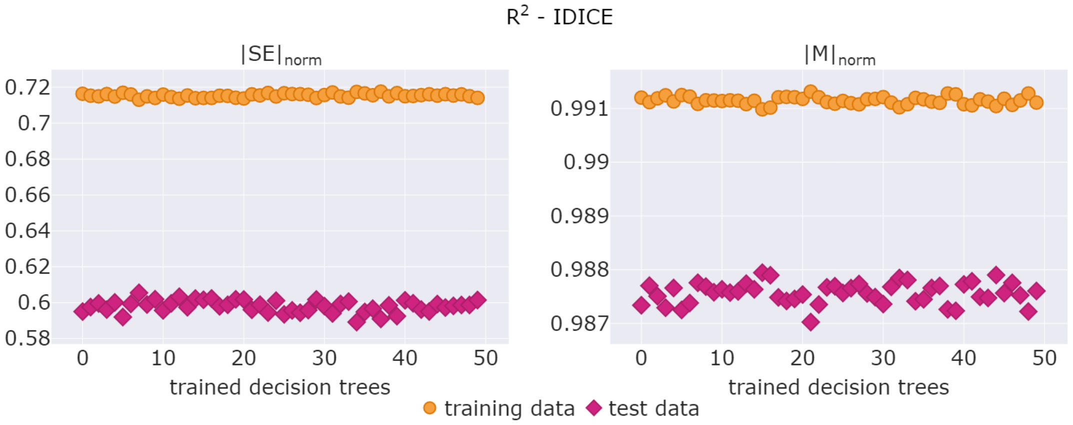
Fig. 6.6.: Determination coefficients (R^2): proportion of the variation of $f(x)_{norm}$ and $f(x)_{norm-max}$ that is predictable from FLA metrics and CT parameters according to 50 trained decision tree regressors

Figure 6.7 shows the results for both heuristics for the $|SE|_{norm}$ and $|M|_{norm}$. For COHDA and $|SE|_{norm}$, the R^2 score is about 0.83 on the training data and 0.79 on the test data. For $|M|_{norm}$ the score is about 0.81 on the training data and about 0.77 on the test data. These results all indicate reasonable accuracy of the trained models, but still a significant amount of variance in the performance metrics that the models could not predict based on the input parameters. Interestingly, the scores for IDICE and $|M|_{norm}$ are much higher, always above 0.98. This indicates that the number of messages exchanged in IDICE is closely related to one or more of the input parameters. For $|SE|_{norm}$, however, predicting IDICE seems to be more difficult. The R^2 value on the test data is only about 0.6.

Overall, the R2 scores show promising results. They indicate that for both optimization heuristics, at least a large portion of the variance in the performance metrics can be easily predicted with simple models, such as decision tree regressors, based solely on the FLA metrics and the parameters of the CTVs. As a whole, the feature set is generally appropriate. The next steps focus on individual features. This allows you to gain insight into the impact of each of these features, as well as identify features that may be unnecessary or misleading.



(a) R^2 for COHDA



(b) R^2 for IDICE

Fig. 6.7.: Determination coefficient (R^2): proportion of the variation of $|SE|_{norm}$ and $|M|_{norm}$ that is predictable from FLA metrics and CT parameters according to 50 trained decision tree regressor

6.2.2 Correlation Analysis

The correlation of individual features with performance metrics can provide an indication of how relevant they are for predicting these performance metrics regardless of a specific model. Spearman's rank correlation coefficient is a nonparametric measure of rank correlation, assessing how well a monotonic function describes the relationship between two variables. The correlation is 1 for maximum positive correlation, -1 for maximum negative correlation, and 0 for no correlation.

The full dataset for COHDA or IDICE was used to compute the coefficients (after removing the top 1% combined outliers for the performance metrics). For each CTV, the mean value of the respective performance metric was calculated for each problem instance. Then, for each topology variant across all problem instances, the correlation coefficients per FLA metric were calculated. Thus, the data series used for the correlation analysis each had 460 data points (number of problem instances).

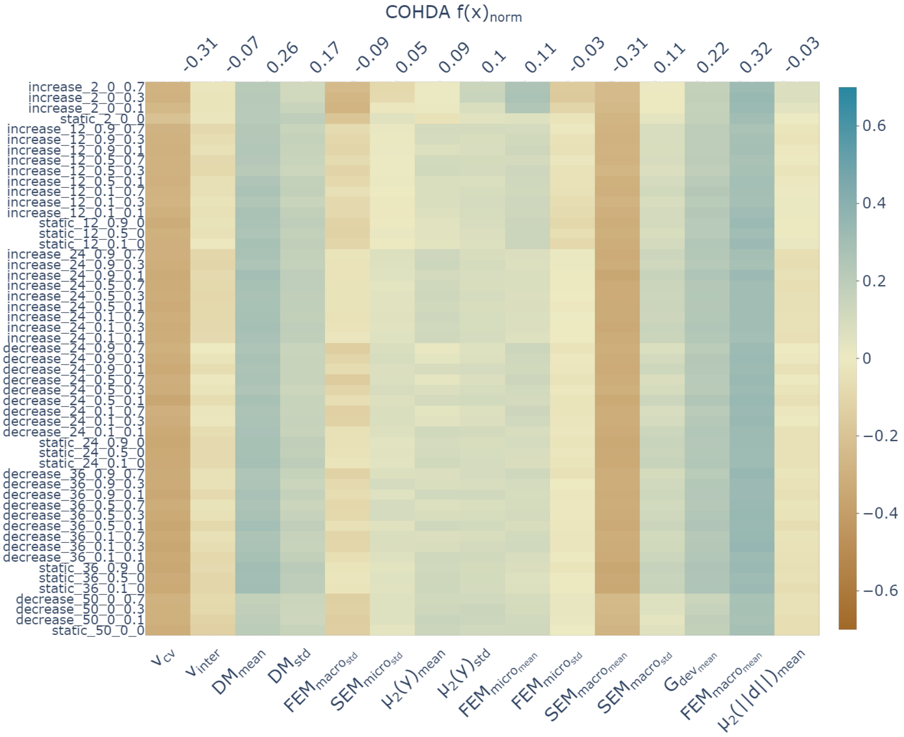
Several guiding questions are of particular relevance in the following discussion of the results, which were visualized in the form of heat maps:

1. **General impression**: What is the value range of the correlation coefficients? Are the correlation values of the FLA feature fairly similar or are there large differences?
2. **FLA correlation values**: Which FLA metrics show the highest correlation and which the lowest? Is the direction of the correlation consistent with expectations for FLA metrics?
3. **Effect of CTVs**: Does the correlation vary depending on the variant of the CT that was used? If so, is there a trend with respect to certain parameters of the topology variants?

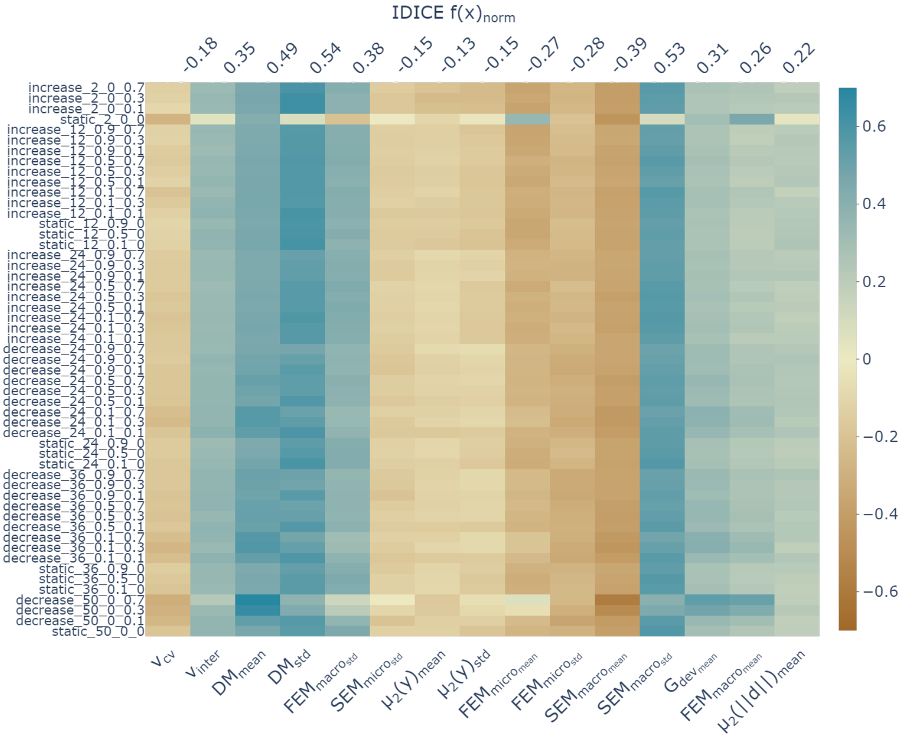
Solution quality

Figure 6.8 shows the results for the performance metric $f(x)_{norm}$ for COHDA (fig. 6.8a) and IDICE (fig. 6.8b). The heat map plots the correlation coefficients for each FLA metric with $f(x)_{norm}$ for each CTV. For each FLA metric, the average across all topology variants is shown at the top of the column. Note that these FLA metrics are representatives for their metric cluster as defined in table 6.1.

The general impression is: Most metrics show a low or moderate correlation. For COHDA, it is not above or below 0.4 and -0.4, respectively. For IDICE, the values range from -0.59 to 0.70. The correlation values are also higher on average, for IDICE. However, there are a few metrics that show a relatively small negative correlation for IDICE, while showing a (very) small positive correlation for COHDA. These are mainly $\mu_2(y)_{mean}$, $\mu_2(y)_{std}$, and $SEM_{micro_{std}}$. These metrics relate to the mean and standard deviation of the fitness variance and the heterogeneity across all subsearch spaces in terms of smoothness at micro scale. They are likely to have little significance for predicting $f(x)_{norm}$.



(a) Spearman coefficients with $f(x)_{norm}$ for COHDA



(b) Spearman coefficients with $f(x)_{norm}$ for IDICE

Fig. 6.8.: Spearman correlation coefficients for the FLA metrics with $f(x)_{norm}$ computed separately for each CTV and across all problem instances.

In the appendix, the FLA metrics with the highest positive and negative correlations are considered in more detail (see tables C.2 and C.1). The positive correlation of DM_{mean} and $FEM_{macro_{mean}}$ is consistent with expectations, as it indicates increased difficulty for optimization problems with multi-funnel shapes, higher ruggedness, modality, and average gradients. Interestingly, IDICE also indicates increased difficulty of optimization problems related to heterogeneity of subsearch spaces in terms of smoothness and global landscape structure.

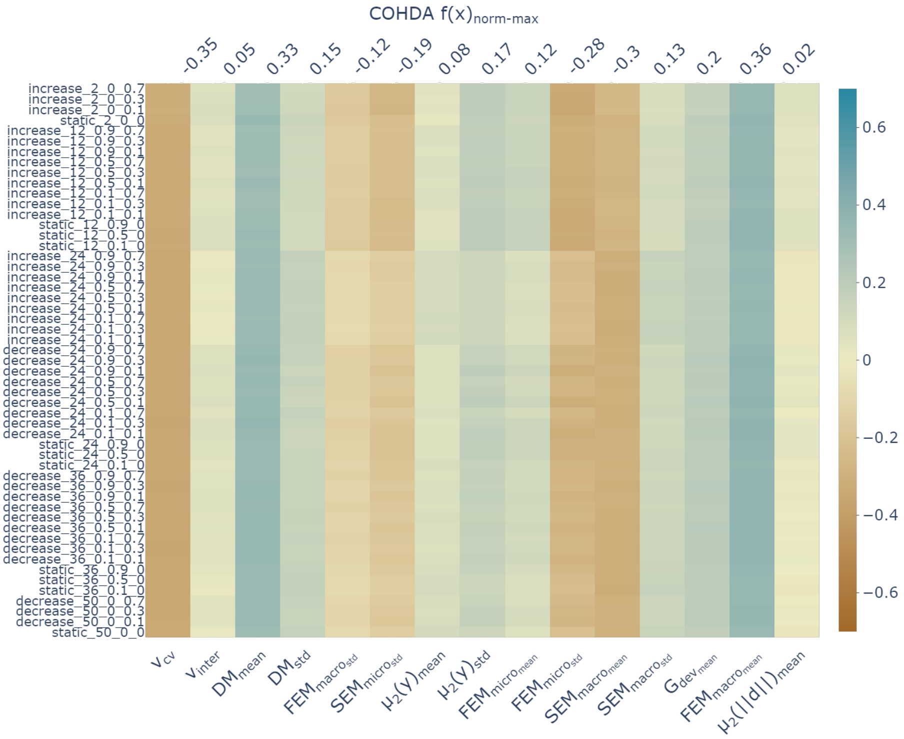
The correlation coefficients are quite similar for most CTVs. However, many metrics show discrepant values for the static ring topology ($static\_2\_0\_0$), especially for IDICE, but also for COHDA. Often the correlation coefficients are much lower than for the other topologies. For some, the direction of the correlation is even reversed. Thus, this topology seems to reduce the correlation between the problem characteristics and the $f(x)_{norm}$.

Figure 6.9 shows the results for the performance metric $f(x)_{norm-max}$ for COHDA (fig. 6.9a) and IDICE (fig. 6.9b). Compared to $f(x)_{norm}$, the patterns of the two heuristics are more similar. Again, there is generally a moderate or low correlation of up to -0.53 and 0.55 for IDICE and -0.35 and 0.37 for COHDA. The FLA metrics with the highest positive and negative correlations are the same as for $f(x)_{norm}$. Among the top metrics with negative correlation, only $FEM_{micro_{std}}$ is added. Thus, lower solution quality $f(x)_{norm-max}$ is correlated with lower heterogeneity in terms of ruggedness and modality on micro scale. Likewise, higher solution quality correlates with higher heterogeneity. The differences between the CTVs are very small. Again, the static ring topology ($static\_2\_0\_0$) shows slightly different results, but the deviation is much smaller than for $f(x)_{norm}$. In general, the differences between the topologies are marginal.

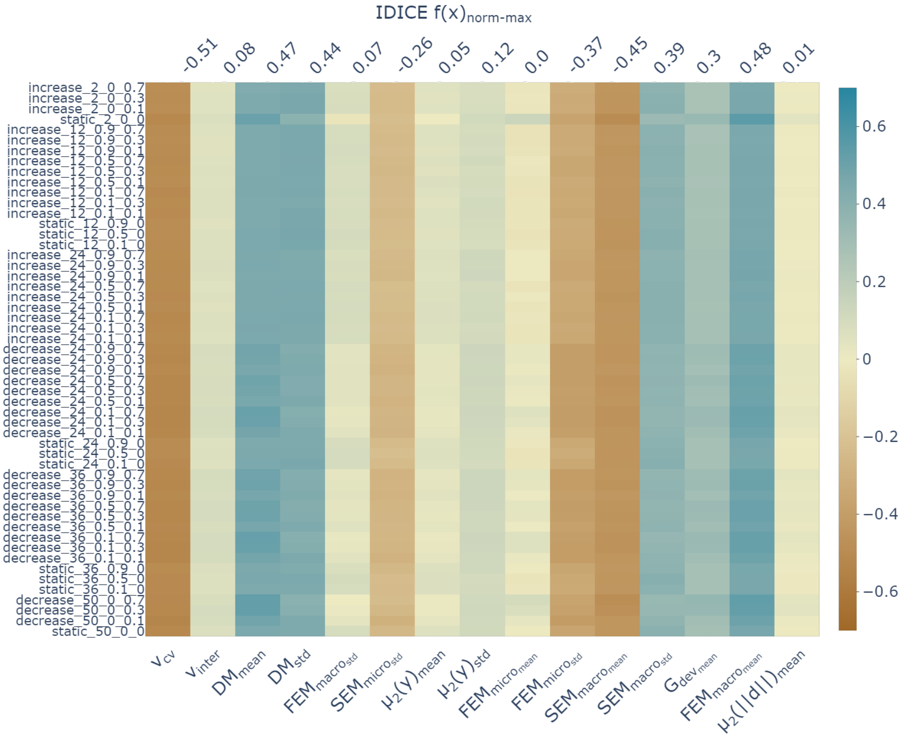
Computational effort

Figure 6.10 shows the results for the performance metric $|SE|_{norm}$ for COHDA (fig. 6.10a) and IDICE (fig. 6.10b). Again, there is a moderate or low correlation between -0.53 and 0.41. This performance metric shows clear differences between the CTVs. Furthermore, the results for the two heuristics are completely different. It is not surprising that there are few parallels here, since the metric $|SE|_{norm}$ measures the computational effort in terms of local search executions in the *decide* step of the heuristics, and these are processed and embedded differently in the heuristics.

For both heuristics, the topology variants with the mode *decrease* have the highest correlation coefficients for the FLA metrics. A possible explanation could be: In this mode, there is initially quite high exchange between the agents, but after a while the



(a) Spearman coefficients with $f(x)_{norm-max}$ for COHDA



(b) Spearman coefficients with $f(x)_{norm-max}$ for IDICE

Fig. 6.9.: Spearman correlation coefficients for the FLA metrics with $f(x)_{norm-max}$ computed separately for each CTV and across all problem instances.

topology transforms into a ring. From the perspective of the individual agents, they then enter an exploitation phase in which they receive little input from other agents and thus only investigate slightly different variants of their own search space (cf. fig. 5.4). Accordingly, properties of the fitness landscape that speed up or slow down the search could have a stronger effect, since in the first case the agents quickly find a favorable region of the overall search space, exploit it individually, and terminate quickly. In the case of unfavorable properties, however, the search can take a very long time, since small improvements need a long time to propagate throughout the system when communicating over a ring topology. It is also noteworthy that several FLA metrics are correlated with opposite signs for COHDA and IDICE (e.g., v_{cv}, $\mu_2(||d||)_{mean}$, $FEM_{macro_{mean}}$). This is probably due to the peculiarities of the two heuristics.

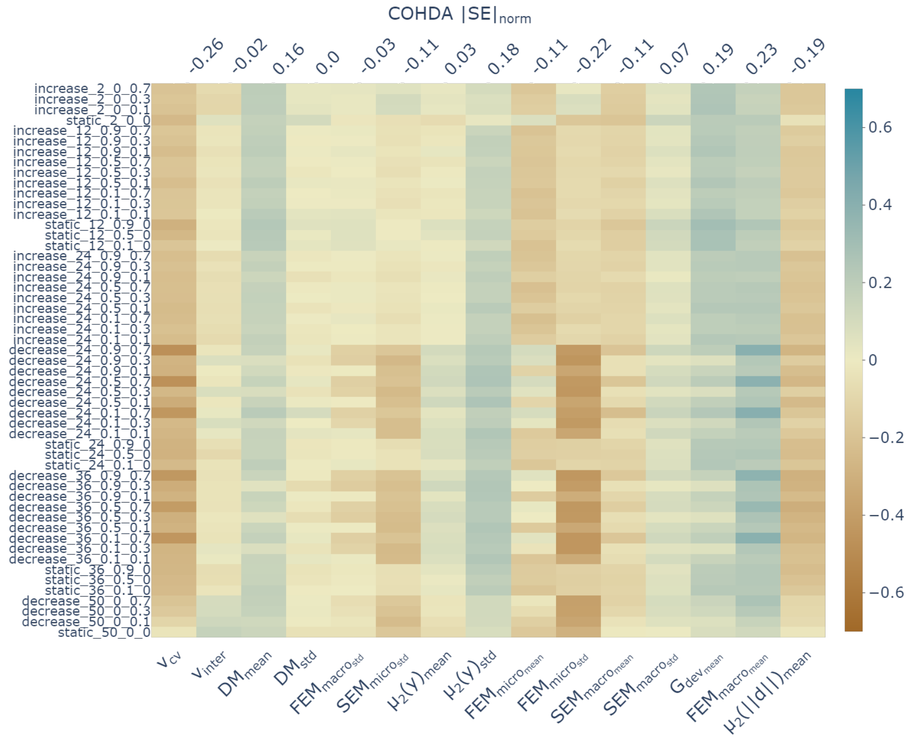
For $|SE|_{norm}$, we can summarize that different FLA metrics have a moderate or low positive or negative impact on the computational effort. It depends on both the heuristic and the CTV which metrics correlate positively and which negatively. For both heuristics, the correlation coefficients are often higher for topology variants with the mode *decrease*.

Communication overhead

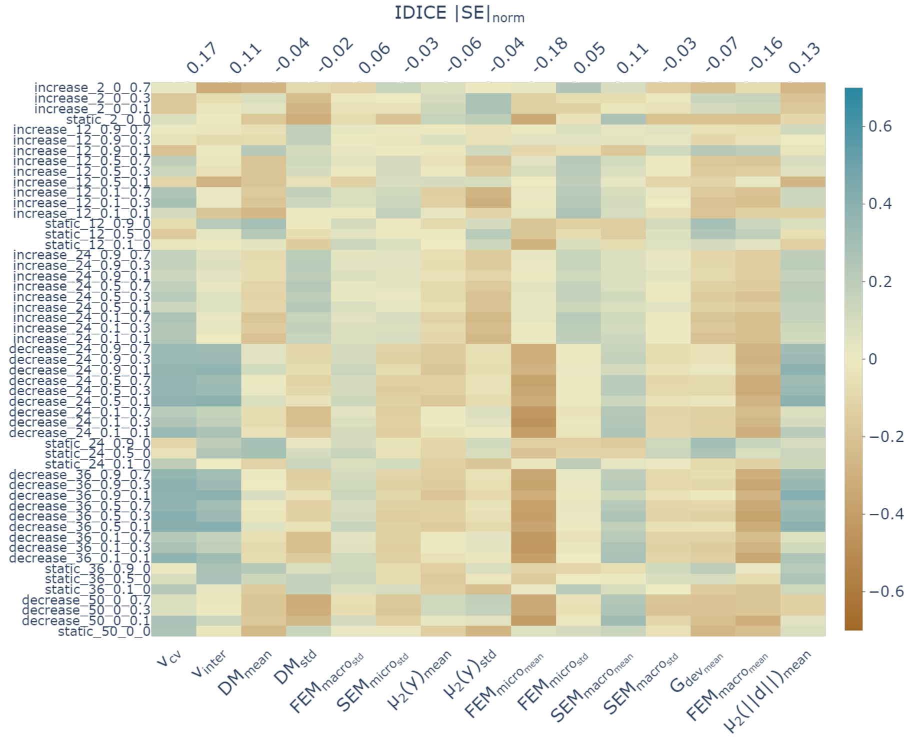
Figure 6.11 shows the results for the performance metric $|M|_{norm}$ for COHDA (fig. 6.11a) and IDICE (fig. 6.11b). The values of the correlation coefficients are relatively low, almost negligible, often in the range [-0.1, 0.1]. A very different pattern emerges for the two heuristics. For IDICE, the correlation coefficients differ mostly depending on the mode of the topology variant. For example, for v_{cv} there is a negative correlation for all topology variants with the modes *increase* and *static* (except for the static ring topology), and a positive correlation for all topology variants with the mode *decrease*. In both cases, however, the correlation coefficients are very small. For COHDA, there are no large differences between modes. However, for several FLA metrics the correlation becomes stronger or weaker depending on the value of α. But again, the correlation coefficients are quite small. This suggests that the expected communication overhead depends primarily on the CTV and is only slightly affected by the properties of the fitness landscape.

Statistical significance and summary

An important part of the correlation analysis is to test the statistical significance of the results. For each of the examined FLA metrics (x) and performance metrics (y), the following null hypothesis is stated:

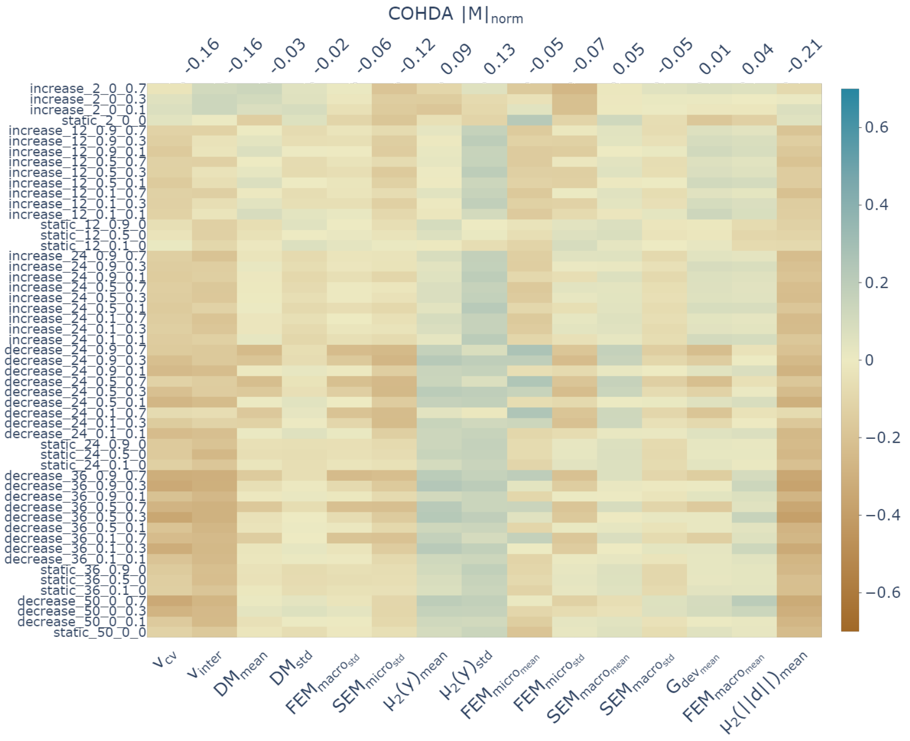


(a) Spearman coefficients with $|SE|_{norm}$ for COHDA

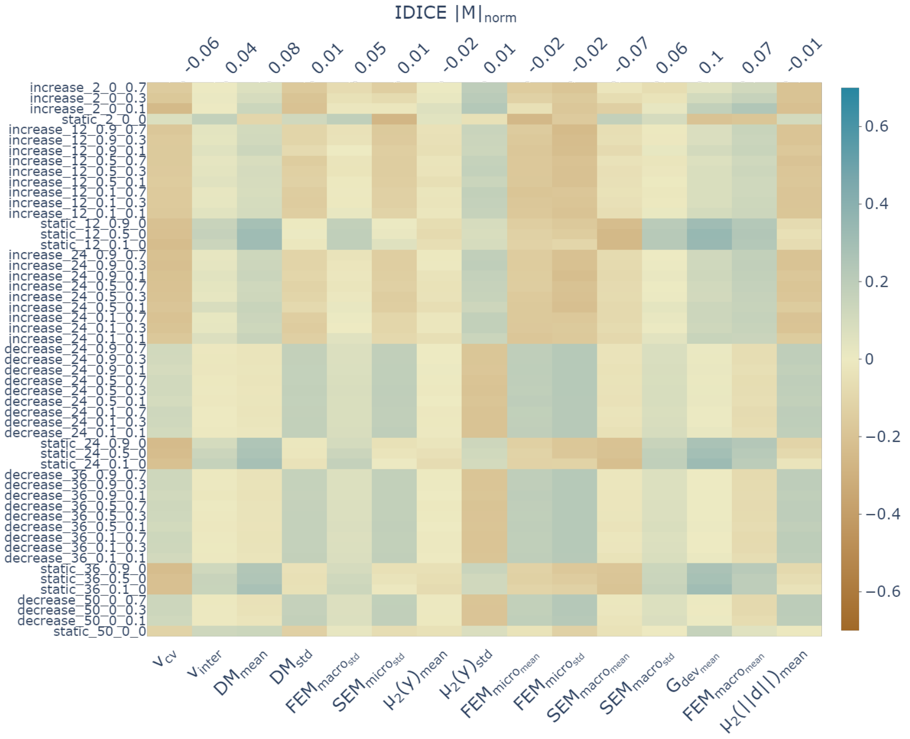


(b) Spearman coefficients with $|SE|_{norm}$ for IDICE

Fig. 6.10.: Spearman correlation coefficients for the FLA metrics with $|SE|_{norm}$ computed separately for each CTV and across all problem instances.



(a) Spearman coefficients with $|M|_{norm}$ for COHDA

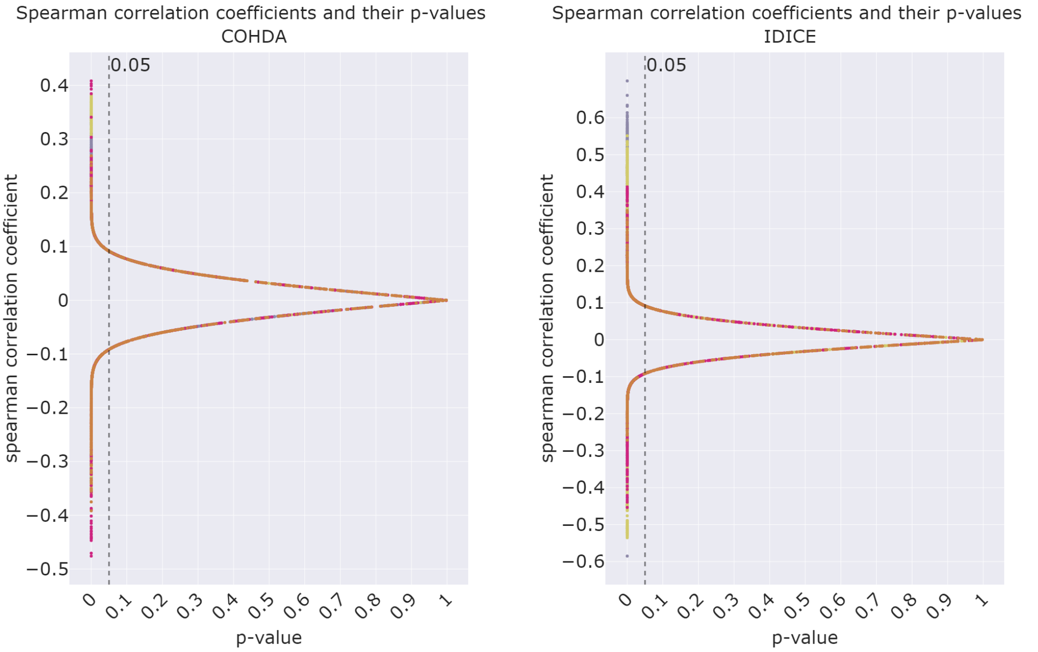


(b) Spearman coefficients with $|M|_{norm}$ for IDICE

Fig. 6.11.: Spearman correlation coefficients for the FLA metrics with $|M|_{norm}$ computed separately for each CTV and across all problem instances.

Null Hypothesis. *There is no significant correlation between the FLA Metric x and the performance metric y.*

The *p-value* indicates the probability with which the respective null hypothesis can be rejected, and thus the probability of obtaining results that are as extreme or more extreme than the calculated correlation coefficients, given that the null hypothesis is true. Since the same setup was used to compute the correlation for each combination of FLA and performance metrics, e.g., number of data points and same underlying problem instances, the absolute magnitude of the correlation coefficients is the main factor determining how likely such values are to occur at random. Figure 6.12 shows the plot of the correlation coefficients versus their corresponding p-values for COHDA (fig. 6.12a) and IDICE (fig. 6.12b). The vertical dashed line shows the p-value of 0.05, which is commonly used as a threshold for statistical significance. For both COHDA and IDICE, all correlation coefficients with an absolute value of approximately 0.091 are above this threshold. For the larger correlation coefficients, the p-values quickly become very small (up to 10^{-69}). Thus, for the correlations between FLA and performance metrics explicitly discussed in the previous sections, statistical significance was found. Even for $|SE|_{norm}$ and $|M|_{norm}$, which generally show very low coefficients, the higher correlation coefficients discussed for several CTVs are in the statistically significant range. In appendix C.1, the p-values are shown in separate heatmaps, analogous to the heatmaps for the correlation coefficients shown earlier.



(a) Spearman coefficients relative to the corresponding p-values for COHDA

(b) Spearman coefficients relative to the corresponding p-values for IDICE

Fig. 6.12.: Spearman coefficients relative to the corresponding p-values

In summary, the analysis of the correlation coefficients yields the following picture: Overall, the magnitude of the coefficients is moderate or low, being almost negligible for $|SE|_{norm}$ and $|M|_{norm}$. Regarding solution quality, there were the expected correlations between problem difficulty and the FLA metrics representing multimodality, ruggedness, gradient height, and multi-funnel shapes. In addition, heterogeneity of subsearch spaces with respect to some landscape features (e.g., global shape and macro smoothness) tends to be associated with increasing problem difficulty, but may also be correlated with better solution quality with respect to other features (e.g., micro-level ruggedness). Regarding the correlation coefficients with respect to solution quality, the results for both heuristics are quite similar, but for $|SE|_{norm}$ and $|M|_{norm}$ they are completely different.

The CTVs show only small differences in solution quality related performance metrics. This is different for the computational overhead $|SE|_{norm}$ and the communication overhead $|M|_{norm}$. The FLA metrics are much less relevant here. Instead, parameters of the CT such as the mode or speed of topology adaptation at runtime α are of interest. Therefore, it is expected that the FLA metrics are most relevant for training prediction models with respect to solution quality, while the relevance of the CT parameters increases for the other performance dimensions. In addition, the importance of the features is expected to be different for the two heuristics.

However, the aspects that can be examined in the performed correlation analysis are limited. Only monotonic 1:1 relationships between features can be analyzed. Therefore, in the next section, the impact of features and their interactions with other features will be examined using permutation importance.

6.2.3 Permutation importance

Permutation importance is a technique for analyzing the relevance of input variables for the prediction of trained machine learning models. This technique was developed by Breiman [Bre01] with the introduction of random forests to explore the random forest mechanism.

The procedure for calculating the permutation importance is based on the implementation in [Ped+11] and follows these steps::

1. A model m is trained on the training data that contains data for the input features IF (in this case the FLA metrics and the CT parameters) and the output features OF (here one of the performance metrics)

2. The test data D is used to compute a base line score s_{base} for the model, in this case the $MSE = \frac{1}{n_{samples}} \sum_{j=1}^{n_{samples}} (y_j - \hat{y}_j)^2$, with y being the actual target values and $\hat{y}$ the predicted target values of the output feature from OF
3. For each input variable i in IF:
    - for each repetition $r \in 1, 2, \dots, R$
        - the data column of the input feature i is randomly shuffled to generate a corrupted version of the data $\widetilde{D}_{i,r}$
        - the score $s_{i,r}$ is computed for the model m on the corrupted data $\widetilde{D}_{i,r}$
    - the permutation importance pi_i of the input variable i is defined as: $pi_i = s_{base} - \frac{1}{R} \sum_{r=1}^{R} s_{i,r}$

Shuffling the column preserves statistical features such as the mean and variance of the input variable, but breaks the relationship between the feature and the target. Therefore, the decrease in the model score is an indication of the model's dependence on the feature. To increase the reliability of the results, the shuffling and subsequent calculation of the score on corrupted data is repeated R times. The permutation importance is a measure of how important a feature is to a particular model, rather than an objective measure of the predictive power of a feature. Accordingly, permutation importance only provides valuable information if the underlying model is sufficiently good.

A potential drawback of permutation feature importance is that correlated features may produce misleading values, since the shuffling of one feature may be compensated by another correlated feature that remains untouched, thus resulting in an underestimation of the importance of the first feature. Since the FLA metrics have already been subjected to correlation analysis and only one representative from each of the identified clusters is included, this is unlikely to be a problem here. The CT parameters cannot be correlated in this sense because they are not measured or calculated, but explicitly specified. Correlation exists where certain values for the parameters occur only in combination, such as the mode and specific values of α or k.

The permutation importance can even be modified in a very useful way to improve the analysis of the influence of the CT parameters. The impact of the CT parameters on the performance measures are expected to be most significant when they are considered together. In fact, it is possible to shuffle several input variables at the same time to calculate common importance values. This is exactly what was done for different combinations of CT parameters.

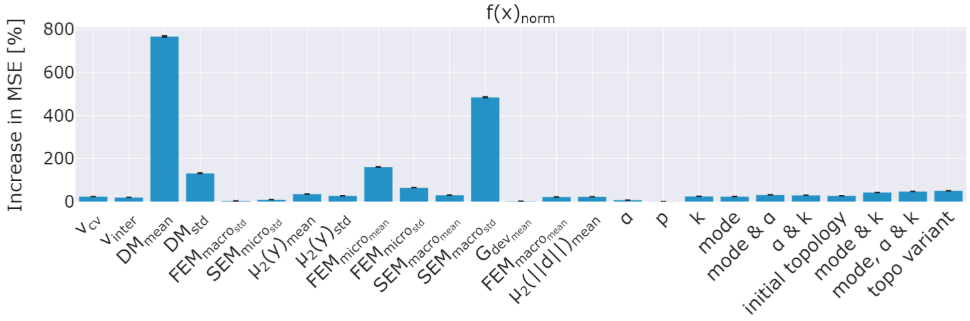
A random forest regressor with 100 trees was used to compute the importance of each feature. This model is suitable considering that the decision tree regressors used in section 6.2.1 for the investigation of the coefficients of determination already showed promising values for the R^2 score and test runs with random forest regressors showed low MSE values. The results for the three performance dimensions are analyzed separately below. The results are also summarized in tables 6.3 and 6.4.

Solution quality

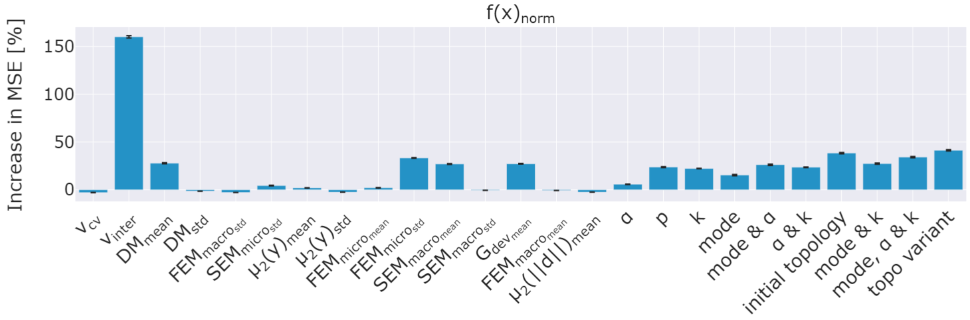
Figure 6.13 show the results for COHDA and IDICE, respectively. The captions of the sub-figures also provide the corresponding reference value for the MSE. The y-axis shows the percentage change in this reference score. For example, in fig. 6.13a we can see that the reference MSE for $f(x)_{norm}$ increased by an average of 768% from 0.0211 to about 0.18 due to the permutation of DM_{mean}. The black error bars show the standard deviation of the permutation importances computed for $R = 3$ times. However, these are so small that they are hardly noticeable.

Figures 6.13a and 6.13b show the results for $f(x)_{norm}$ for COHDA and IDICE. For both heuristics, several FLA metrics clearly stand out, while the influence of the CT parameters is rather small to moderate. Which FLA metrics are most important for the predictions differs significantly between the two optimization heuristics. For COHDA, DM_{mean} (+768%), DM_{std} (+132%), $FEM_{micro_{mean}}$ (+161%), and $SEM_{macro_{std}}$ (+486%) are the most important. These are the mean and standard deviation of the dispersion metric, i.e. the global shape, the ruggedness at the micro level, and the heterogeneity of the subsearch spaces with respect to smoothness at the macro level. These are the same FLA metrics that had the highest coefficients in the correlation analysis, except for $FEM_{macro_{mean}}$. This could be due to the fact that DM_{mean} correlates to some extent with $FEM_{macro_{mean}}$, respectively with the cluster it represents (see section 3.7). In contrast, the most relevant variable for predicting $f(x)_{norm}$ for IDICE is v_{inter}, i.e. the degree of variable interaction. For IDICE, some features have negative scores, meaning that the MSE has decreased with their permutation, including DM_{std} (-1. 4%), $FEM_{macro_{std}}$ (-2.7%), $\mu_2(y)_{std}$ (-2.4%), $SEM_{macro_{std}}$ (-0.5 %), and $FEM_{macro_{mean}}$ (-0.7 %). However, these negative values are very small, suggesting that the features are irrelevant to IDICE rather than actually impairing the prediction of $f(x)_{norm}$. For both optimization heuristics, the parameters of the CT are of similar importance. In COHDA, the CT (*topo variant*) reaches +51 %, with mode and k having the largest contribution. For IDICE, the influence is similarly high (+41 %). Here, however, p and k, which together form the initial topology, are most important.

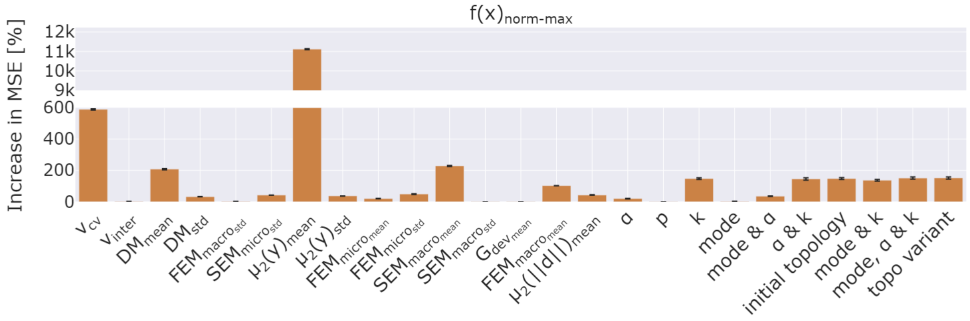
The figures 6.13c and 6.13d show the results for $f(x)_{norm-max}$. For both heuristics, the FLA metric $\mu_2(y)_{mean}$ is outstanding with + 11,118 % for COHDA and + 2,680 % for IDICE. This is the mean of the fitness variance, which can also be considered as a measure of ruggedness. The y-axis has been broken for $f(x)_{norm-max}$ to make



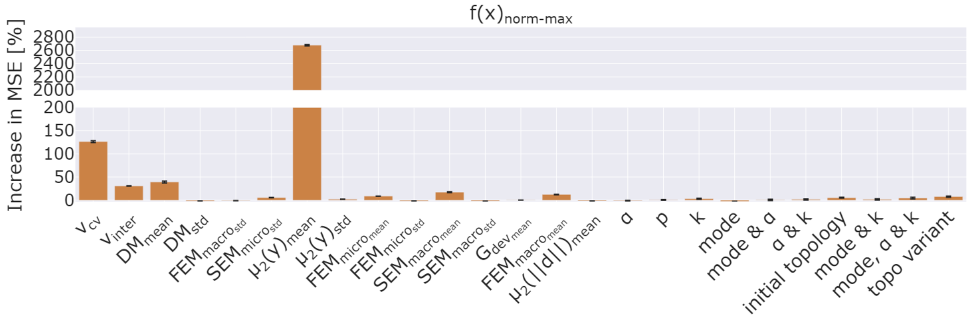
(a) Permutation importances $f(x)_{norm}$ for COHDA; base line MSE: 0.0211



(b) Permutation importances $f(x)_{norm}$ for IDICE; base line MSE: 0.0259



(c) Permutation importances $f(x)_{norm-max}$ for COHDA; base line MSE: 0.0005



(d) Permutation importances $f(x)_{norm-max}$ for IDICE; base line MSE: 0.0012

Fig. 6.13.: Permutation importances with respect to MSE for $f(x)_{norm}$ and $f(x)_{norm-max}$; The CT parameters are considered individually as well as in combination. The *initial topology* includes k and p, the *topo variant* considers all topology parameters.

it easier to read. One explanation for the extremely high values of $\mu_2(y)_{mean}$ for COHDA is that the baseline MSE for COHDA is much lower than for IDICE. The permutation of $\mu_2(y)_{mean}$ increases the MSE from 0.0005 to 0.0566 for COHDA and from 0.0012 to 0.0332 for IDICE. The second most important feature for both heuristics is v_{cv}, i.e. the coefficient of variation of the variable sensitivity, which indicates how evenly the influence of individual variables on the result of the optimization is distributed. Relative to this, most of the other features have low importance, although this time in COHDA the parameters of the CT have a greater influence, and various combinations of topology parameters show an importance score around +140%, with the main influence coming from k (+148%).

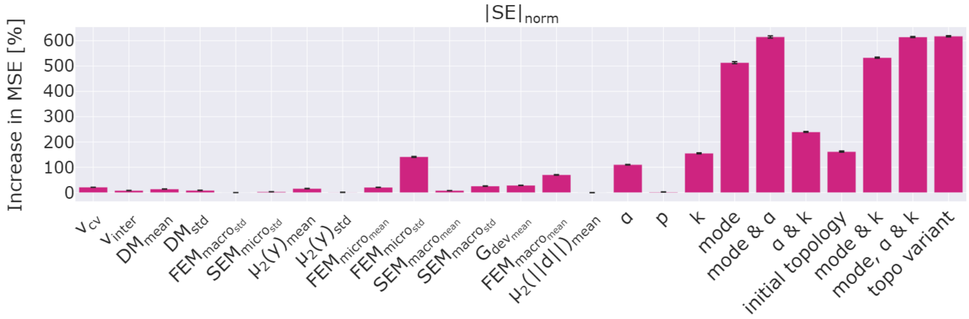
Computational effort

Figures 6.14a and 6.14b show the results for permutation importance regarding the computational effort $|SE|_{norm}$. For both heuristics, the effects on the MSE are greatest with permutations of the topology parameters. For COHDA, the mode is most relevant. But k and α are also important. However, mode and α alone are as important (+615 %) as all topology parameters together (+617 %). For IDICE, the parameter p, i.e. the degree of random rewiring when creating the initial small-world topology, also has a high importance score of +95%. Together, the topology parameters have a score of +271%.

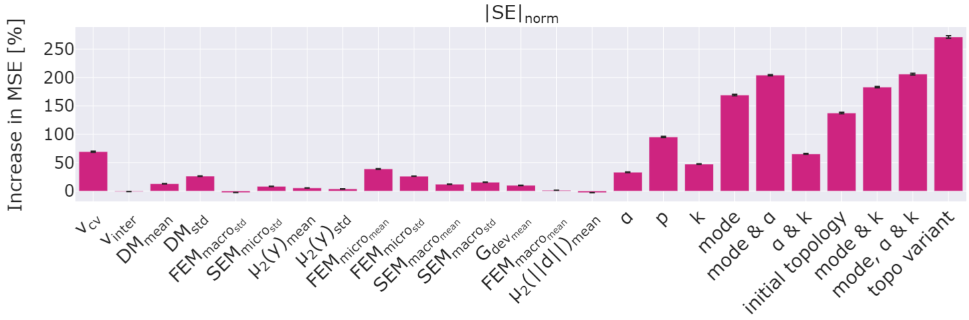
For this performance metric, only a few FLA metrics are relevant. For COHDA this refers only to $FEM_{micro_{mean}}$ (+ 141 %), which represents the average ruggedness and modality on micro scale across all subsearch spaces. For IDICE, all non-topology-related features are below an importance score of 100%. Some even have very small negative scores and are therefore considered irrelevant.

Communication overhead

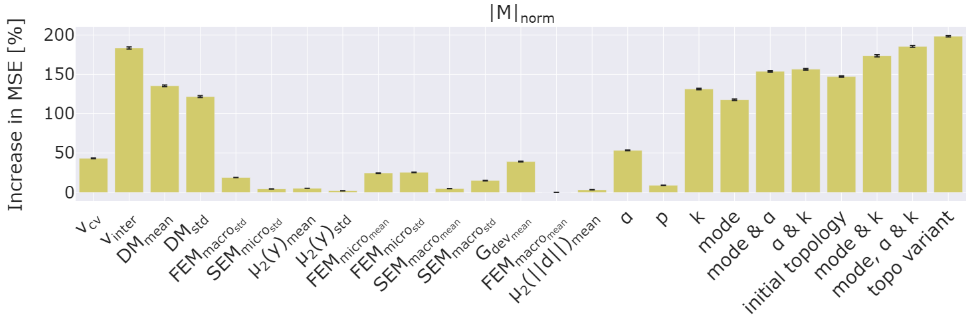
Figures 6.14c and 6.14d show the results for the permutation importances related to communication overhead $|M|_{norm}$. Again, the parameters of the CT are very important for both heuristics. For IDICE, the y-axis in fig. 6.14d has been interrupted because the very small reference MSE of 0.0005 results in extremely high permutation importance scores. The parameter k (+14,554%) and the mode (+6,220%) are already individually very important in IDICE. In combination, the parameters of the CT reach an importance of +16,6717% and an increase of the MSE to 0.0801. Topology parameters are also very important for COHDA, reaching an importance score of +198%. However, there are also some FLA metrics that are important for both heuristics. For COHDA, these include v_{inter} (+183%), DM_{mean} (+135%), and DM_{std}



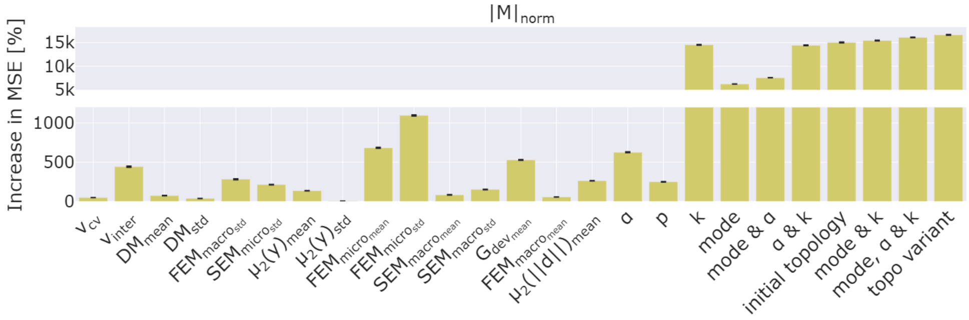
(a) Permutation importances $|SE|_{norm}$ for COHDA; base line MSE: 0.0093



(b) Permutation importances $|SE|_{norm}$ for IDICE; base line MSE: 0.0161



(c) Permutation importances $|M|_{norm}$ for COHDA; base line MSE: 0.0114



(d) Permutation importances $|M|_{norm}$ for IDICE; base line MSE: 0.0005

Fig. 6.14.: Permutation importances with respect to MSE for $|SE|_{norm}$ and $|M|_{norm}$; The CT parameters are considered individually as well as in combination. The *initial topology* includes k and p, the *topo variant* considers all topology parameters.

features	$f(x)_{norm}$	$f(x)_{norm-max}$	$\lvert SE\rvert_{norm}$	$\lvert M\rvert_{norm}$
v_{cv}	24.12%	589.32%	21.25%	43.20%
v_{inter}	20.05%	2.37%	8.28%	183.42%
DM_{mean}	767.82%	208.10%	14.22%	135.32%
DM_{std}	132.43%	32.75%	8.91%	121.69%
$FEM_{macro_{std}}$	3.55%	2.94%	0.16%	18.88%
$SEM_{micro_{std}}$	9.59%	42.68%	3.55%	4.37%
$\mu_2(y)_{mean}$	35.92%	11117.66%	15.90%	5.09%
$\mu_2(y)_{std}$	27.52%	37.14%	1.11%	2.05%
$FEM_{micro_{mean}}$	161.40%	20.14%	20.70%	24.47%
$FEM_{micro_{std}}$	65.22%	49.65%	141.45%	25.38%
$SEM_{macro_{mean}}$	30.76%	228.85%	8.23%	4.76%
$SEM_{macro_{std}}$	485.56%	-0.57%	25.93%	15.02%
$G_{dev_{mean}}$	2.29%	-0.53%	28.65%	39.08%
$FEM_{macro_{mean}}$	22.19%	102.52%	70.19%	-0.02%
$\mu_2(\lvert\lvert d\rvert\rvert)_{mean}$	23.33%	43.02%	-0.10%	3.28%
α	7.50%	20.39%	110.27%	53.35%
p	0.53%	-0.44%	2.38%	8.97%
k	24.99%	148.12%	155.40%	131.30%
$mode$	24.21%	2.90%	513.42%	117.69%
$mode\,\&\,\alpha$	31.90%	35.56%	615.18%	153.70%
$\alpha\,\&\,k$	30.21%	145.64%	239.70%	156.46%
$initial\,topology$	27.82%	148.14%	162.20%	147.13%
$mode\,\&\,k$	43.05%	137.06%	533.27%	173.49%
$mode,\,\alpha\,\&\,k$	47.64%	151.59%	614.85%	185.53%
$topo\,variant$	50.84%	151.91%	617.81%	198.44%

Tab. 6.3.: Permutation feature importance as a measure of increase or decrease of the MSE in % for COHDA

(+122%). For IDICE, v_{inter} (+442%), $FEM_{micro_{mean}}$ (+683%), $FEM_{micro_{std}}$ (+1096%), and $G_{dev_{mean}}$ (+528%) are again particularly important. Since these FLA metrics are also associated with solution quality or problem difficulty in several cases, it is not surprising that they are also relevant for predicting communication overhead. It is noteworthy that, with the exception of v_{inter}, there is no overlap between the two optimization heuristics. Since the communicative behavior of the two heuristics is very different, and since the correlation analysis already yielded very different results for M_{norm}, the different importance of the FLA metrics is not unexpected. Nevertheless, the strong influence of the CT parameters can be observed for both heuristics.

Summary

The following observations can be summarized after considering the individual permutation importances for the performance dimensions: As expected based on the

features	$f(x)_{norm}$	$f(x)_{norm-max}$	$\|SE\|_{norm}$	$\|M\|_{norm}$
v_{cv}	-2.85%	126.38%	69.36%	48.65%
v_{inter}	160.07%	30.62%	-0.95%	442.25%
DM_{mean}	27.81%	39.17%	12.96%	73.74%
DM_{std}	-1.36%	-1.81%	26.21%	37.96%
$FEM_{macro_{std}}$	-2.74%	-1.06%	-2.61%	282.70%
$SEM_{micro_{std}}$	4.43%	5.46%	8.08%	213.55%
$\mu_2(y)_{mean}$	2.04%	2679.80%	5.42%	135.92%
$\mu_2(y)_{std}$	-2.42%	2.10%	3.76%	4.56%
$FEM_{micro_{mean}}$	2.08%	8.63%	39.03%	682.66%
$FEM_{micro_{std}}$	33.31%	-1.82%	26.14%	1095.65%
$SEM_{macro_{mean}}$	27.01%	17.16%	12.00%	83.59%
$SEM_{macro_{std}}$	-0.50%	-2.13%	15.37%	152.15%
$G_{dev_{mean}}$	27.24%	0.16%	9.97%	528.06%
$FEM_{macro_{mean}}$	-0.68%	12.24%	1.37%	55.94%
$\mu_2(\|\|d\|\|)_{mean}$	-2.41%	-1.72%	-2.79%	262.91%
α	5.78%	-1.11%	33.06%	626.13%
p	23.75%	0.46%	95.27%	249.46%
k	22.24%	2.89%	47.44%	14553.68%
$mode$	15.36%	-2.76%	169.14%	6220.19%
$mode\,\&\,\alpha$	26.19%	0.63%	204.10%	7564.66%
$\alpha\,\&\,k$	23.61%	1.48%	65.38%	14439.93%
$initial\,topology$	38.45%	5.31%	137.44%	15064.37%
$mode\,\&\,k$	27.44%	1.70%	183.07%	15476.63%
$mode,\ \alpha\,\&\,k$	34.17%	4.64%	205.96%	16125.36%
$topo\,variant$	41.27%	7.67%	271.36%	16671.70%

Tab. 6.4.: Permutation feature importance as a measure of increase or decrease of the MSE in % for IDICE

correlation analysis, the FLA features are most relevant for predicting the solution quality based performance metrics. Which features are most important depends on the heuristic. However, the topology parameters also have an influence, which is higher for COHDA at $f_{norm-max}$ and for IDICE at f_{norm}. In contrast, SE_{norm}, i.e. the computational cost, depends mainly on the topology parameters. The influence of FLA metrics is very small. Which metrics have any influence at all differs between the two heuristics. Also for M_{norm}, the communication overhead, the influence of the topology parameters is very high for both heuristics. Here, some FLA metrics are slightly more relevant (compared to SE_{norm}), which are again different FLA metrics for the two heuristics.

Tables 6.3 and 6.4 summarize the feature importance for all input features. The importance of each feature varies greatly depending on the optimization heuristic and performance metric. However, each feature is justified in the sense that it has a unique contribution to at least one combination of heuristic and performance metric that is relevant to the prediction. The few negative importance scores are very small. This indicates that the features do not interfere much with the predictions. Since each feature is relevant to at least one prediction constellation, the feature set presented is kept as-is for the sake of generalizability.

The main findings so far are as follows:

- Even untuned random forest regressors are able to make good predictions with low MSEs for all performance metrics.
- The resulting permutation importance scores show that both the problem characteristics in the form of the FLA metrics and the parameters of the CTVs influence the performance dimensions and can be used to predict the performance metrics.
- The importance of individual input features varies depending on the performance metric and optimization heuristic.
- Each feature is relevant to the prediction of at least one performance metric for one heuristic.
- These results support the postulated causal relationship between problem characteristics, CTVs, and performance dimensions.

The analysis of the importance of features has shown that there is a statistical correlation between the distributed FLA metrics in combination with the parameters of the CT and the performance dimensions, and that ML-based predictions are possible.

So far, the predictive ability of ML models has been studied only in an abstract way, using metrics such as MSE or R2 to investigate the general performance of the models. In the following, a more in-depth investigation will be carried out by examining the predictions of the trained models in more detail and drawing conclusions for the postulated causal relationship.

6.3 Evaluation of predictive capability of ML models

In order to use ML models to support research hypothesis 3, it is first necessary to train models that produce sufficiently good prediction results. Second, these predictions must be examined to determine whether they distinguish reasonably between different problem instances and CTVs. Together, these two aspects determine the predictive capability of a model for the use case in question.

Therefore, a specific ML model type is selected in section 6.3.1 below. The focus is on finding a sufficiently good model with reasonable effort. It is not necessary to find the best possible model for this evaluation. Then, in section 6.3.2, the predictions of the trained models are analyzed in detail, and in section 6.3.3, they are applied to perform an exemplary selection of topology variants. Finally, as a first step towards transfer to real-world problems, the performance of the models is investigated under the influence of noise in section 6.3.4.

For all subsequent investigations, the data was initially split into training and validation data (80/20), stratified by problem instance. Thus, 80% of all data, equally distributed across all problem instances, was used to train the models, while 20% was kept for later evaluation. The evaluation data is used in sections 6.3.2 and 6.3.3.

6.3.1 Choosing models for ToVarO

In the permutation importance study in the previous section, random forest regressors were used. They immediately showed good results with respect to the MSE. Therefore, they are a reasonable candidate for the selection of a suitable model for the Topology Variant Optimization (ToVarO). The goal is to find models which are well suited to investigate learnability of relations between problem characteristics, exchange topologies, and performance measures. Therefore, a short investigation whether another machine learning method might be more suitable will be carried out.

The prediction problem is a nonlinear problem and the feature importance analysis showed that the features have different influences. Artificial neural networks are the state of the art for such problems and are therefore well suited for comparison with the random forest regressors. Since there are no temporal dependencies between the data, a Multi-layer Perceptron (MLP) was tested. For the MLP, parameter tuning was performed using Bayesian optimization. Due to the assumed complexity of the problem, 1-2 hidden layers were assumed to be sufficient. Since the parameter *mode* was preprocessed with One Hot Encoding, the number of input parameters is 21. Accordingly, a reasonable number of neurons could be approximately in this range. Furthermore, the optimizer, the learning rate, the batch size, and the number of epochs were varied. Parameter tuning was also performed for the random forests. For this purpose, a random search for hyper parameter optimization [BB12] was applied[4]. Table 6.5 provides an overview of the value ranges used for the two hyper parameter tunings.

	parameter	**values**
MLP	**number** of **hidden layers**	1,2
	number of **neurons** (for one or two layers)	[5,50]
	optimizer	Adam, SGD
	learning rate	$[1e^{-5}, 0.1]$
	batch size	[32, 1024]
	epochs	[20, 40]
forest	**number of trees**	[10, 300]
	criterion to measure the quality of a split	squared error, friedman mse, poisson
	max depth of trees	10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, None
	minimum number of **samples** required to **split** an internal node	2, 5, 10
	minimum number of **samples** required to be at a **leaf** node	1, 2, 4

Tab. 6.5.: Parameter setups for MLP parameter tuning with Bayesian optimizer and random forest regressor with randomize grid search, both evaluated with 3-fold cross validation

The Bayesian optimizer performed 110 evaluations, each with 3-fold cross-validation, again stratifying the folds by problem instances. The random search for hyperparameter optimization of the random forest regressors ran 30 iterations for each performance metric and optimization heuristic, also using 3-fold cross-validation and stratifying the folds by problem instances. The final parameter selection is provided in the appendix in table C.9. The tuned MLPs performed well. However, they did not quite achieve the same results as the random forest regressors. Table 6.6

---

[4] implementation by [Ped+11]

shows the results of the trained MLPs and random forests for all combinations of optimization heuristics and performance metrics. The MSE and R^2 are obtained for the evaluation data, which is unseen data to the models. The random forests always perform at least as well, and often better than the MLPs. It is likely that the MLPs could be improved even further with more extensive parameter tuning. However, since the random forest regressors already show quite low error values, they seem to be suitable for further evaluation of the predictive capabilities. Thus, the following evaluation is performed with the random forest regressors retrained with the full training data set.

		$f(x)_{norm}$		$f(x)_{norm-max}$		$\|SE\|_{norm}$		$\|M\|_{norm}$	
		MSE	R^2	MSE	R^2	MSE	R^2	MSE	R^2
COHDA	**MLP**	0.0301	0.8020	0.0016	0.9588	0.0104	0.7277	0.0175	0.4629
	forest	0.0208	0.8354	0.0005	0.9878	0.0091	0.7778	0.0111	0.6740
IDICE	**MLP**	0.0269	0.6424	0.0012	0.9479	0.0170	0.5794	0.0014	0.9619
	forest	0.0235	0.688	0.0011	0.9567	0.0151	0.6181	0.0005	0.9889

Tab. 6.6.: Quality Metrics for the best found version of the MLP and the random forest regressor, both for COHDA and IDICE and all performance parameters on the evaluation data

6.3.2 Analysis of predictions

The fact that the model predictions are on average close to the actual values is already indicated by the low MSEs in table 6.6. However, it is still unclear what this means at a more detailed level. For example, the models might predict the same value for a problem instance regardless of the topology variant. After all, for COHDA, the permutation importance scores of the topology parameters at $f(x)_{norm}$ were rather low (see fig. 6.13a). Therefore, in the following, the predictions for several problem instances will be examined in order to investigate the prediction ability in more detail. For this purpose, the same instances were chosen as in section 6.1.2, namely *Drop Wave 1* and *Weierstrass 1*. These two problems are particularly well suited because they differ greatly both in the effects of the CTVs and in the relevance of the CT for solution quality (see fig. 6.3). Furthermore, the previous figures provide additional illustrative material.

The figures 6.15 and 6.16 show the predictions obtained with the trained random forest regressors for all performance metrics and both heuristics. The corresponding metric is always plotted on the y-axis. The CTVs are plotted on the x-axis. For the same problem instance, the input features of the ML model differ only in the topology parameters. The other input parameters are the FLA metrics, which are always the same for a particular problem instance. The mean value from the training data was plotted as an orange diamond for each problem instance and each topology variant. The mean from the evaluation data was plotted as a blue circle. The predictions are plotted as yellow stars.

First, we take a closer look at the results for COHDA in fig. 6.15. The predictions, i.e. the stars, are usually very close to the mean values of the training data, i.e. the diamonds. This means that the models have learned to predict values close to the mean of the training data per problem instance and per CTV. Of course, this could be an indication of overfitting. However, the parameters were determined by 3-fold cross-validation, and the trees were then retrained on the entire training data for further evaluation. The fact that these parameter settings performed well in the cross-validation supports the notion that predictions close to the mean are good approximations. The mean values from the evaluation data are slightly different, but in general the pattern of differences between the topology variants is still very similar to the training data. These differences are due to variation caused by random effects and maybe other unknown factors. Previous figures (e.g. figures 6.3, 5.7 and 5.8) also show that the results for a CTV exhibit some variance. The low MSEs indicate that the predictions close to the mean values of the training data learned by the models are good approximations of how a topology variant will perform on a problem instance. If the results of a topology variant were completely unpredictable, this would result in much higher error rates. The learning of this pattern by the models shows that the CTVs have distinct and learnable effects on all performance dimensions. Furthermore, these effects clearly differ between different problem instances, as in the examples *Drop Wave 1* and *Weierstrass 1*.

The results for IDICE in fig. 6.16 show a slightly different pattern. For $|SE|_{norm}$ and $|M|_{norm}$, the predictions are again very close to the mean values of the training data. The ML models learned slightly more generalized patterns for the two metrics based on solution quality. Since the results are much more scattered than for COHDA, this is to be expected. As a consequence, the R^2 scores for IDICE are also significantly lower than for COHDA (see table 6.6). At the same time, the MSEs are still at a low level. Thus, predictions for IDICE seem to be more challenging. Considering the low MSE scores, the predicted values still appear to be accurate enough to select an appropriate CTV for a specific problem. Accordingly, both heuristics show a learnable pattern of relationships between the problem characteristics, the CTVs, and the algorithmic performance.

The two selected example problem instances differ significantly in their FLA metrics. In order to show that even smaller differences have an impact and that the models do not only differentiate according to the basic benchmark functions, fig. 6.17 shows

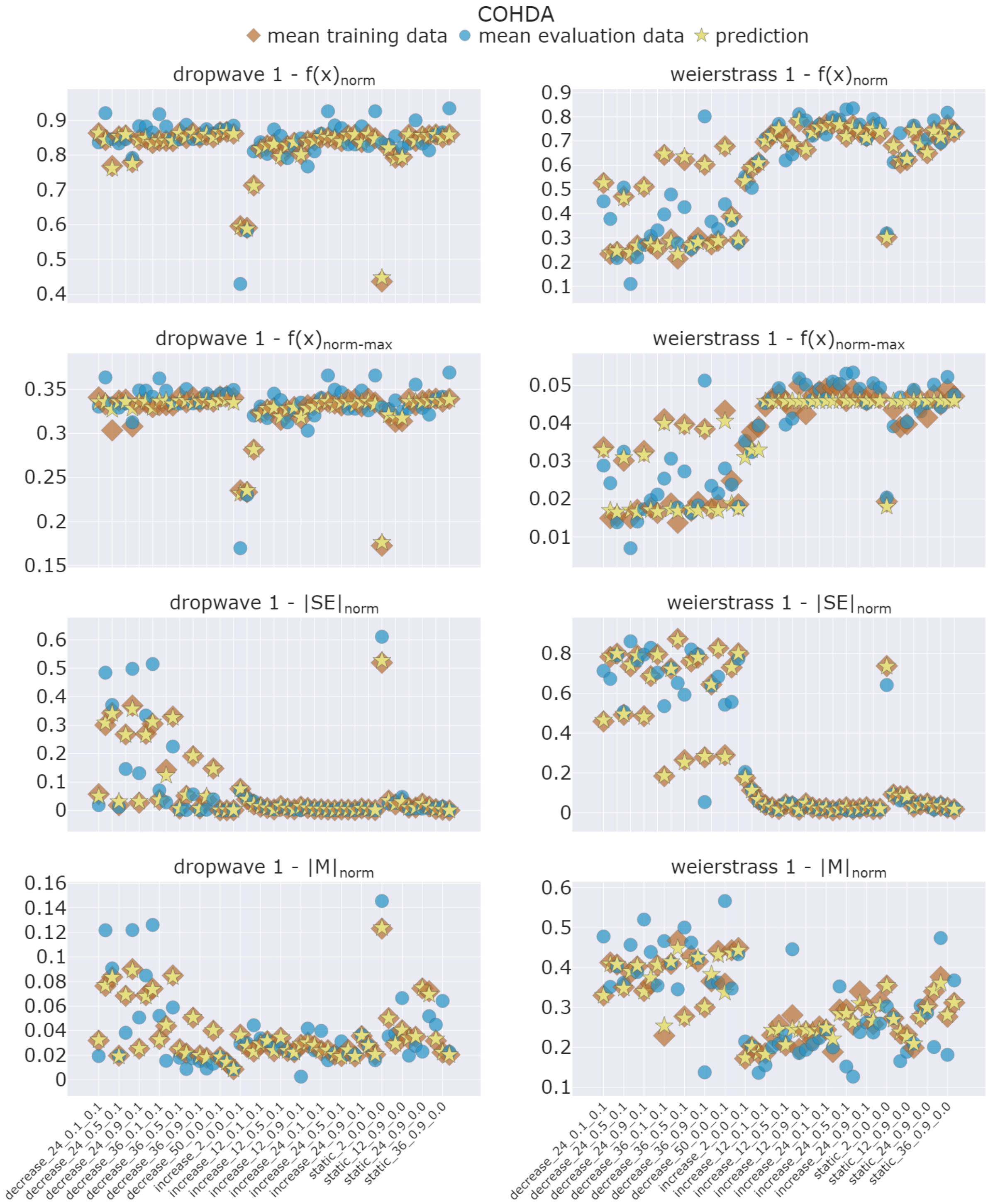


Fig. 6.15.: Exemplary predictive performance for COHDA; The corresponding metric is always plotted on the y-axis. The CTVs are plotted on the x-axis and are encoded in the form $mode\_k\_p\_\alpha$.

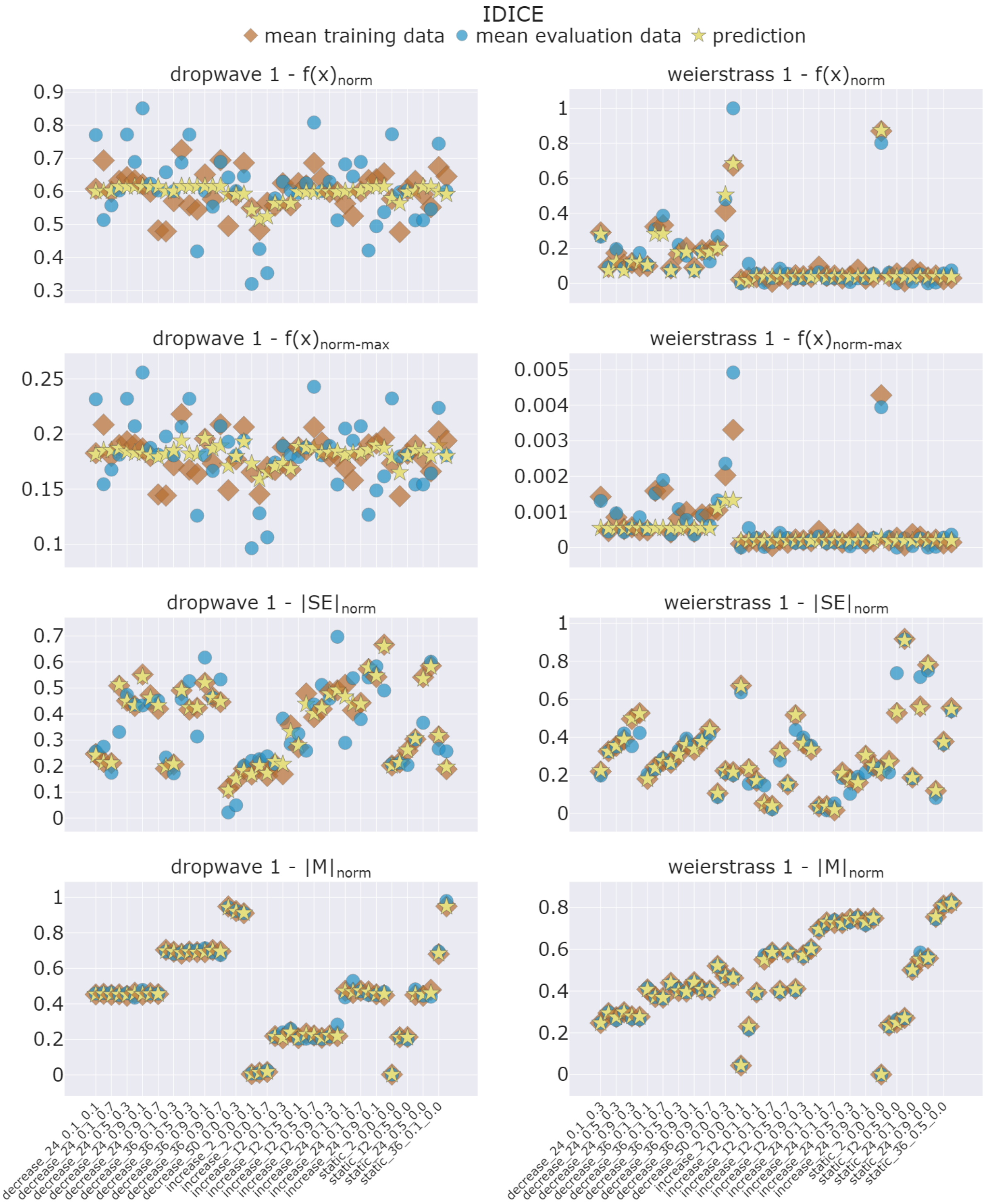


Fig. 6.16.: Exemplary predictive performance for IDICE; The corresponding metric is always plotted on the y-axis. The CTVs are plotted on the x-axis and are encoded in the form $mode\_k\_p\_\alpha$.

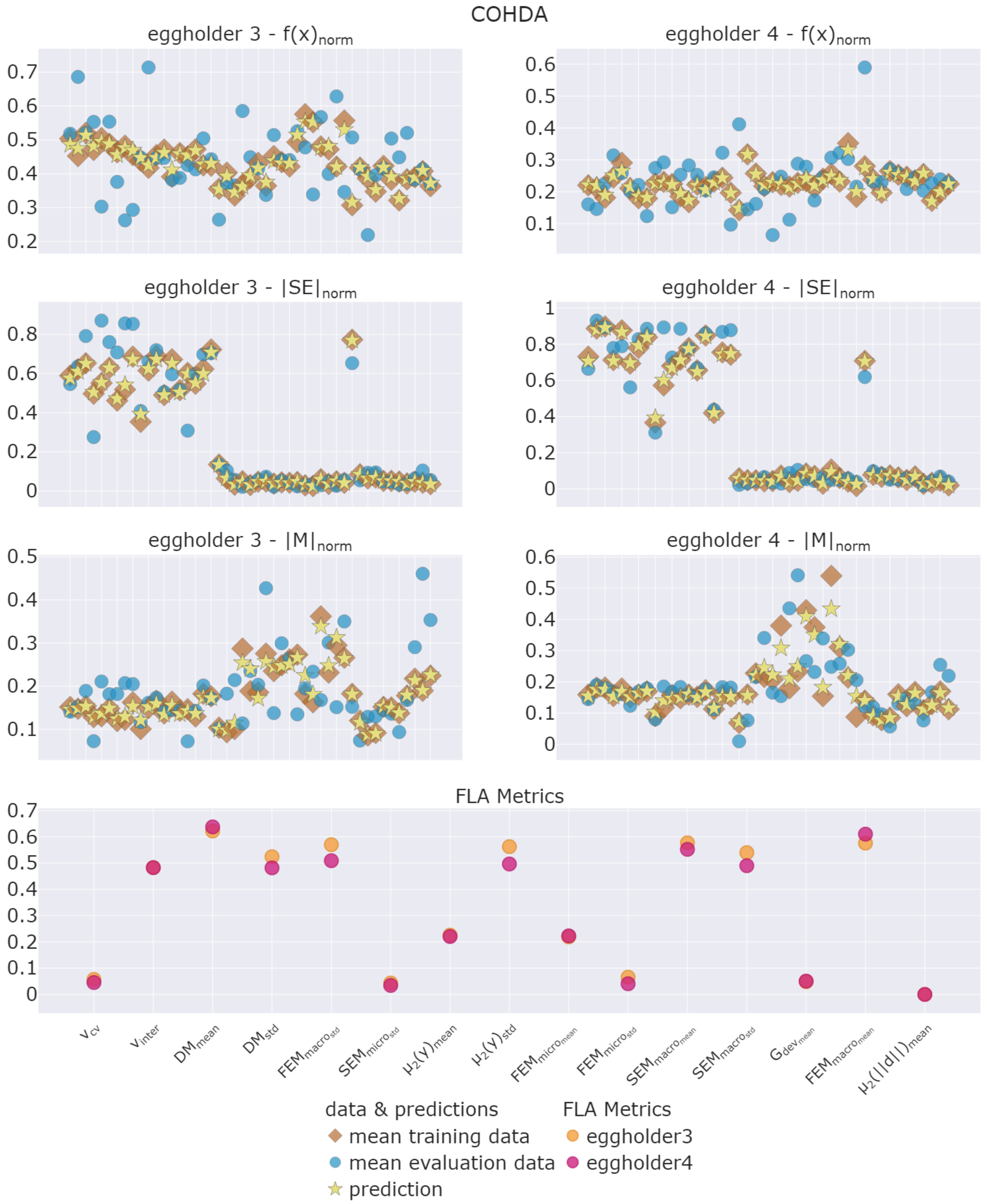


Fig. 6.17.: Exemplary predictive performance for highly similar problem instances; The corresponding metric is always plotted on the y-axis. The CTVs are plotted on the x-axis. Additionally, the FLA metrics for the problem instances are shown

the comparison for two quite similar *Eggholder*-based problem instances for COHDA. The bottom row shows the FLA metrics for the two problem instances. The values for all metrics are very close, showing that both instances are quite similar, but with minor differences. Looking at the performance metrics, however, it is clear that there are distinctions in the effects of the CTVs. The ML models again learned values close to the respective mean values of the training data. This shows that even the small variations in the FLA metrics are sufficient to change the effect of the CTVs and that the models are able to learn them. This is a confirmation of the suitability of the FLA metrics as indicators for predicting the performance of CTVs and also for characterizing the problem properties.

Intermediate results

- The random forest regressors learn to predict values close to the mean values of the training data per problem instance and per CTV.
- Combined with the low error rates on the evaluation data, this implies that the mean values are good approximations of the performance of the CTVs.
- The examples show differences between the topology variants, which emphasizes the purpose of the targeted selection.
- The performance of the CTVs varies even between very similar problem instances. The mean values learned by the ML models are still good predictors and the differentiation based on the subtle differences in the input parameters is sufficient. This underpins the suitability of the distributedly computed FLA features for problem-specific parameter selection for heuristics.

Now that the general prediction behavior has been investigated, the next step is to use the trained models to perform an exemplary topology variant selection.

6.3.3 Prediction based topology selection

The selection of an appropriate topology variant is a multi-objective optimization problem, since all different problem dimensions can be considered. The prioritization of the performance dimensions may differ depending on the use case. For example, in some use cases, solution quality is of paramount importance, while computational and communication efforts are less critical. However, there are also situations where solution quality is considered less important and computational efficiency is more relevant. A simple way to represent the different priorities of the performance dimensions is to use a weighted sum. An individual weighting factor can be assigned to each dimension.

In addition, there are two performance metrics related to solution quality. While $f(x)_{norm}$ shows how different CTVs affect the solution quality, $f(x)_{norm-max}$ is an indicator of how important the CT is for the solution quality in general. Therefore, it is reasonable to use only $f(x)_{norm}$ for the weighted sum by default. Optionally, the weighting factor of $f(x)_{norm}$ can be multiplied by $f(x)_{norm-max}$ to reduce the influence of $f(x)_{norm}$ on the selection of the topology variant. This is useful when the influence of $f(x)_{norm}$ on the specific problem instance is rather small and it is more appropriate to put emphasis on the other performance dimensions. In the following, different prioritizations (or weight distributions) are tested, both with and without the consideration of $f(x)_{norm-max}$. Table 6.7 shows the setups for the study.

setup1	$w_1 = 0.33 \cdot f(x)_{norm} + 0.33 \cdot \|SE\|_{norm} + 0.33 \cdot \|M\|_{norm}$ Setup with equal weights of the power dimensions
setup2	$w_2 = 0.8 \cdot f(x)_{norm} + 0.1 \cdot \|SE\|_{norm} + 0.1 \cdot \|M\|_{norm}$ Setup with strong emphasis on solution quality
setup3	$w_3 = 0.8 \cdot f(x)_{norm} \cdot f(x)_{norm-max} + 0.1 \cdot \|SE\|_{norm} + 0.1 \cdot \|M\|_{norm}$ Setup with strong emphasis on solution quality, but relativized by the general influence of CTVs on solution quality $f(x)_{norm-max}$

Tab. 6.7.: Setups to prioritize performance dimensions using weighted sums

The chosen setups allow to show the result of an equal weighting of the performance dimensions (w_1), the result of a strong focus on the solution quality (w_2), and how the simultaneous inclusion of the general relevance of the topology variant for the solution quality affects the result in the second case (w_3). In order to use the trained models to select an appropriate CTV for a problem instance, the following steps must be performed:

1. Predict all performance metrics for all topology variants
2. Calculate scores for the given weighted sum representing the prioritization of performance dimensions
3. Select the best topology variant

This procedure was performed using the *Penalized Schwefel 2.26 - 8* problem instance and the optimization with COHDA as an example. The results are displayed in table 6.8. The best topology variants for w_1, w_2, and w_3 are highlighted in different colors. For a better illustration of the results, all predictions and the values of the training data for all performance metrics for *Penalized Schwefel 2.26 - 8* are also plotted in fig. 6.18. The best topology variants are highlighted in the same colors as in table 6.8. For this problem instance, all CTVs with mode *decrease* and the static ring topology show good solution qualities, but require high effort. In contrast, the variants with mode *increase* or *static* are comparatively weak in terms of solution quality, but have an advantage for $|SE|_{norm}$ and $|M|_{norm}$.

For w_1 the variant $increase$_2_0.0_0.1 was chosen. This is the topology variant that starts with a ring topology ($k = 2,\ p = 0$) and slowly increases the number of edges ($\alpha = 0.1$). Looking at fig. 6.18, it is clear that this topology variant is one of the least computationally and communicationally expensive, but at the same time results in moderate values of $f(x)_{norm}$. Therefore, this topology seems to be a reasonable choice if the performance dimensions are equally weighted.

For w_2, the focus was on the solution quality. Here the topology variant $decrease$_36_0.5_0.7 was chosen. This starts with a rather strongly meshed initial topology ($k = 36,\ p = 0.5$) and reduces the number of edges very quickly ($\alpha = 0.7$). This topology variant is the one with the smallest predicted value for $f(x)_{norm}$. The increased $|SE|_{norm}$ and $|M|_{norm}$ are negligible in consequence of w_2. However, if the importance of the topology variants for the solution quality is taken into account, as in w_3, the choice is $increase$_2_0.0_0.7. This variant starts again with a ring topology, but increases the number of edges fast ($\alpha = 0.7$). This is because the values for $f(x)_{norm-max}$ are very small. Therefore, $f(x)_{norm}$ has even less weight than for w_1. This stronger focus on $|SE|_{norm}$ and $|M|_{norm}$ might be more appropriate considering this assumed low relevance of CTV for solution quality.

Intermediate results and discussion

The example illustrates how the weighted sum approach and ML-based predictions can be used to select topology variants for a given problem. With theoretical problem instances, the feasibility of the overall concept as presented in the development objective (see fig. 6.1) was thus demonstrated. The application works for all problem instances investigated in this thesis.

Nevertheless, the exemplary evaluation revealed several areas for potential improvement. One reasonable extension could be the additional consideration of uncertainties. For different topology variants, the variance of the results may vary. Such uncertainty factors could be part of the selection process, e.g. by additional predictions.

Another improvement could be a more sophisticated evaluation of the relevance of CTVs to solution quality. Including $f(x)_{norm-max}$ seems reasonable for all cases where computational and communication overhead is relevant. So far, for its computation a value determined by maximization has been used as an upper bound. This is a theoretically achievable maximum value of bad solution quality. But no conclusion could be drawn about the probability of occurrence of such a value.

topology variant	$f(x)_{norm}$	$f(x)_{norm-max}$	$\|SE\|_{norm}$	$\|M\|_{norm}$	w_1	w_2	w_3
decrease_24_0.1_0.1	0.162	0.013	0.396	0.427	0.325	0.212	0.084
decrease_24_0.1_0.3	0.053	0.013	0.648	0.637	0.441	0.170	0.129
decrease_24_0.1_0.7	0.035	0.013	0.589	0.569	0.394	0.144	0.116
decrease_24_0.5_0.1	0.173	0.013	0.370	0.409	0.314	0.216	0.080
decrease_24_0.5_0.3	0.067	0.013	0.627	0.608	0.430	0.177	0.124
decrease_24_0.5_0.7	0.078	0.013	0.562	0.534	0.387	0.172	0.110
decrease_24_0.9_0.1	0.174	0.013	0.431	0.464	0.353	0.229	0.091
decrease_24_0.9_0.3	0.050	0.013	0.567	0.557	0.388	0.153	0.113
decrease_24_0.9_0.7	0.036	0.013	0.602	0.606	0.410	0.149	0.121
decrease_36_0.1_0.1	0.164	0.013	0.434	0.503	0.363	0.225	0.095
decrease_36_0.1_0.3	0.039	0.013	0.676	0.672	0.458	0.166	0.135
decrease_36_0.1_0.7	0.051	0.013	0.632	0.637	0.435	0.167	0.127
decrease_36_0.5_0.1	0.165	0.013	0.617	0.687	0.485	0.262	0.132
decrease_36_0.5_0.3	0.102	0.013	0.685	0.686	0.486	0.219	0.138
decrease_36_0.5_0.7	0.032	0.013	0.572	0.560	0.384	**0.139**	0.114
decrease_36_0.9_0.1	0.185	0.013	0.435	0.502	0.371	0.242	0.096
decrease_36_0.9_0.3	0.112	0.013	0.571	0.574	0.415	0.204	0.116
decrease_36_0.9_0.7	0.046	0.013	0.545	0.523	0.368	0.144	0.107
decrease_50_0.0_0.1	0.199	0.013	0.445	0.547	0.393	0.259	0.101
decrease_50_0.0_0.3	0.065	0.013	0.622	0.638	0.437	0.178	0.127
decrease_50_0.0_0.7	0.098	0.013	0.598	0.598	0.427	0.198	0.121
increase_2_0.0_0.1	0.224	0.034	0.055	0.106	**0.127**	0.195	0.022
increase_2_0.0_0.3	0.298	0.034	0.034	0.122	0.150	0.254	0.024
increase_2_0.0_0.7	0.299	0.034	0.018	0.106	0.140	0.252	**0.021**
increase_12_0.1_0.1	0.294	0.034	0.016	0.158	0.154	0.253	0.025
increase_12_0.1_0.3	0.400	0.034	0.010	0.190	0.198	0.340	0.031
increase_12_0.1_0.7	0.385	0.034	0.012	0.284	0.225	0.337	0.040
increase_12_0.5_0.1	0.303	0.034	0.016	0.157	0.157	0.260	0.026
increase_12_0.5_0.3	0.333	0.034	0.011	0.202	0.180	0.288	0.030
increase_12_0.5_0.7	0.388	0.034	0.010	0.256	0.216	0.337	0.037
increase_12_0.9_0.1	0.293	0.034	0.017	0.173	0.160	0.254	0.027
increase_12_0.9_0.3	0.353	0.034	0.010	0.187	0.182	0.302	0.029
increase_12_0.9_0.7	0.381	0.034	0.012	0.284	0.223	0.334	0.040
increase_24_0.1_0.1	0.303	0.034	0.009	0.186	0.164	0.262	0.028
increase_24_0.1_0.3	0.295	0.034	0.011	0.291	0.197	0.267	0.038
increase_24_0.1_0.7	0.290	0.034	0.010	0.284	0.193	0.261	0.037
increase_24_0.5_0.1	0.290	0.034	0.011	0.220	0.172	0.255	0.031
increase_24_0.5_0.3	0.302	0.034	0.009	0.245	0.184	0.267	0.034
increase_24_0.5_0.7	0.296	0.034	0.011	0.291	0.197	0.267	0.038
increase_24_0.9_0.1	0.318	0.034	0.011	0.227	0.183	0.278	0.032
increase_24_0.9_0.3	0.323	0.034	0.011	0.286	0.204	0.288	0.038
increase_24_0.9_0.7	0.302	0.034	0.009	0.253	0.186	0.268	0.034
static_2_0.0_0.0	0.072	0.034	0.598	0.569	0.409	0.174	0.119
static_12_0.1_0.0	0.305	0.034	0.029	0.159	0.163	0.263	0.027
static_12_0.5_0.0	0.251	0.034	0.028	0.151	0.142	0.218	0.025
static_12_0.9_0.0	0.262	0.034	0.033	0.172	0.154	0.230	0.028
static_24_0.1_0.0	0.282	0.034	0.017	0.200	0.165	0.247	0.029
static_24_0.5_0.0	0.248	0.034	0.021	0.255	0.173	0.226	0.034
static_24_0.9_0.0	0.263	0.034	0.021	0.250	0.176	0.238	0.034
static_36_0.1_0.0	0.294	0.034	0.015	0.290	0.198	0.266	0.038
static_36_0.5_0.0	0.279	0.034	0.016	0.322	0.204	0.257	0.041
static_36_0.9_0.0	0.275	0.034	0.012	0.220	0.167	0.243	0.031
static_50_0.0_0.0	0.293	0.034	0.010	0.291	0.196	0.265	0.038

Tab. 6.8.: Predictions for *Penalized Schwefel 2.26 - 8* using the weighted sums specified in table 6.7; The best topology variants are highlighted in color.

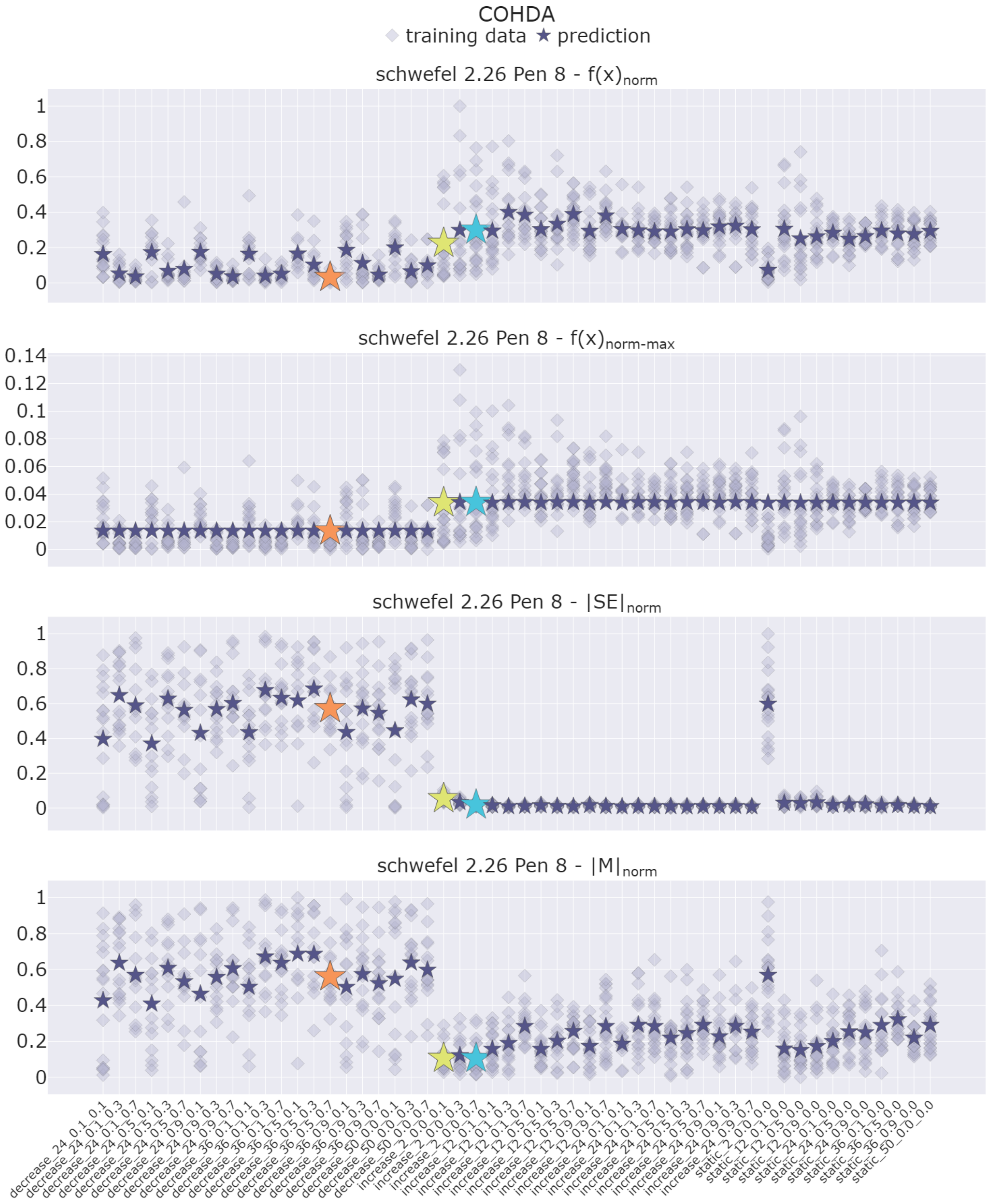


Fig. 6.18.: Exemplary prediction based CTV selection for *Penalized Schwefel 2.26 - 8*; The corresponding metric is always plotted on the y-axis. The CTVs are plotted on the x-axis. Topology variants are encoded in the form $mode\_k\_p\_\alpha$.

This could lead to an underestimation of the influence of the exchange topologies with the current approach. To evaluate the relevance of CTVs to solution quality, other metrics could be used or specific domain knowledge could be applied. These approaches remain subject to future research.

After demonstrating the applicability of the overall concept to the problem instances generated on the basis of the benchmark functions, the last step of the evaluation is to investigate the effects of noise. This represents a first step towards transferability to real-world problems.

6.3.4 Sensitivity to noise

In real-world optimization problems, the same choice of input parameters does not always lead to the same result. This can be caused by incomplete knowledge or control of the system, or by measurement inaccuracies. In theoretical optimization problems, this property can be modeled by noise, i.e. the addition of randomness to the return values of the objective functions.

So far, training ML models has shown that there are learnable patterns in the data that can be used to select a problem-specific CTV to improve performance. The question is whether this is still the case for noisy data. The predicted values were in most cases very close to the mean values of the training data. This could be an indication of overfitting. However, if the patterns can be learned consistently even in the presence of noise, this further supports the validity of these patterns. Furthermore, it is a first step towards transferability to real-world problems that require the ability to process noisy data.

To investigate the robustness of the patterns to noise, the optimization runs were repeated with noise for each of the 23 benchmark functions for the first three derived composite spaces. In this way, noisy data is available for 69 problem instances. For each problem instance, all 53 CTVs were again computed with 9 different seed combinations. In total, the noisy data set contains 32,913 optimization runs per optimization heuristic. No seeds were set for the application of the noise itself. This resulted in different values for the same points in the search space when computed multiple times. In addition to the optimization runs, the distributed FLA was also run with noise applied.

The resulting noisy dataset was again split into training and evaluation data (70:30) ensuring an even distribution of problem instances. The random forest regressors were re-trained on the combined training set of noiseless and noisy data. The same settings as in table 6.5 were used for parameter tuning with 3-fold cross-validation.

The resulting parameter settings are shown in table C.10. However, before the effects of noise on the prediction of performance metrics can be examined, the type of noise must first be specified. Different types of noise are introduced below.

Application of noise

Typically, noise is applied by adding or multiplying random values to a function value with some probability. In [Fin+], Finck et al. presented three different types of noise for real-parameter black-box optimization benchmarking: the Gaussian, the uniform, and the Cauchy noise model.

The **Gaussian noise model** is defined as:

$$f_{GN}(f, \beta) = f \cdot exp(\beta\, \mathcal{N}(0,1)) \tag{6.1}$$

where f is the noiseless value obtained by the benchmark function and $\mathcal{N}(0,1)$ draws a random value in the interval $[0,1]$ from a log-normal distribution. The parameter β controls the noise strength. It is set to $\beta = 0.1$ for moderate noise. This noise model is scale invariant.

The **uniform noise model** is defined as:

$$f_{UN}(f, \alpha, \beta) = f \cdot \mathcal{U}(0,1)^{\beta}\, max\Big(1, \big(\frac{10^9}{f+\epsilon}\big)^{\alpha\, \mathcal{U}(0,1)}\Big) \tag{6.2}$$

where $\mathcal{U}(0,1)$ draws a random samples in the interval $[0,1]$ from a uniform distribution. The parameters α and β control the noise strength and are set to $\alpha = 0.01(0.49 + \frac{1}{D})$ (D is the number of dimensions) and $\beta = 0.01$. The parameter ϵ is set to 10^{-99} to prevent the division by zero. This noise model is not scale invariant, since the strength of the noise increases with decreasing positive values of f and thus becomes more severe for most benchmark functions near (local) optima.

The **Cauchy noise model** is defined as:

$$f_{CN}(f, \alpha, p) = f + \alpha\, max\Big(0, 1000 + \prod_{\mathcal{U}(0,1)<p} \frac{\mathcal{N}(0,1)}{|\mathcal{N}(0,1)| + \epsilon}\Big) \tag{6.3}$$

where α defines the strength and p the frequency of the noise. The parameters are set to $\alpha = 0.01$ and $p = 0.2$ for moderate but frequent noise. Again the parameter ϵ serves to prevent the division by 0. It is set to $\epsilon = 10^{-199}$.

In [Fin+], Finck et al. propose to apply a final step after the calculation of the noise affected function values in order to "to achieve a convenient testing for the target function value". This step is applied here as well. The **final function value** is thus computed as follows:

$$f_{XX}(f,\ldots) = \begin{cases} f_{XX}(f,\ldots) + 1.01 \cdot 10^{-8} & \text{if } f \geq 10^{-8} \\ f & \text{otherwise} \end{cases} \tag{6.4}$$

Section 6.3.4 shows an exemplary application of the noise models to the benchmark function Alpine no. 2 with 2 decision variables. While the Gaussian noise is quite moderate and evenly distributed, the uniform noise model is more severe and the noise level increases in certain areas since it is not scale invariant. The Cauchy noise model shows a completely different type of noise. The majority of the function values are not affected by the noise. But there are some large outliers that occur occasionally. This makes it hard to predict the degree of noise for a given value.

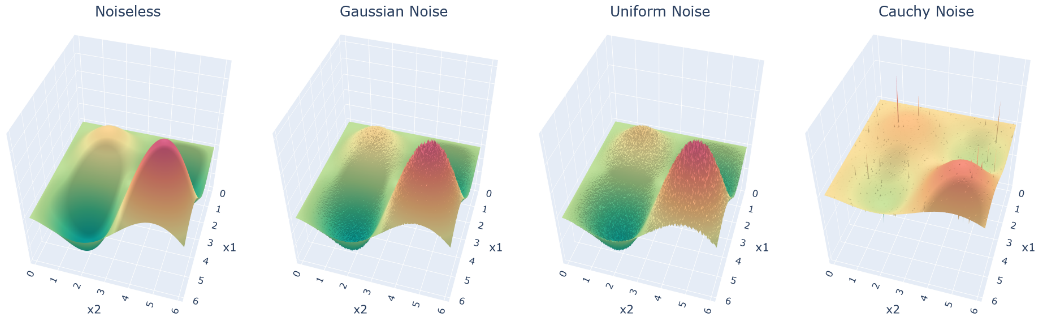


Fig. 6.19.: Application example of the noise models

In the following, the Gaussian noise model is applied since it is one of the most commonly used types of noise. Furthermore, it is a reasonable choice for modeling noise in optimization problems because many real-world phenomena, such as electronic circuits and communication systems, exhibit approximately Gaussian behavior [PP02]. Moderate Gaussian noise with $\beta = 0.1$ is applied.

Effect of Noise

To investigate the effects of the noisy data, the overall performance of the retrained random forest regressors is examined using the MSE and R^2 values. This is followed by a detailed look at the predictions for individual problem distances to gain a better understanding of the learned patterns.

Noise directly affects $f(x)_{norm}$ and $f(x)_{norm-max}$. For $|SE|_{norm}$ and $|M|_{norm}$, the effects are indirect because the noise is not applied to the values of the metrics themselves. However, there may well be effects on computational or communication overhead when random noise is applied to the values of the objective functions. These effects can be either positive or negative. By applying Gaussian noise, the function values are sometimes significantly higher or lower than those obtained without noise. For comparability, the original bounds were used for scaling. This leads to values outside the interval [0,1]. These deviations were small for all performance metrics except $f(x)_{norm}$. Here, a repeated min-max scaling was performed on the entire training data set.

		$f(x)_{norm}$		$f(x)_{norm-max}$		$\lvert SE\rvert_{norm}$		$\lvert M\rvert_{norm}$	
		MSE	R^2	MSE	R^2	MSE	R^2	MSE	R^2
COHDA	**training data**	0.0167	0.8600	0.0006	0.9852	0.1092	0.7789	0.0304	0.7691
	noiseless evaluation data	0.0179	0.8530	0.0005	0.9879	0.0092	0.7756	0.0112	0.6705
	noisy evaluation data	0.0218	0.7461	0.0020	0.9606	0.91105	0.7381	0.1974	0.7369
IDICE	**training data**	0.0173	0.7727	0.0011	0.9595	0.0108	0.7738	0.0005	0.9871
	noiseless evaluation data	0.0989	0.3419	0.0011	0.9565	0.0153	0.6142	0.0005	0.9888
	noisy evaluation data	0.0138	0.7713	0.0014	0.9510	0.0099	0.8541	0.0012	0.9713

Tab. 6.9.: MSE and R^2 scores for the random forest regressor after retraining with partly noisy training data

Table 6.9 shows the results for the newly trained random forest regressors. Both MSE and R^2 for the noiseless evaluation data are very similar to the original models for both heuristics and all performance measures (cf. Table 6.6). The only exception is $f(x)_{norm}$ for IDICE, where the scores for the noiseless evaluation data are strongly degraded in comparison. A bias in the noiseless scoring data could be responsible for this. Although a uniform distribution of the problem instances was ensured during the splitting, this is not the case for the CTs. However, for $f(x)_{norm}$ in IDICE, the R^2 score actually improved for the training data. The improvement could be due to the fact that the noisy data is easier to predict, or that overfitting is avoided or reduced by its integration. For the noisy evaluation data, the scores are often slightly worse than for the noiseless data. In general, the differences are small. In addition, the difference in the MSE is usually larger, as this measure is more sensitive to outliers. However, the R^2 in these cases tends to show similarly good values as it does for the noiseless data.

In some cases, the predictions for the noisy data are even slightly better, namely for COHDA and $|M|_{norm}$, and for IDICE and $|SE|_{norm}$. For these two cases, the values for R^2 have been greatly improved by training on the partially noisy data. The lower values for R^2 with the noisy data here still roughly correspond to the values obtained with the standard models trained without noisy data (cf. table 6.6). In these cases, the noisy data itself seems to be more predictable. As a result, the score for the training data, which now contains noisy data, also increases. Moreover, these are performance metrics that are only indirectly affected by the noise application.

Overall, the models trained using partly noisy data tend to perform at least as well as those trained without it. The prediction quality of the noisy evaluation data is generally good. Only for IDICE and $f(x)_{norm}$ the prediction quality of the noiseless evaluation data decreases significantly.

After examining the general performance of the newly trained models and the predictability of the noisy data, we now take a detailed look at specific problem instances. Figures 6.20 and 6.21 show the predictions of the new models for noiseless and noisy evaluation data for the two problem instances *Drop Wave 1* and *Weierstrass 1* for COHDA and IDICE. The first impression shows: The solution quality of the two problem instances is usually worse with noisy data. Furthermore, the computational and communication effort for *Drop Wave 1* tends to decrease with noisy data. For *Weierstrass 1* it tends to increase.

The differences between the performance of CTVs tend to follow similar patterns for noiseless and noisy data. However, with respect to the solution quality, the variations are usually less pronounced with noise. But there is an exception with *Drop Wave 1* and IDICE. Here, for some CTVs with mode increase without noise, $f(x)$ goes down, which means increased solution quality, while with noise the values go up, which means decreased solution quality. Thus, for this problem instance, such CTVs seem to suffer more from noise.

Figure 6.20 shows that the predictions for COHDA for both noisy and noiseless evaluation data are very close to the mean values of the respective evaluation data. Compared to the previous models, where the predictions were closest to the means of the training data, this is actually an improvement. This implies that some overfitting occurred earlier, although similarly good results were obtained for MSE and R^2. The predictions for IDICE in fig. 6.21 show a similar picture. *Drop Wave 1* is an example that shows that the noiseless data for $f(x)_{norm}$ are more scattered and therefore more difficult to predict than the noisy data. However, the pattern is actually reproduced more accurately compared to the models trained without noise. For *Weierstrass 1*, on the other hand, both noisy and noiseless predictions for $f(x)_{norm}$ are excellent. For $f(x)_{norm-max}$, however, the model is inaccurate. Here, the predicted value is almost always identical, although there are differences in the actual data points. For $|SE|_{norm}$ and $|M|_{norm}$, both heuristics approximate the patterns very well.

Intermediate results

The following results were obtained by examining the predictive ability of ML models trained on a mixture of noiseless and noisy data:

- Data obtained from optimization runs with moderate Gaussian noise still follow similar patterns as their noiseless counterparts.
- ML models trained on a mixture of noisy and noiseless data can typically predict the noiseless data just as well or even better.
- Predictions for noisy data are usually about as accurate as for noiseless data.
- Training with a proportion of moderately noisy data reduces the risk of overfitting

These results support the general suitability of the approach for a future transfer to problems in the real world.

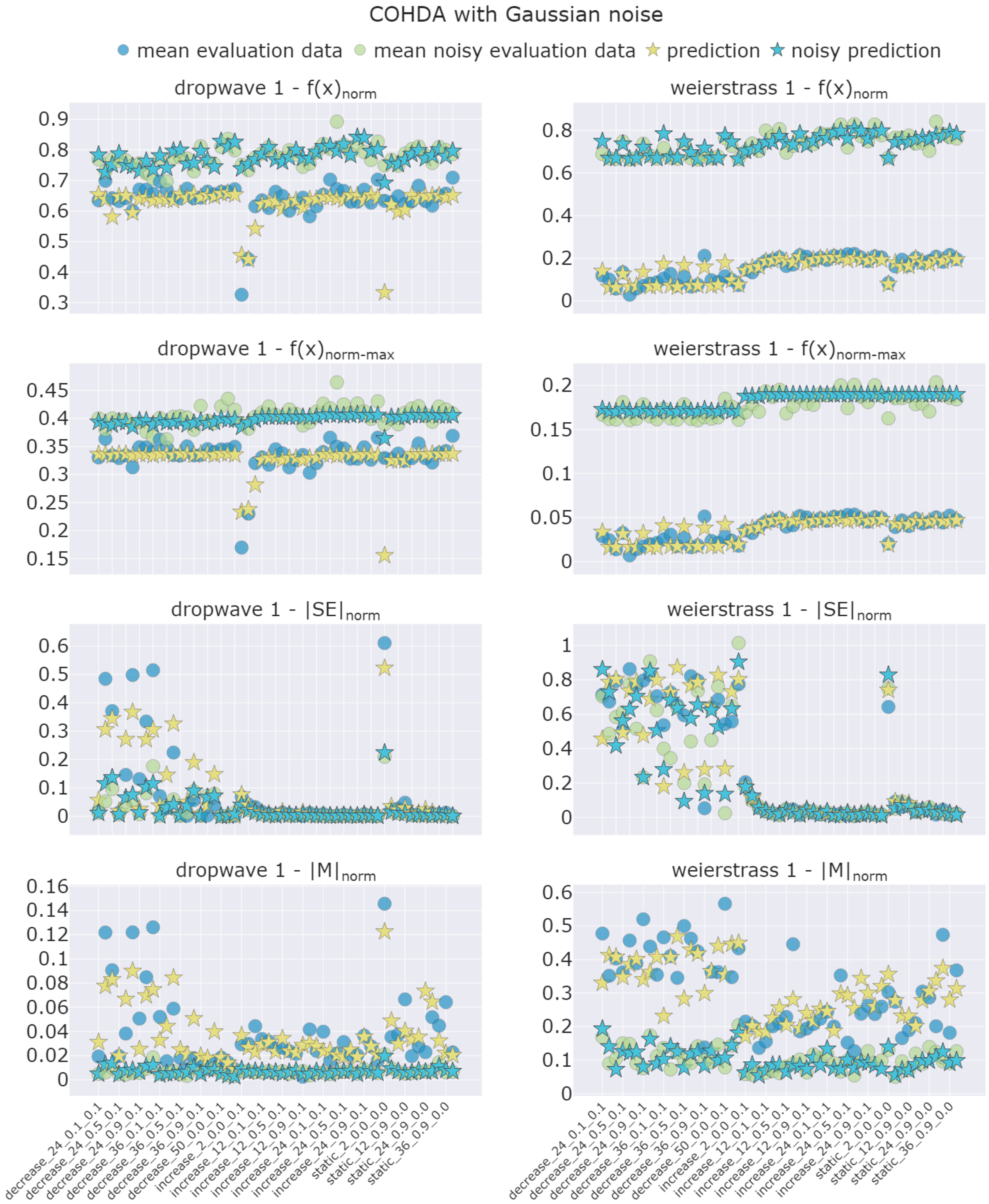


Fig. 6.20.: Predictions for noiseless and noisy evaluation data after retraining with a mixed set of training data. The corresponding metric is always plotted on the y-axis. The CTVs are plotted on the x-axis and are encoded in the form $mode\_k\_p\_\alpha$.

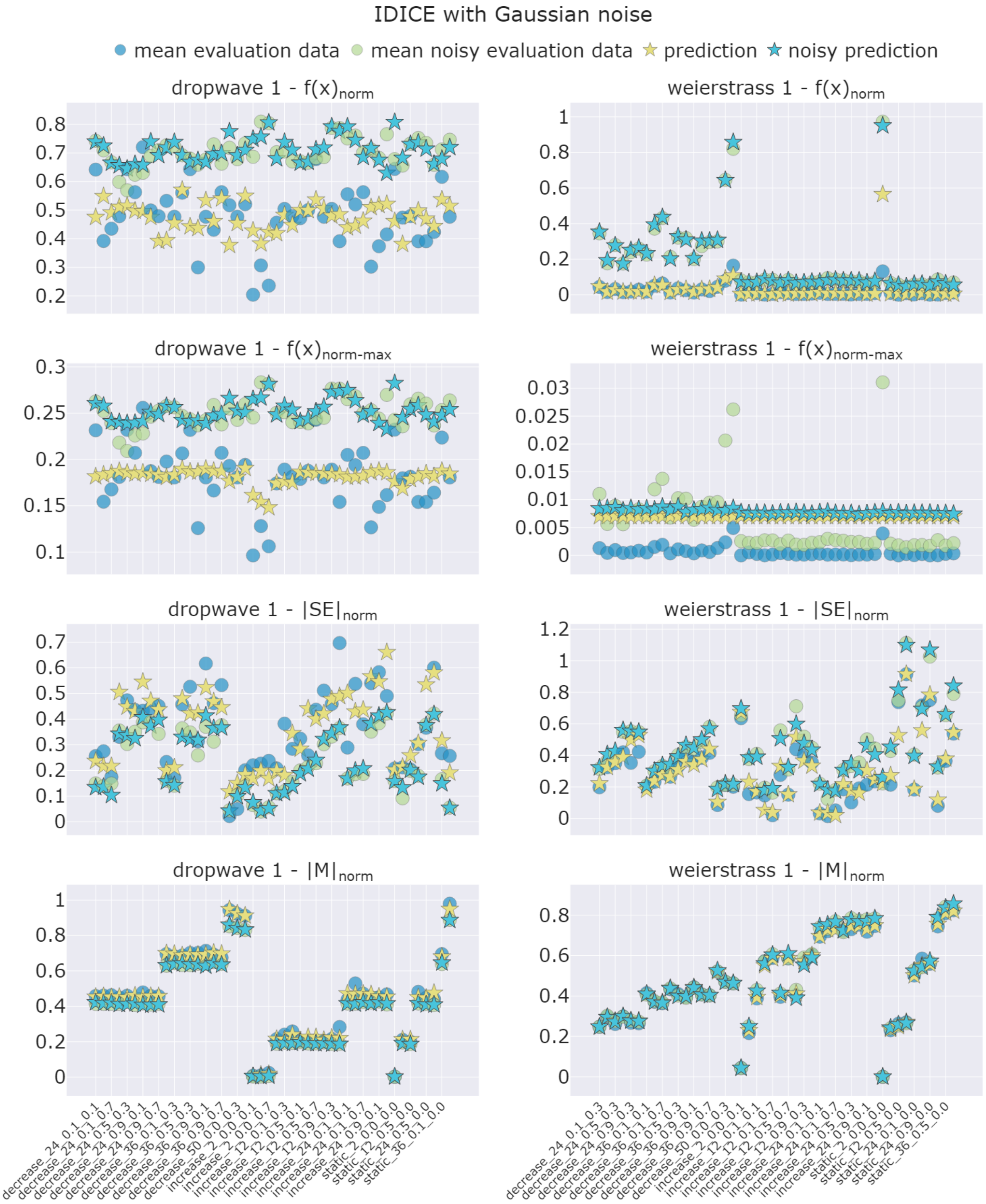


Fig. 6.21.: Predictions for noiseless and noisy evaluation data after retraining with a mixed set of training data. The corresponding metric is always plotted on the y-axis. The CTVs are plotted on the x-axis and are encoded in the form $mode\_k\_p\_\alpha$.

Part V

Conclusion

In this final part, the results and conclusions of this thesis are summarized and the multifaceted possibilities for further research are highlighted.

7

Conclusion

Power system operation involves a wide range of complex optimization problems, requiring the coordination of power generation and consumption across a large number of distributed units. When applying parallel optimization heuristics to such spatially distributed problems, the communication topology, which specifies the communication paths between distributed solvers, is an important hyperparameter that affects algorithm performance in several aspects, including solution quality, computational cost, and communication overhead. In this thesis, the effects of run-time adaptation of the Communication Topology (CT) and the relevance of the problem characteristics in this process have been investigated. Eventually, the idea of a causal relationship between these two aspects and the algorithm's performance was assessed. For this purpose, such an adaptation of the CT had to be modeled and a method for capturing problem properties in distributed systems developed. In the following, the results of the investigations and the research contribution are summarized in section 7.1. Finally, an overview of possible extensions and research topics for future work is given in section 7.2.

7.1 Results and research contributions

The work was divided into several parts, each dealing with one of the main aspects. Part II was devoted to Fitness Landscape Analysis (FLA), which serves to determine problem characteristics. Part III dealt with distributed optimization algorithms and in particular the role of communication topologies and the modeling of topology adaptation strategies at runtime. Finally, part IV combined both aspects to investigate the relationship between problem characteristics, Communication Topology Variant (CTV)s, and algorithm performance.

An overview of the main research contributions is provided below, before a more detailed explanation of the approach and the resulting contributions for each part follows in the remainder of the section.

Main contributions:

Method for distributed computation of fitness landscape metrics in spatially distributed systems.	RQ 2	part II
Modeling the adaptation of the communication topology for distributed optimization heuristics at runtime.	RQ 1	part III
IDICE: Island Model Differential coevolution as a combination of distributed evolutionary algorithm approaches to obtain an optimization heuristic that operates in a spatially distributed manner through network communication and with decision set decomposition.		part III
Investigation of predictability of algorithm performance based on the FLA metrics and CTV parameters.	RQ 3	part IV

7.1.1 Part II: Distributed Fitness Landscape Analysis

Part II focused on the Fitness Landscape Analysis (FLA). In chapter 2, the basics of FLA were explained by first introducing different features of fitness landscapes that are associated with the difficulty of optimization problems (section 2.1). Then, in section 2.2, concrete techniques and metrics that can be used to quantify these features were presented. Next, the concept of dynamic and coupled fitness landscapes was introduced in section 2.3. These fitness landscapes are characterized by the fact that the overall search space is divided into several subsearch spaces, and the selection of decision variables in one subsearch space influences the fitness landscape of the other subsearch spaces. This is the case, for example, in energy optimization problems where several power plants have to reach a common target power value. If one plant chooses a certain power value, the suitability of the other plants' power values may change with respect to the target power.

A new approach has been developed and presented in chapter 3, as there are no methods available to analyze such dynamic and coupled fitness landscapes in spatially distributed systems. This approach is based on a combination of an initial local sampling of all subsearch spaces, an exchange of these samples, and a subsequent recombination of the transmitted samples and their use for local fitness landscape analyses for each subsearch space. In the final step, the results from all subsearch spaces are combined. This basic principle was applied to the different types of FLA techniques in the following sections 3.3 - 3.5. Parameter tuning was performed for the different parameters such as number of samples and sampling techniques. A parameter selection considering the trade-off between computational effort and expressiveness of the FLA metrics was conducted by comparing the results with centrally computed FLA metrics (see Appendix appendix B.1). Finally, a correlation analysis for the distributed set of FLA metrics was performed in section 3.7.

The main findings of part II can be summarized as follows:

- The sampling-based method for distributed FLA provides a reasonable trade-off between computational and communication effort and the expressiveness of the computed features.
- For the same problem instances, the absolute values of the centralized and decentralized FLA metrics differ. However, for most metrics, the decentralized calculation comes close to the centrally calculated reference values. In addition, the relationship for a given FLA metric is usually similar across problem instances (e.g., higher for problem x than for problem y).

7.1.2 Part III: Communication Topologies for distributed Optimization Algorithms

Part III focused on the importance of exchange topologies for distributed optimization algorithms. In chapter 4, the basics of distributed optimization algorithms were presented. A special focus was put on the information sharing cooperation mechanism. This is the mechanism that defines all aspects of communication between distributed solvers, including the communication topology. The two optimization heuristics used in this work, COHDA and IDICE, were also introduced. IDICE is the result of combining different concepts of distributed evolutionary algorithms, namely the island model and coevolution.

The main original contribution in part III was made in chapter 5. The modeling of the CT adaptation was inspired by the cooling process of simulated annealing. The adaptation process is specified by a so-called topology schedule, which contains the number of edges that the CT should have at different steps of the optimization process. Edges can be removed or added during the optimization, and the speed of the adaptation, i.e. the number of modified edges, can be varied. This adaptation strategy is combined with a slightly modified Watts-Strogatz small-world graph as initial topology to form so-called communication topology variants (CTVs). Many combinations of initial topology and topology adaptation strategies can be created with just a few parameters

The main findings of part III can be summarized as follows:

- A preliminary study in section 5.3 showed for exemplary benchmark problems that topology variants affect all performance dimensions and that the effects differ depending on the optimization heuristic and problem instance.
- It was demonstrated that adapting the CT at runtime can improve the algorithm's performance.
- The choice of the best topology depends not only on the heuristic and the problem instance, but also on the prioritization of the performance dimensions. There is a natural trade-off between these dimensions.
- Since there is both a global view of the optimization process and a local view of individual optimizers, it is not trivial which topology leads to exploratory and which to exploitative behavior.

7.1.3 Part IV: Learning Optimal Topology Variants

In part IV the artifacts from the previous two parts were incorporated by investigating the relationship between the characteristics of the distributed optimization problems and the effects of the CTVs. The experimental setup that was employed has been explained in section 6.1. This included the considered problem instances, the set of distributed FLA metrics remaining after the correlation analysis of Section section 3.7, and the parameter setup for the CTVs. The first part of the evaluation was performed in Section section 6.2. There, the analysis of correlation coefficients and permutation importance scores was used to investigate the relevance of the FLA metrics and the parameters of the CTVs for predicting the performance metrics. All parameters were relevant to the prediction of the performance metrics, but the importance varied greatly depending on the optimization heuristic and the specific performance metric.

The second part of the evaluation in section 6.3 investigated the predictive capability of ML models. First, an appropriate ML model type was selected in section 6.3.1. Random forest regressors were chosen. These showed low error rates for both optimization heuristics and all performance metrics. Then, in section 6.3.2, the patterns learned by the models were examined using exemplary problem instances. The predictions were clearly dependent on the problem instance as well as on the CTV. This further confirmed the learnable relationship between these two aspects and the algorithm performance. In section 6.3.3, the trained models were used to select appropriate topology variants for different prioritizations of the performance dimensions. This demonstrated the feasibility of the overall concept. Afterwards, the tolerance to noisy data was investigated to make a first step towards transferability to real world problems.

The main findings of part IV can be summarized as follows:

- Distributedly computed FLA metrics and CTV parameters are important factors in predicting performance metrics, although the importance of individual metrics varies depending on the heuristic and performance metric.
- Distributedly computed FLA metrics are suitable for problem-specific parameter tuning and control for distributed optimization heuristics.
- An adequate CTV can be selected tailored to the problem and a prioritization of performance dimensions.
- The assumption of a causal relationship could be further supported.
- The observed patterns persisted even with moderate noise and were still learnable by ML models.

7.2 Future work

In preparing this thesis, it was necessary to establish a fixed scope. This results in many starting points for subsequent research topics. On the one hand, this concerns the expansion of individual aspects in the overall concept developed. Some of them have already been mentioned in this thesis. On the other hand, the logical next step is to transfer the concept from the theoretical problem instances to real-world optimization problems in power systems, which have formed the motivating context. In the following, the individual research topics are explained in more detail.

7.2.1 Extension of distributed FLA

The method for sampling-based distributed FLA developed in this thesis provides opportunities for future work.

Extension to more landscape features and techniques:

The selection of FLA features and methods used in this work was done systematically. The goal was to cover a wide range of different properties, but at the same time to take advantage of synergies in the computations. However, the set is by no means complete. Investigating further concepts of FLA for their suitability for a distributed computation approach and, if possible, developing adapted concepts would be an interesting prospect. These might include fitness clouds [Van+04], which illustrate evolvability with respect to a particular search operator, or local optima networks (LONs) [Och+14], a graph-based model for representing landscape structure in terms of local optima.

Enhanced sampling techniques:

The developed method for distributed FLA includes the step of *local FLA*, which involves the recombination of all samples received by an agent from other agents. Until now, classical methods such as Halton Sampling or Latin Hypercube were used, where the samples of other agents were treated as categorical and thus only the transmitted values could be used. The fact that relations between the decision variables of other agents are not taken into account was already pointed out in the corresponding chapter. This was unproblematic for the problem instances used in this work. However, when applied to real-world scenarios, this will most likely not be the case. Therefore, a first step towards transferability would be the development of further methods for recombining the samples, especially taking into account interdependencies between decision variables. In addition, other aspects, such as the best possible coverage of the space of possible local fitness landscapes, could be pursued more thoroughly.

Investigation of problem classes

The FLA is a preparatory step for the actual optimization. Therefore, as already mentioned in the corresponding chapter, suitable CTVs should be selected not only for a specific problem instance, but for a problem class. An essential research question is consequently how such classes can be reasonably assigned, and how much a problem has to change in order to leave a class and thus require a new selection of CTVs. In the motivating use cases from energy systems, this could be the case if there is a sufficiently large change in the target schedule, or if DERs are joining or leaving (and thus changing the number of decision variables of the overall problem).

7.2.2 Refining communication topology adaptation strategies

The topology adaptation model provides also an area for further research.

Adaptive modes:

So far, the *static*, *decrease*, and *increase* modes have been used, in which either no topology adaptation takes place or a largely deterministic adaptation is performed. The number of edges is adapted according to a deterministic policy and is either always decreased or always increased after a certain number of local optimizations. It would be worth investigating the use of an adaptive mode that, for example, reacts to the progress of the optimization and can reactively change the direction of the edge adaptation. The design of such a mode is not trivial, since exploration and exploitation in parallel optimization heuristics with decision set decomposition can be considered on both global and local levels, and different effects can be attributed to the exchange topologies depending on the perspective (see section 5.2.2 for more explanation).

Purposeful selection of edges:

Both the edges in the initial CT and the edges added or removed in the *increase* and *decrease* modes were randomly selected. A number of different criteria could be used to select the edges.

Considering **graph properties** would be one possibility. Selected aspects of graph theory could be optimized in both the initial and intermediate topologies. One example is the second largest eigenvalue (SLE) of the graph Laplacian (also known as algebraic connectivity or Fiedler eigenvalue), which has been shown to be an important factor in quantifying the convergence speed of consensus algorithms [OFM07]. Other examples would be the optimization of average shortest path lengths or average clustering coefficients.

The **algorithmic progress** of individual agents could also be a criterion for edge adaptation at runtime. Stagnating agents can get new impulses from new or different neighbors. Such an approach is also used in some forms of Particle Swarm Optimization (PSO).

Another edge selection criterion could be the **contribution potential** of individual agents. For example, in the motivating energy use case, agents that have greater flexibility may also have a greater contribution to the achievement of the overall goal. Integrating such agents more communicatively (through more edges) might be advantageous.

When applied to real-world problems, the underlying **physical communication infrastructure** may also be an important aspect. For instance, if some agents can only be reached via a limited communication link, it is probably not ideal to send a large number of messages to these agents. Accordingly, they should be connected with only a few edges in the initial and intermediate exchange topologies.

7.2.3 Advancement of algorithmic aspects

Further possibilities for research and development arise from the algorithmic aspects. These are related to the developed overall concept, the assessment of performance and to the optimization heuristic IDICE.

Extended application of the overall concept:

So far, two optimization heuristics have been used to test the overall concept of distributed FLA and the corresponding selection of an appropriate topology variant. The evaluation showed that the effects of FLA properties and topology variants can be very different for the two heuristics. Therefore, a next step would be to explore the concept with **additional parallel optimization heuristics** that can handle both decision set decomposition and spatial distribution of solvers.

Furthermore, some steps of the overall concept, namely the selection of topology variants and the adaptation of the topology at runtime, have so far been performed centrally, in the controller. It would be desirable to investigate the extent to which these aspects can also be **performed in a decentralized manner**. For example, agents could decide locally or bilaterally about the addition or removal of edges. But this would require more knowledge about the overall system at the local level.

However, it is also possible to **further develop the learning component** in a centralized way. First, the methodology has not been tested for unknown problems. This would be a logical next step, probably requiring a revision of ML models. Moreover, there may be a need for additional FLA metrics for better prediction quality. A more drastic transformation from offline to online learning would also be conceivable. For instance, reinforcement learning could be used to learn optimal topology variants.

Advancement of the performance assessment:

So far, **uncertainty** has not been a part of the prediction of performance metrics. For different CTVs within a problem instance, the variance of the results can vary significantly. Therefore, it would be reasonable to consider this uncertainty when selecting an appropriate topology variant. For example, the variance or the standard deviation could be predicted for each of the topology variants.

Another highly relevant extension would be to **expand the performance dimensions**. An example would be the consideration of robustness with respect to communication impairments or component failures. Such additional performance dimensions would first have to be assessed, and then the effects of CTVs on them would have to be determined. Improving performance dimensions such as robustness and resilience through an appropriate selection of CTVs would then also contribute to strengthening the self-healing capabilities of the system.

In addition, **assessing the relevance** of CTVs for solution quality could be improved. The metric $f(x)_{norm-max}$ is already an indicator for this, but might lead to an underestimation of the relevance. One possibility to implement a reasonable alternative could be the use of domain knowledge.

Further development and evaluation of IDICE:

The Island Model Differential Co-Evolution (IDICE) was developed to combine the ability to deal with decision set decomposition (coevolution) and to work in a spatially distributed system where solvers communicate via a CT (island model). So far, however, the heuristic has only been used to evaluate the CTVs. In the future, the heuristic will be compared with other centralized and decentralized state-of-the-art optimization heuristics in all performance dimensions. The scalability will also be explicitly investigated and compared. Finally, it is likely that the heuristic itself has room for improvement.

7.2.4 Transfer to real-world energy system optimization problems

The main obstacle in the transfer of the overall concept to energy system optimization problems, where the supply and consumption of many distributed plants must be coordinated, is the transfer of the FLA. In the motivating real-world problems, the subsearch spaces that make up the total search space are the flexibilities of the individual plants. These can be modeled in a wide variety of ways. These range from simple sets of feasible schedules to complex representations that reflect the technical degrees of freedom and constraints of the units in detail and unfold a multifaceted space of feasible schedules, e.g., [Tie+22], [BRS11].

In order to capture the characteristics of the problem, the concepts of FLA have to be transferred to flexibility representations. This requires sampling methods that are applicable to the particular flexibility model and that sample the space of possibilities sufficiently well. This is likely to vary in difficulty depending on the type of system. For example, battery storage systems can be extremely flexible. Therefore, they offer a very wide range of possible schedules. There is also a temporal component, as coordinating generation and consumption often involves more than one time step. In addition, the flexibility that is used by a plant in one time step often affects the flexibility that is available in other time steps. Ideally, a method for conducting FLA would be developed for scenarios with multiple asset types, multiple flexibility models, and extended timeframes. The applicability of the calculated metrics would have to be demonstrated, for example, through prediction of algorithm performance or selection of algorithms or algorithm parameters on their basis.

In an actual field test, the system would also have to cope with limited observability and measurement errors. Investigating the predictive ability of ML models in the presence of noise was a first step in this direction. However, more extensive investigations and possibly the extension of the overall system by self-healing mechanisms would be necessary. Under these assumptions, the presented approach could be applied in real applications. It could contribute to making distributed control systems smarter by improving their self-optimization capabilities.

Acronyms

AC Autonomic Computing.

COHDA Combinatorial Optimization Heuristic for Distributed Agents.

CT Communication Topology.

CTV Communication Topology Variant.

DE Differential Evolution.

DER Distributed Energy Resource.

DERs Distributed Energy Resources.

EA Evolutionary Algorithm.

EAs Evolutionary Algorithms.

FL Fitness Landscape.

FLA Fitness Landscape Analysis.

FLM Fitness Landscape Metric.

FSD-FLA Fitness and State Distribution based Fitness Landscape Analysis.

GA Genetic Algorithm.

IDICE Island Model Differential Co-Evolution.

ISCM Information-Sharing Cooperation Mechanism.

LDS Low Discrepancy Sequence.

LHS Latin Hypercube.

MAS Multi-Agent Aystems.

MAS Multi-Agent System.

ML Machine Learning.

MLP Multi-layer Perceptron.

OC Organic Computing.

PSO Particle Swarm Optimization.

RMSE Root Mean Square Error.

SA Simulated Annealing.

SA-FLA Sensitivity Analysis based Fitness Landscape Analysis.

SE-FLA Structure Exploring Fitness Landscape Analysis.

SHGO Simplicial Homology Global Optimization.

ToVarO Topology Variant Optimization.

List of Symbols

$f(x)_{norm}$ final value of the objective function $f(x)$ after an optimization run, normalized by min-max scaling using only values obtained by optimization runs.

$f(x)_{norm-max}$ final value of the objective function $f(x)$ after an optimization run, normalized by min-max scaling using a value obtained by a maximization run as upper bound.

$|M|_{norm}$ total number of messages sent by all agents during an optimization run, normalized by min-max scaling.

$|SE|_{norm}$ total number of local optimizations ("decide" steps) of all agents during the optimization run, normalized by min-max scaling.

$\mu_2(y)$ Fitness Variance.

$\mu(||d||)$ Mean state distance.

FEM_{micro} First Entropic Measure (for ruggedness) on micro scale.

FEM_{macro} First Entropic Measure (for ruggedness) on macro scale.

SEM_{micro} Second Entropic Measure (for smoothness) on micro scale.

SEM_{macro} Second Entropic Measure (for smoothness) on macro scale.

PIC Partial Information Content (for modality).

FEM First Entropic Measure (for ruggedness).

SEM Second Entropic Measure (for smoothness).

PIC_{micro} Partial Information Content (for modality) on micro scale.

PIC_{macro} Partial Information Content (for modality) on macro scale.

$\mu_2(||d||)$ State variance.

$\mathcal{N}_{FLA}$ notion of neighborhood in the context of FLA.

Υ_i best known solution candidate.

Ω_i (believed) current system state (choices of other agents).

v_{cv} Coefficient of Variation in Variable Sensitivity.

λ_j version counter for variable choices of agents in COHDA.

v_{inter} Degree of Variable Interaction.

κ_i working memory of an agent that it sends to its neighbors.

Bibliography

[Al-15] Ali R. Al-Roomi. *Unconstrained Single-Objective Benchmark Functions Repository*. Halifax, Nova Scotia, Canada, 2015 (cit. on pp. 212, 217).

[AR22] Ivano Azzini and Rossana Rosati. „A function dataset for benchmarking in sensitivity analysis“. In: *Data in brief* 42 (2022), p. 108071 (cit. on pp. 61, 62).

[Arn+13] Ignacio Arnaldo, Iván Contreras, David Millán-Ruiz, J Ignacio Hidalgo, and Natalio Krasnogor. „Matching island topologies to problem structure in parallel evolutionary algorithms“. In: *Soft Computing* 17 (2013), pp. 1209–1225 (cit. on pp. 14, 99).

[BB12] James Bergstra and Yoshua Bengio. „Random search for hyper-parameter optimization.“ In: *Journal of machine learning research* 13.2 (2012) (cit. on p. 154).

[BF12] Hans-Georg Beyer and Steffen Finck. „HappyCat–A simple function class where well-known direct search algorithms do fail“. In: *International conference on parallel problem solving from nature*. Springer. 2012, pp. 367–376 (cit. on p. 217).

[BH00] Meriema Belaidouni and Jin-Kao Hao. „An analysis of the configuration space of the maximal constraint satisfaction problem“. In: *International Conference on Parallel Problem Solving from Nature*. Springer. 2000, pp. 49–58 (cit. on p. 33).

[BL17] Joerg Bremer and Sebastian Lehnhoff. „An Agent-based Approach to Decentralized Global Optimization-Adapting COHDA to Coordinate Descent“. In: *International Conference on Agents and Artificial Intelligence*. Vol. 2. SCITEPRESS. 2017, pp. 129–136 (cit. on p. 87).

[BL19] Jörg Bremer and Sebastian Lehnhoff. „The Effect of Laziness on Agents for Large Scale Global Optimization“. In: *International Conference on Agents and Artificial Intelligence*. Springer. 2019, pp. 317–337 (cit. on pp. 89, 223).

[BR03] Christian Blum and Andrea Roli. „Metaheuristics in combinatorial optimization: Overview and conceptual comparison“. In: *ACM computing surveys (CSUR)* 35.3 (2003), pp. 268–308 (cit. on p. 108).

[Bre01] Leo Breiman. „Random forests“. In: *Machine learning* 45.1 (2001), pp. 5–32 (cit. on p. 144).

[BRS11] Jörg Bremer, Barbara Rapp, and Michael Sonnenschein. „Encoding distributed search spaces for virtual power plants“. In: *2011 IEEE Symposium on Computational Intelligence Applications In Smart Grid (CIASG)*. IEEE. 2011, pp. 1–8 (cit. on pp. 5, 183).

[Can99] Erick Cantú-Paz. „Topologies, migration rates, and multi-population parallel genetic algorithms“. In: *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 1*. 1999, pp. 91–98 (cit. on p. 15).

[ČLM13] Matej Črepinšek, Shih-Hsi Liu, and Marjan Mernik. „Exploration and exploitation in evolutionary algorithms: A survey“. In: *ACM computing surveys (CSUR)* 45.3 (2013), pp. 1–33 (cit. on pp. 13, 14, 105, 108).

[CM20] Gianfranco Chicco and Andrea Mazza. „Metaheuristic optimization of power and energy systems: underlying principles and main issues of the ‘rush to heuristics’“. In: *energies* 13.19 (2020), p. 5097 (cit. on p. 4).

[Cra19] Teodor Crainic. „Parallel metaheuristics and cooperative search“. In: *Handbook of Metaheuristics*. Springer, 2019, pp. 419–451 (cit. on pp. 3, 81–83, 88, 101).

[CT07] Teodor Gabriel Crainic and Michel Toulouse. „Explicit and emergent cooperation schemes for search algorithms“. In: *International Conference on Learning and Intelligent Optimization*. Springer. 2007, pp. 95–109 (cit. on pp. 3, 83–85, 101).

[Dan51] George B Dantzig. „Maximization of a linear function of variables subject to linear inequalities“. In: *Activity analysis of production and allocation* 13 (1951), pp. 339–347 (cit. on p. 12).

[Dav90] Yuval Davidor. „Epistasis variance: Suitability of a representation to genetic algorithms“. In: *Complex Systems* 4.4 (1990), pp. 369–383 (cit. on p. 32).

[De 07] Edwin D De Jong. „Objective fitness correlation“. In: *Proceedings of the 9th annual conference on Genetic and evolutionary computation*. 2007, pp. 440–447 (cit. on pp. 51, 52).

[Dör+19] Florian Dörfler, Saverio Bolognani, John W Simpson-Porco, and Sergio Grammatico. „Distributed control and optimization for autonomous power grids“. In: *2019 18th European Control Conference (ECC)*. IEEE. 2019, pp. 2436–2453 (cit. on p. 3).

[EHM99] Ágoston E Eiben, Robert Hinterding, and Zbigniew Michalewicz. „Parameter control in evolutionary algorithms“. In: *IEEE Transactions on evolutionary computation* 3.2 (1999), pp. 124–141 (cit. on pp. 13, 14, 108).

[ESF18] Stefan C Endres, Carl Sandrock, and Walter W Focke. „A simplicial homology algorithm for Lipschitz optimisation“. In: *Journal of Global Optimization* 72.2 (2018), pp. 181–217 (cit. on pp. 87, 223).

[Fin+] Steffen Finck, Nikolaus Hansen, Raymond Ros, and Anne Auger. *Real-parameter black-box optimization benchmarking 2010: Presentation of the noisy functions*. Tech. rep. Citeseer (cit. on pp. 166, 218).

[FRP98] Cyril Fonlupt, Denis Robilliard, and Philippe Preux. „A bit-wise epistasis measure for binary search spaces“. In: *International Conference on Parallel Problem Solving from Nature*. Springer. 1998, pp. 47–56 (cit. on p. 32).

[Gie+19] Dolf Gielen, Francisco Boshell, Deger Saygin, et al. „The role of renewable energy in the global energy transformation“. In: *Energy Strategy Reviews* 24 (2019), pp. 38–50 (cit. on p. 3).

[GK01] J Garnier and L Kallel. „How to detect all maxima of a function“. In: *Theoretical aspects of evolutionary computing*. Springer, 2001, pp. 343–370 (cit. on p. 34).

[Gol89] David E Goldberg. „Genetic algorithms and walsh functions-partii: Deception and its analysis“. In: *Complex systems* 3 (1989), pp. 153–171 (cit. on p. 35).

[Gon+15] Yue-Jiao Gong, Wei-Neng Chen, Zhi-Hui Zhan, et al. „Distributed evolutionary algorithms and their models: A survey of the state-of-the-art“. In: *Applied Soft Computing* 34 (2015), pp. 286–300 (cit. on p. 91).

[HC09] Muhannad Hijaze and David Corne. „An investigation of topologies and migration schemes for asynchronous distributed evolutionary algorithms“. In: *2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)*. IEEE. 2009, pp. 636–641 (cit. on pp. 97, 98).

[HC11] Muhannad Hijaze and David Corne. „Distributed evolutionary algorithm topologies with adaptive migration schemes“. In: *2011 IEEE Congress of Evolutionary Computation (CEC)*. IEEE. 2011, pp. 608–615 (cit. on pp. 97, 98).

[He+07] Jun He, Colin Reeves, Carsten Witt, and Xin Yao. „A note on problem difficulty measures in black-box optimization: Classification, realizations and predictability“. In: *Evolutionary Computation* 15.4 (2007), pp. 435–443 (cit. on p. 31).

[Hea+21] Tim Head, Manoj Kumar, Holger Nahrstaedt, Gilles Louppe, and Iaroslav Shcherbatyi. *scikit-optimize/scikit-optimize*. Version v0.9.0. Oct. 2021 (cit. on p. 61).

[Hin14] Christian Hinrichs. „Selbstorganisierte Einsatzplanung dezentraler Akteure im smart grid“. PhD thesis. Universität Oldenburg, 2014 (cit. on pp. 5, 86, 89, 97, 223).

[HK05] Wim Hordijk and Stuart A Kauffman. „Correlation analysis of coupled fitness landscapes“. In: *Complexity* 10.6 (2005), pp. 41–49 (cit. on pp. 51, 52, 57).

[HLS13a] Christian Hinrichs, Sebastian Lehnhoff, and Michael Sonnenschein. „A decentralized heuristic for multiple-choice combinatorial optimization problems“. In: *Operations Research Proceedings 2012: Selected Papers of the International Annual Conference of the German Operations Research Society (GOR), Leibniz University of Hannover, Germany, September 5-7, 2012*. Springer. 2013, pp. 297–302 (cit. on p. 5).

[HLS13b] Christian Hinrichs, Sebastian Lehnhoff, and Michael Sonnenschein. „COHDA: A combinatorial optimization heuristic for distributed agents“. In: *International Conference on Agents and Artificial Intelligence*. Springer. 2013, pp. 23–39 (cit. on p. 89).

[HN20] Stefanie Holly and Astrid Nieße. „On the effects of communication topologies on the performance of distributed optimization heuristics in smart grids“. In: *INFORMATIK 2020: Back to the Future - 50. Jahrestagung der Gesellschaft für Informatik* (2020) (cit. on pp. 21, 79, 97, 106, 213).

[HN21a] Stefanie Holly and Astrid Nieße. „Distributed fitness landscape analysis for cooperative search with domain decomposition“. In: *IEEE Symposium Series on Computational Intelligence (IEEE SSCI 2021)*. in Press. IEEE. 2021 (cit. on pp. 4, 21, 22, 30, 97, 121, 129, 131).

[HN21b] Stefanie Holly and Astrid Nieße. „Dynamic communication topologies for distributed heuristics in energy system optimization algorithms“. In: *2021 16th Conference on Computer Science and Intelligence Systems (FedCSIS)*. IEEE. 2021, pp. 191–200 (cit. on pp. 4, 21, 30, 97, 121, 129, 131).

[Hol+20] Stefanie Holly, Astrid Nieße, Martin Tröschel, et al. „Flexibility management and provision of balancing services with battery-electric automated guided vehicles in the Hamburg container terminal Altenwerder“. In: *Energy Informatics* 3.1 (2020), pp. 1–20 (cit. on p. 4).

[HS17] Christian Hinrichs and Michael Sonnenschein. „A distributed combinatorial optimisation heuristic for the scheduling of energy resources represented by self-interested agents.“ In: *IJBIC* 10.2 (2017), pp. 69–78 (cit. on pp. 23, 86, 87).

[HS98] Wim Hordijk and Peter F Stadler. „Amplitude spectra of fitness landscapes“. In: *Advances in Complex Systems* 1.01 (1998), pp. 39–66 (cit. on p. 37).

[HU17] Jon Herman and Will Usher. „SALib: An open-source Python library for Sensitivity Analysis“. In: *The Journal of Open Source Software* 2.9 (Jan. 2017) (cit. on pp. 61, 227).

[IRB12] Dario Izzo, Marek Ruciński, and Francesco Biscani. „The generalized island model“. In: *Parallel Architectures and Bioinspired Algorithms*. Springer, 2012, pp. 151–169 (cit. on p. 91).

[IUH22] Takuya Iwanaga, William Usher, and Jonathan Herman. „Toward SALib 2.0: Advancing the accessibility and interpretability of global sensitivity analyses“. In: *Socio-Environmental Systems Modelling* 4 (May 2022), p. 18155 (cit. on pp. 61, 227).

[JB03] Nicholas R Jennings and Stefan Bussmann. „Agent-based control systems“. In: *IEEE control systems* 23.3 (2003), pp. 61–74 (cit. on p. 7).

[JTV99] Márk Jelasity, Boglárka Tóth, and Tamás Vinkó. „Characterizations of trajectory structure of fitness landscapes based on pairwise transition probabilities of solutions“. In: *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*. Vol. 1. IEEE. 1999, pp. 623–630 (cit. on p. 35).

[Kan+15] Abhilash Kantamneni, Laura E Brown, Gordon Parker, and Wayne W Weaver. „Survey of multi-agent systems for microgrid control“. In: *Engineering applications of artificial intelligence* 45 (2015), pp. 192–203 (cit. on p. 3).

[Kar84] Narendra Karmarkar. „A new polynomial-time algorithm for linear programming“. In: *Proceedings of the sixteenth annual ACM symposium on Theory of computing*. 1984, pp. 302–311 (cit. on p. 12).

[Kau89] Stuart A Kauffman. „Adaptation on rugged fitness landscapes“. In: *Lectures in the Science of Complexity* 1 (1989), pp. 527–618 (cit. on p. 36).

[KC03] Jeffrey O Kephart and David M Chess. „The vision of autonomic computing“. In: *Computer* 36.1 (2003), pp. 41–50 (cit. on pp. 9, 10).

[KHE14] Giorgos Karafotias, Mark Hoogendoorn, and Ágoston E Eiben. „Parameter control in evolutionary algorithms: Trends and challenges“. In: *IEEE Transactions on Evolutionary Computation* 19.2 (2014), pp. 167–187 (cit. on p. 14).

[KNR01] Leila Kallel, Bart Naudts, and Colin R Reeves. „Properties of fitness functions and search landscapes“. In: *Theoretical aspects of evolutionary computing*. Springer, 2001, pp. 175–206 (cit. on pp. 31, 34).

[Kra08] Oliver Kramer. *Self-adaptive heuristics for evolutionary computation*. Vol. 147. Springer, 2008 (cit. on p. 14).

[Lah+15] Nadia Lahrichi, Teodor Gabriel Crainic, Michel Gendreau, et al. „An integrative cooperative search framework for multi-decision-attribute combinatorial optimization: Application to the MDPVRP“. In: *European Journal of Operational Research* 246.2 (2015), pp. 400–412 (cit. on p. 85).

[LF07] David Lazer and Allan Friedman. „The network structure of exploration and exploitation“. In: *Administrative science quarterly* 52.4 (2007), pp. 667–694 (cit. on p. 15).

[Li+13] Xiaodong Li, Ke Tang, Mohammad N Omidvar, et al. „Benchmark functions for the CEC 2013 special session and competition on large-scale global optimization“. In: *gene* 7.33 (2013), p. 8 (cit. on pp. 32, 98, 212, 218).

[Lip91] Mark Lipsitch. „Adaptation on rugged landscapes generated by local interactions of neighboring genes“. In: *Proceedings of the Fourth International Conference on Genetic Algorithms, San Mateo, CA*. Citeseer. 1991 (cit. on pp. 37, 43).

[Liu+07] Wenguo Liu, Alan FT Winfield, Jin Sa, Jie Chen, and Lihua Dou. „Towards energy optimization: Emergent task allocation in a swarm of foraging robots“. In: *Adaptive behavior* 15.3 (2007), pp. 289–305 (cit. on p. 11).

[LK95] Bennett Levitan and Stuart Kauffman. „Adaptive walks with noisy fitness measurements“. In: *Molecular diversity* 1.1 (1995), pp. 53–68 (cit. on p. 33).

[LLA04] Kian Hsiang Low, Wee Kheng Leow, and Marcelo H Ang Jr. „Task allocation via self-organizing swarm coalitions in distributed mobile sensor network“. In: *AAAI*. Vol. 4. 2004, pp. 28–33 (cit. on p. 11).

[LM05] Manuel Laguna and Rafael Marti. „Experimental testing of advanced scatter search designs for global optimization of multimodal functions“. In: *Journal of Global Optimization* 33.2 (2005), pp. 235–255 (cit. on p. 217).

[Loe+22] Inga Loeser, Martin Braun, Christian Gruhl, et al. „The Vision of Self-Management in Cognitive Organic Power Distribution Systems“. In: *Energies* 15.3 (2022), p. 881 (cit. on p. 10).

[Lop+20] João Abel Peças Lopes, André Guimarães Madureira, Manuel Matos, et al. „The future of power systems: Challenges, trends, and upcoming paradigms“. In: *Wiley Interdisciplinary Reviews: Energy and Environment* 9.3 (2020), e368 (cit. on p. 4).

[LTZ12] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. „Isolation-based anomaly detection“. In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 6.1 (2012), pp. 1–39 (cit. on p. 125).

[LW06] Monte Lunacek and Darrell Whitley. „The dispersion metric and the CMA evolution strategy“. In: *Proceedings of the 8th annual conference on Genetic and evolutionary computation*. 2006, pp. 477–484 (cit. on pp. 35, 42).

[Mal+14] Katherine Mary Malan et al. „Characterising continuous optimisation problems for particle swarm optimisation performance prediction“. PhD thesis. University of Pretoria, 2014 (cit. on p. 35).

[Mal21] Katherine Mary Malan. „A survey of advances in landscape analysis for optimisation“. In: *Algorithms* 14.2 (2021), p. 40 (cit. on pp. 17, 29).

[ME09] Katherine M Malan and Andries P Engelbrecht. „Quantifying ruggedness of continuous landscapes using entropy“. In: *2009 IEEE Congress on evolutionary computation*. IEEE. 2009, pp. 1440–1447 (cit. on pp. 45, 46).

[ME13a] Katherine M Malan and Andries P Engelbrecht. „A survey of techniques for characterising fitness landscapes and some possible ways forward“. In: *Information Sciences* 241 (2013), pp. 148–163 (cit. on pp. 29, 31–37).

[ME13b] Katherine M Malan and Andries P Engelbrecht. „Ruggedness, funnels and gradients in fitness landscapes and the effect on PSO performance“. In: *2013 IEEE Congress on Evolutionary Computation*. IEEE. 2013, pp. 963–970 (cit. on pp. 35, 42, 70).

[ME13c] Katherine M Malan and Andries P Engelbrecht. „Steep gradients as a predictor of PSO failure“. In: *Proceedings of the 15th annual conference companion on Genetic and evolutionary computation*. 2013, pp. 9–10 (cit. on pp. 43, 47).

[ME14] Katherine M Malan and Andries P Engelbrecht. „A progressive random walk algorithm for sampling continuous fitness landscapes“. In: *2014 IEEE Congress on Evolutionary Computation (CEC)*. IEEE. 2014, pp. 2507–2514 (cit. on pp. 43, 44, 69).

[Mer04] Peter Merz. „Advanced fitness landscape analysis and the performance of memetic algorithms“. In: *Evolutionary Computation* 12.3 (2004), pp. 303–325 (cit. on p. 34).

[Mil67] Stanley Milgram. „The small world problem“. In: *Psychology today* 2.1 (1967), pp. 60–67 (cit. on p. 105).

[Mol+17] Daniel K Molzahn, Florian Dörfler, Henrik Sandberg, et al. „A survey of distributed optimization and control algorithms for electric power systems“. In: *IEEE Transactions on Smart Grid* 8.6 (2017), pp. 2941–2962 (cit. on pp. 3, 6, 80).

[MT17] Christian Müller-Schloer and Sven Tomforde. *Organic Computing-Technical Systems for Survival in the Real World*. Springer, 2017 (cit. on pp. 9, 10).

[Mül11] Daniel Müllner. „Modern hierarchical, agglomerative clustering algorithms“. In: *arXiv preprint arXiv:1109.2378* (2011) (cit. on pp. 72, 244).

[MY13] Jamil Momin and Xin-She Yang. „A literature survey of benchmark functions for global optimization problems“. In: *Int. Journal of Mathematical Modelling and Numerical Optimisation* 4.2 (2013), pp. 150–194 (cit. on pp. 98, 209, 212, 213, 217, 218).

[NTS13] Astrid Nieße, Martin Tröschel, and Michael Sonnenschein. „Designing Dependable and Sustainable Smart Grids – How to Apply Algorithm Engineering to Distributed Control in Power Systems“. In: *Environmental Modelling & Software* (2013) (cit. on p. 79).

[NW99] Mark EJ Newman and Duncan J Watts. „Renormalization group analysis of the small-world network model“. In: *Physics Letters A* 263.4-6 (1999), pp. 341–346 (cit. on p. 108).

[Och+08] Gabriela Ochoa, Marco Tomassini, Sebástien Vérel, and Christian Darabos. „A study of NK landscapes' basins and local optima networks". In: *Proceedings of the 10th annual conference on Genetic and evolutionary computation*. 2008, pp. 555–562 (cit. on p. 34).

[Och+14] Gabriela Ochoa, Sébastien Verel, Fabio Daolio, and Marco Tomassini. „Local optima networks: A new model of combinatorial fitness landscapes". In: *Recent advances in the theory and application of fitness landscapes* (2014), pp. 233–262 (cit. on p. 179).

[OFM07] Reza Olfati-Saber, J Alex Fax, and Richard M Murray. „Consensus and cooperation in networked multi-agent systems". In: *Proceedings of the IEEE* 95.1 (2007), pp. 215–233 (cit. on p. 181).

[PA12] Erik Pitzer and Michael Affenzeller. „A comprehensive survey on fitness landscape analysis". In: *Recent advances in intelligent engineering systems* (2012), pp. 161–191 (cit. on pp. 30–32, 34).

[Ped+11] Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, et al. „Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830 (cit. on pp. 125, 133, 144, 154).

[PP02] A Papoulis and SU Pillai. *Probability, Random Variables and Stochastic Processes Athanasios)*. 2002 (cit. on p. 167).

[PS22] Vagelis Plevris and German Solorzano. „A collection of 30 multidimensional functions for global optimization benchmarking". In: *Data* 7.4 (2022), p. 46 (cit. on pp. 209, 217–219).

[RAT18] Yara Rizk, Mariette Awad, and Edward W Tunstel. „Decision making in multiagent systems: A survey". In: *IEEE Transactions on Cognitive and Developmental Systems* 10.3 (2018), pp. 514–529 (cit. on p. 80).

[REA96] Helge Rosé, Werner Ebeling, and Torsten Asselmeyer. „The density of states—a measure of the difficulty of optimisation problems". In: *International Conference on Parallel Problem Solving from Nature*. Springer. 1996, pp. 208–217 (cit. on p. 33).

[RIB10] Marek Ruciński, Dario Izzo, and Francesco Biscani. „On the impact of the migration topology on the island model". In: *Parallel Computing* 36.10-11 (2010), pp. 555–571 (cit. on pp. 97, 101).

[Ric06] Hendrik Richter. „Evolutionary optimization in spatio–temporal fitness landscapes". In: *Parallel Problem Solving from Nature-PPSN IX*. Springer, 2006, pp. 1–10 (cit. on pp. 51, 52).

[Ric08] Hendrik Richter. „Coupled map lattices as spatio-temporal fitness functions: Landscape measures and evolutionary optimization". In: *Physica D: Nonlinear Phenomena* 237.2 (2008), pp. 167–186 (cit. on pp. 51, 52).

[Ric14] Hendrik Richter. „Codynamic fitness landscapes of coevolutionary minimal substrates". In: *2014 IEEE Congress on Evolutionary Computation (CEC)*. IEEE. 2014, pp. 2692–2699 (cit. on pp. 51, 52).

[RK10] Enda Ridge and Daniel Kudenko. „Tuning an algorithm using design of experiments". In: *Experimental methods for the analysis of optimization algorithms*. Springer, 2010, pp. 265–286 (cit. on p. 14).

[Roc97] Sophie Rochet. „Epistasis in genetic algorithms revisited“. In: *Information Sciences* 102.1-4 (1997), pp. 133–155 (cit. on p. 32).

[RS01] Christian M Reidys and Peter F Stadler. „Neutrality in fitness landscapes“. In: *Applied Mathematics and Computation* 117.2-3 (2001), pp. 321–350 (cit. on p. 37).

[Rus10] Stuart J Russell. *Artificial intelligence a modern approach*. Pearson Education, Inc., 2010 (cit. on p. 7).

[Sal+10] Andrea Saltelli, Paola Annoni, Ivano Azzini, et al. „Variance based sensitivity analysis of model output. Design and estimator for the total sensitivity index“. In: *Computer physics communications* 181.2 (2010), pp. 259–270 (cit. on pp. 38, 40).

[Sal02] Andrea Saltelli. „Making best use of model evaluations to compute sensitivity indices“. In: *Computer physics communications* 145.2 (2002), pp. 280–297 (cit. on p. 61).

[Sch81] Hans-Paul Schwefel. *Numerical optimization of computer models*. John Wiley & Sons, Inc., 1981 (cit. on p. 213).

[SGK05] Giovanna Di Marzo Serugendo, Marie-Pierre Gleizes, and Anthony Karageorgos. „Self-organization in multi-agent systems“. In: *The Knowledge engineering review* 20.2 (2005), pp. 165–189 (cit. on pp. 8, 10).

[SGK06] Giovanna Di Marzo Serugendo, Marie-Pierre Gleizes, and Anthony Karageorgos. „Self-organisation and emergence in MAS: An overview“. In: *Informatica* 30.1 (2006) (cit. on p. 8).

[SJ15] Meera Sanu and G Jeyakumar. „Empirical performance analysis of distributed differential evolution for varying migration topologies“. In: *International Journal of Applied Engineering Research* 10.5 (2015), pp. 11–919 (cit. on pp. 97, 99).

[SKH16] Yuan Sun, Michael Kirley, and Saman K Halgamuge. „Quantifying variable interactions in continuous optimization problems“. In: *IEEE Transactions on Evolutionary Computation* 21.2 (2016), pp. 249–264 (cit. on p. 32).

[Sob01] Ilya M Sobol. „Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates“. In: *Mathematics and computers in simulation* 55.1-3 (2001), pp. 271–280 (cit. on p. 38).

[Sob67] Il'ya Meerovich Sobol'. „On the distribution of points in a cube and the approximate evaluation of integrals“. In: *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki* 7.4 (1967), pp. 784–802 (cit. on p. 40).

[SP97] Rainer Storn and Kenneth Price. „Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces“. In: *Journal of global optimization* 11.4 (1997), pp. 341–359 (cit. on p. 92).

[SS04] Raj Subbu and Arthur C Sanderson. „Network-based distributed planning using coevolutionary agents: architecture and evaluation“. In: *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* 34.2 (2004), pp. 257–269 (cit. on pp. 85, 91, 94).

[Ste87] Michael Stein. „Large sample properties of simulations using Latin hypercube sampling“. In: *Technometrics* 29.2 (1987), pp. 143–151 (cit. on p. 40).

[Ste99] Soraya Rana Stevens. „Examining the role of local optima and schema processing in genetic search“. PhD thesis. Colorado State University, 1999 (cit. on p. 34).

[Str01] Steven H Strogatz. „Exploring complex networks“. In: *nature* 410.6825 (2001), pp. 268–276 (cit. on p. 97).

[Sun+14] Yuan Sun, Saman K Halgamuge, Michael Kirley, and Mario A Munoz. „On the selection of fitness landscape analysis metrics for continuous optimization problems“. In: *7th International Conference on Information and Automation for Sustainability*. IEEE. 2014, pp. 1–6 (cit. on pp. 31, 34).

[SZ18] Jialong Shi and Qingfu Zhang. „A new cooperative framework for parallel trajectory-based metaheuristics“. In: *Applied Soft Computing* 65 (2018), pp. 374–386 (cit. on pp. 97, 98).

[TAA20] Khadija Tazi, Fouad Mohamed Abbou, and Farid Abdi. „Multi-agent system for microgrids: design, optimization and performance“. In: *Artificial Intelligence Review* 53 (2020), pp. 1233–1292 (cit. on p. 3).

[Tal09] El-Ghazali Talbi. *Metaheuristics: From Design to Implementation*. Vol. 74. John Wiley & Sons, 2009 (cit. on pp. 3, 12, 13, 29, 81, 83, 84, 95, 110, 111).

[TÇT07] Eric Tatara, Ali Çınar, and Fouad Teymour. „Control of complex distributed systems with distributed intelligent agents“. In: *Journal of process control* 17.5 (2007), pp. 415–427 (cit. on p. 8).

[Tie+22] Paul Hendrik Tiemann, Marvin Nebel-Wenner, Stefanie Holly, et al. „Operational flexibility for multi-purpose usage of pooled battery storage systems“. In: *Energy Informatics* 5.1 (2022), pp. 1–13 (cit. on pp. 5, 183).

[Tom+11] Sven Tomforde, Holger Prothmann, Jürgen Branke, et al. „Observation and control of organic systems“. In: *Organic Computing—A Paradigm Shift for Complex Systems* (2011), pp. 325–338 (cit. on p. 9).

[Van+04] Leonardo Vanneschi, Manuel Clergue, Philippe Collard, Marco Tomassini, and Sébastien Vérel. „Fitness clouds and problem hardness in genetic programming“. In: *Genetic and Evolutionary Computation–GECCO 2004: Genetic and Evolutionary Computation Conference, Seattle, WA, USA, June 26-30, 2004. Proceedings, Part II*. Springer. 2004, pp. 690–701 (cit. on p. 179).

[Van+07] Leonardo Vanneschi, Marco Tomassini, Philippe Collard, et al. „A comprehensive view of fitness landscapes with neutrality and fitness clouds“. In: *European Conference on Genetic Programming*. Springer. 2007, pp. 241–250 (cit. on p. 37).

[VFM00] Vesselin K Vassilev, Terence C Fogarty, and Julian F Miller. „Information characteristics and the structure of landscapes“. In: *Evolutionary computation* 8.1 (2000), pp. 31–60 (cit. on pp. 34, 37, 43, 45, 47).

[VFM02] Vesselin K Vassilev, Terence C Fogarty, and Julian F Miller. „Smoothness, Ruggedness and Neutrality of Fitness Landscapes: from Theory to Application“. In: *Advances in Evolutionary Computing: Theory and Applications* (2002), p. 3 (cit. on pp. 36, 37, 43, 45–47).

[Vir+20] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, et al. „SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python“. In: *Nature Methods* 17 (2020), pp. 261–272 (cit. on pp. 87, 92, 223, 224).

[VPC06] Leonardo Vanneschi, Yuri Pirola, and Philippe Collard. „A quantitative study of neutrality in GP boolean landscapes“. In: *Proceedings of the 8th annual conference on Genetic and evolutionary computation*. 2006, pp. 895–902 (cit. on p. 37).

[Wai+19] Christoph Waibel, Georgios Mavromatidis, Ralph Evins, and Jan Carmeliet. „A comparison of building energy optimization problems and mathematical test functions using static fitness landscape analysis“. In: *Journal of Building Performance Simulation* 12.6 (2019), pp. 789–811 (cit. on pp. 32, 38, 40, 41).

[Wan+19] Ting-Chen Wang, Chih-Yu Lin, Rung-Tzuo Liaw, and Chuan-Kang Ting. „Empirical analysis of Island model on large scale global optimization“. In: *2019 IEEE Congress on Evolutionary Computation (CEC)*. IEEE. 2019, pp. 342–349 (cit. on pp. 97, 99).

[Wat04] Duncan J Watts. *Six degrees: The science of a connected age*. WW Norton & Company, 2004 (cit. on p. 105).

[WC00] Michael Wooldridgey and Paolo Ciancarini. „Agent-oriented software engineering: The state of the art“. In: *International workshop on agent-oriented software engineering*. Springer. 2000, pp. 1–28 (cit. on p. 7).

[Wei90] Edward Weinberger. „Correlated and uncorrelated fitness landscapes and how to tell the difference“. In: *Biological cybernetics* 63.5 (1990), pp. 325–336 (cit. on pp. 37, 43).

[WJ95] Michael Wooldridge and Nicholas R Jennings. „Intelligent agents: Theory and practice“. In: *The knowledge engineering review* 10.2 (1995), pp. 115–152 (cit. on p. 7).

[WMZ20] Christoph Waibel, Georgios Mavromatidis, and Yong-Wei Zhang. „Fitness Landscape Analysis Metrics based on Sobol Indices and Fitness-and State-Distributions“. In: *2020 IEEE Congress on Evolutionary Computation (CEC)*. IEEE. 2020, pp. 1–8 (cit. on pp. 32, 33, 38, 40, 41, 64).

[Woo01] Michael Wooldridge. „Intelligent agents: The key concepts“. In: *ECCAI Advanced Course on Artificial Intelligence*. Springer. 2001, pp. 3–43 (cit. on p. 7).

[WS98] Duncan J Watts and Steven H Strogatz. „Collective dynamics of ‘small-world’networks“. In: *nature* 393.6684 (1998), pp. 440–442 (cit. on pp. 105, 106).

[XCP09] Bin Xin, Jie Chen, and Feng Pan. „Problem difficulty analysis for particle swarm optimization: deception and modality“. In: *Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation*. 2009, pp. 623–630 (cit. on p. 35).

[Yan+19] Tao Yang, Xinlei Yi, Junfeng Wu, et al. „A survey of distributed optimization“. In: *Annual Reviews in Control* 47 (2019), pp. 278–305 (cit. on pp. 8, 79).

List of Figures

1.1 Observer/Controller architecture . . . 9
1.2 Effect of CT connectivity on parallel metaheuristics . . . 15
1.3 Example of the inherent decomposition of the search space . . . 16
1.4 Development Objective . . . 23
1.5 Thesis structure . . . 25

2.1 Exemplary 2D Fitness Landscapes of continuous benchmark functions . 30
2.2 Summary of terms used to structure FLA aspects . . . 31
2.3 Rana function in the range [-5, 5] . . . 33
2.5 Different degrees of modality . . . 35
2.6 Examples for single- and multi-funnel landscapes . . . 36
2.7 Starting zones for progressive random walks . . . 44
2.8 Example for coupled landscapes . . . 50

3.1 Distributed FLA approach . . . 56
3.2 Examples for 10D composite spaces . . . 60
3.3 Examples for distributed computation of sensitivity indices . . . 62
3.4 Comparison of globally and distributedly computed sensitivity indices for 10-dimensional benchmark functions . . . 65
3.5 Comparison of globally and distributedly computed sensitivity indices for 100-dimensional benchmark functions . . . 66
3.6 Correlation of RMSE for first order effects and other FLA metrics in dimension 10 . . . 67
3.7 Overview of parallel distributed calculation of all feature group metrics 71
3.8 Heatmap showing Spearman correlation coefficients for all FLA metrics 73
3.9 Clusters 9 - 11: values of clustered metrics across problem instances . . 76

4.1 Types of parallel metaheuristics . . . 81
4.2 Design decisions regarding the cooperation mechanism . . . 83
4.3 Exemplary behavior in COHDA . . . 90
4.4 Exemplary behavior in IDICE . . . 96

5.1 Performance dimensions of cooperative search algorithms . . . 102
5.2 Updating the perceived current system state with strongly and weakly meshed topologies . . . 104
5.3 Examples of 10D Watts–Strogatz small-world graphs . . . 107

5.4 Global and local view on dynamic topology adaptation 109
5.5 Exemplary topology schedules with slow adaption 111
5.6 Exemplary topology schedules with fast adaption 112
5.7 Results for optimization runs with COHDA 114
5.8 Results for optimization runs with IDICE 115

6.1 Development Objective 119
6.2 Illustration of normalization approaches for $f(x)$ 126
6.3 Examples for normalization in all performance dimensions 128
6.4 Illustration of labeling problem 130
6.5 General structure of the ML model as core of ToVarO 131
6.6 Determination coefficients (R^2) for $f(x)_{norm}$ and $f(x)_{norm-max}$ 134
6.7 Determination coefficients (R^2) for $|SE|_{norm}$ and $|M|_{norm}$ 135
6.8 Spearman correlation coefficients for FLA metrics with $f(x)_{norm}$ 137
6.9 Spearman correlation coefficients for FLA metrics with $f(x)_{norm-max}$. 139
6.10 Spearman correlation coefficients for FLA metrics with $|SE|_{norm}$ 141
6.11 Spearman correlation coefficients for FLA metrics with $|M|_{norm}$ 142
6.12 Spearman coefficients relative to the corresponding p-values 143
6.13 Permutation importances for $f(x)_{norm}$ and $f(x)_{norm-max}$ 147
6.14 Permutation importances for $|SE|_{norm}$ and $|M|_{norm}$ 149
6.15 Exemplary predictive performance for COHDA 157
6.16 Exemplary predictive performance for IDICE 158
6.17 Exemplary predictive performance for highly similar problem instances 159
6.18 Exemplary prediction based CTV selection 164
6.19 Application example of the noise models 167
6.20 Predictions for COHDA after retraining with a partly noisy training data 171
6.21 Predictions for IDICE after retraining with a partly noisy training data . 172

A.1 FLA metrics for 100-dimensional benchmark functions 211
A.2 Ackley 214
A.3 Eggholder 214
A.4 Griewank 214
A.5 Rana 215
A.6 Salomon 215
A.7 Sargan 215
A.8 Schaffer F6 215
A.9 Schwefel 2.26 216
A.10 Penalized Schwefel 2.26 216
A.11 Qing 216
A.12 Alpine No. 2 219
A.13 Drop Wave 219
A.14 Happy Cat 219

A.15 Levy 220
A.16 Michalewicz 220
A.17 Mishra 11 220
A.18 Rastrigin 220
A.19 Rosenbrock 221
A.20 Sphere 221
A.21 Wavy function 221
A.22 Weierstrass 221
A.23 Whitley 222
A.24 Xin-She Yang 1 222
A.25 Comparison of performance of IDICE and COHDA 226

B.1 Effects of the initial sampling parameters (10D) 228
B.2 Effects of the initial sampling parameters (100D) 229
B.3 Effects of the local sampling parameters (10D) 230
B.4 Effects of the local sampling parameters (100D) 231
B.5 Effect of local sampling parameters across problem instances (100D) . 232
B.6 Effect of initial sampling parameters on the deviation between central and distributed FSD-FLA (10D) 234
B.7 Effect of initial sampling parameters on the deviation between central and distributed FSD-FLA (100D) 234
B.8 Effect of local sampling parameters on the deviation between central and distributed FSD-FLA (10D) 235
B.9 Effect of local sampling parameters on the deviation between central and distributed FSD-FLA (100D) 235
B.10 Effect of initial sampling parameters on the deviation between central and distributed FEM (10D) 237
B.11 Effect of initial sampling parameters on the deviation between central and distributed FEM (100D) 238
B.12 Effect of local sampling parameters on the deviation between central and distributed FEM (10D) 238
B.13 Effect of local sampling parameters on the deviation between central and distributed FEM (100D) 239
B.14 Effect of local sampling parameters and repetitions on the deviation between central and distributed FEM (10D) 239
B.15 Effect of local sampling parameters and repetitions on the deviation between central and distributed FEM (10D) 240
B.16 Comparison of central and distributed calculation of information theoretic measures for 10-D problem instances 241
B.17 Comparison of central and distributed calculation of gradient based measures and the dispersion metric for 10-D problem instances 242
B.18 Clusters 9 - 11: values of clustered metrics across problem instances . . 246

B.19 Clusters 12 - 14: values of clustered metrics across problem instances . 247
B.20 Cluster 15: values of clustered metrics across problem instances 248

C.1 Probability value for spearman coefficients with $f(x)_{norm}$ 254
C.2 Probability value for spearman coefficients with $f(x)_{norm-max}$ 255
C.3 Probability value for spearman coefficients with $|SE|_{norm}$ 256
C.4 Probability value for spearman coefficients with $|M|_{norm}$ 257

List of Tables

1.1 Definitions of terms used in eq. (1.1) and eq. (1.2) 5

2.1 Overview of applied FLA techniques 49

3.1 Parameters of distributed FLA 58
3.2 Clustered FLA metrics based on Spearman correlation coefficient . . . 74

5.1 Employed topology variants for 100-D benchmark problems 113

6.1 FLA metrics clustered by Spearman coefficient and representatives . . 123
6.2 Setups for CTVs 124
6.3 Permutation feature importance for COHDA 150
6.4 Permutation feature importance for IDICE 151
6.5 Setups for MLP and random forest regressor parameter tuning 154
6.6 Performance of best found MLP and the random forest regressor 155
6.7 Setups to prioritize performance dimensions using weighted sums . . . 161
6.8 Predictions for *Penalized Schwefel 2.26 - 8* 163
6.9 MSE and R^2 after training with partly noisy data 168

A.1 Benchmark function set No. 1 213
A.2 Benchmark function set No. 2 219
A.3 Applied parameter Setting for COHDA 224
A.4 Applied parameter Setting for IDICE 225

B.1 Overview: impact of parameters and the chosen values for FLA features 244

C.1 Proportionally high **negative** correlation for $f(x)_{norm}$ 249
C.2 Proportionally high **positive** correlation for $f(x)_{norm}$ 251
C.3 Proportionally high **positive** correlation for $|SE|_{norm}$ 252
C.4 Proportionally high **negative** correlation for $|SE|_{norm}$ 253
C.5 Retrieved MSE with permuted features for COHDA 258
C.6 Retrieved MSE with permuted features for COHDA 259
C.7 Retrieved MSE with permuted features for IDICE 260
C.8 Retrieved MSE with permuted features for IDICE 261
C.9 Parameter setups random forest regressors 262
C.10 Parameter setups random forest regressors with noise 263

Erklärung

Hiermit erkläre ich, dass ich diese Arbeit eigenständig verfasst und keine anderen als die angegebenen Hilfsmittel und Quellen benutzt habe. Ebenso versichere ich, dass diese Dissertation weder in ihrer Gesamtheit noch in Teilen einer anderen wissenschaftlichen Hochschule zur Begutachtung in einem Promotionsverfahren vorgelegen hat.

Declaration

I declare that I have written this thesis independently and that I have used only the resources indicated. I also affirm that this dissertation has not been submitted, in whole or in part, to any other university for assessment in a doctoral procedure.

Oldenburg, 23.05.2023

Stefanie Isabel Holly

Appendix A

A

A.1 Benchmark functions

This appendix includes the real-valued continuous benchmark functions for global optimization that were used for this study. Most of these benchmark functions were taken from collections of continuous benchmark functions for global optimization, mainly [MY13] and [PS22]. Scalability was an important selection criterion, as it should be possible to construct scenarios of arbitrary size. Also, most of the functions are inseparable to reflect the high interdependence of solution variables in the motivating energy optimization problems. Figure A.1 shows the values of the FLA metrics presented in section 2.2 for the 100-dimensional versions of the benchmark functions. The metrics have been split into two sets for this purpose due to different scales and for better visibility. The figures show that the values of the FLA metrics vary widely across the full set of benchmark functions. Some patterns emerge, such as high values for the ruggedness metrics, i.e. FEM_{macro} and FEM_{micro}, usually being accompanied by low values for the smoothness metrics SEM_{macro} and SEM_{micro}. But functions that are closely aligned at some points usually still have significantly different values for at least one other metric. Of course, a set of 23 functions can by no means represent a complete coverage of all possible combinations of function properties. Nevertheless, the set presented should provide a solid basis for the experiments performed in this thesis.

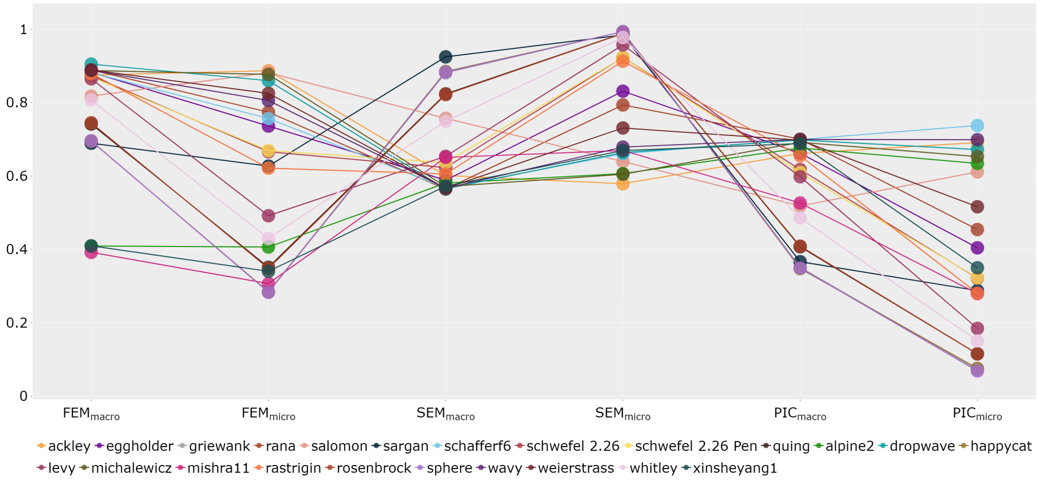
The function are split in two sets. Set 1 was primarily used for parameter tuning or preliminary studies at various points in the development of the overall concept. Together with set 2, the total set of functions used for the more extensive experiments is obtained. The tables A.1 and A.2 display the definitions, the default domains, and the values and positions of the global optima when the functions are used in the regular way. In addition, 3-D plots of each function are shown from different sections of the domain. From left to right:

- Full domain
- 10% of the domain, centered around the original domain center

- 10% of the domain, center shifted 25% of the original domain range from the original center
- 1% of the origin domain, centered around the original domain center

On the one hand, the plots are intended to give an impression of the different properties of the fitness landscapes of the functions. On the other hand, they show how different the characteristics of the different subspaces of a composites space generated on the basis of a function might be.

(a) metrics set 1



(b) metrics set 2

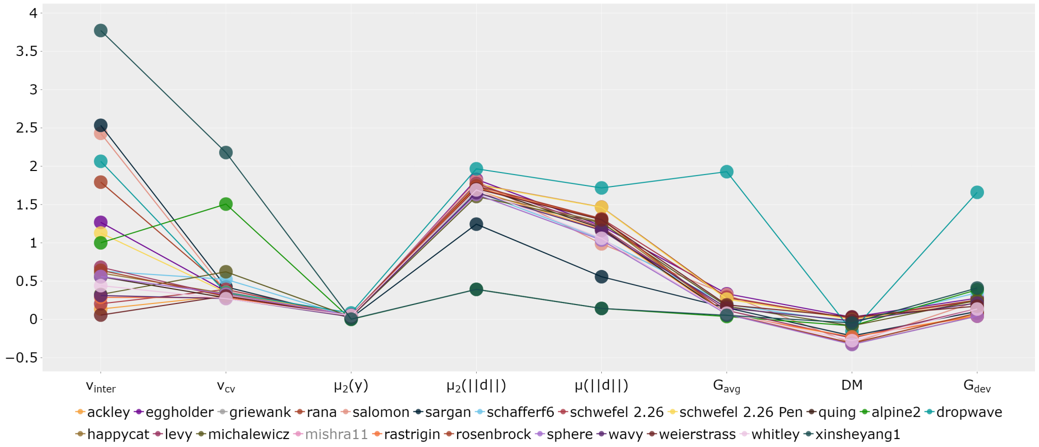


Fig. A.1.: FLA metrics for 100-dimensional benchmark functions

A.1.1 Function set 1

function	**Definition, Domain and global optimum** $f(\vec{x}^*)$	**figure no.**
Ackley [MY13; Li+13]	$f(\vec{x}) = -20exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2})$ $-exp(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)) + a + exp(1)$ $x_i \in [-35, 35]$ $f(\vec{x}^*)$ = 0, with $x^* = (0, \dots, 0)$	A.2
Eggholder [MY13; Al-15]	$f(\vec{x}) = \sum_{i=1}^{n-1}[-x_i sin(\sqrt{\lvert x_i - x_{i+1} - 47\rvert})$ $-(x_{i+1} + 47)sin(\sqrt{\lvert 0.5x_i + x_{i+1} + 47\rvert})]$ $x_i \in [-512, 512]$ $f(\vec{x}^*)$ = -959.64, with $x^* = (512, 404)$ for $n = 2$	A.3
Griewank [MY13]	$f(\vec{x}) = 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}})$ $x_i \in [-100, 100]$ $f(\vec{x}^*)$ = 0, with $x^* = (0, \dots, 0)$	A.4
Rana [MY13]	$f(\vec{x}) = \sum_{i=1}^{n}[x_i sin(t_2)cos(t_1) + (x_1 + 1)sin(t_1)cos(t_2)]$ with $t_1 = \sqrt{\lvert x_1 + x_i + 1\rvert}$ and $t_2 = \sqrt{\lvert x_1 - x_i + 1\rvert}$ $x_i \in [-500, 500]$ $f(\vec{x}^*)$ = -959.64, with $x^* = (512, 404)$ for $n = 2$	A.5
Salomon [MY13]	$f(\vec{x}) = 1 - cos(2\pi\sqrt{\sum_{i=1}^{D} x_i^2}) + 0.1\sqrt{\sum_{i=1}^{D} x_i^2}$ $x_i \in [-100, 100]$ $f(\vec{x}^*)$ = 0, with $x^* = (0, \dots, 0)$	A.6

function	**Definition, Domain and global optimum** $f(\vec{x}^*)$	**figure no.**
Sargan [MY13]	$f(\vec{x}) = \sum_{i=1}^{n} n(x_i^2 + 0.4\sum_{j\neq i}^{n} x_i x_j)$ $x_i \in [-100, 100]$ $f(\vec{x}^*) = 0$, with $x^* = (0, \ldots, 0)$	A.7
Schaffer F6 [MY13]	$f(\vec{x}) = \sum_{i=1}^{n} 0.5 + \frac{sin^2(\sqrt{x_i^2+x_{i+1}^2})-0.5}{[1+0.001\cdot(x_i^2+x_{i+1}^2)]^2}$ $x_i \in [-100, 100]$ $f(\vec{x}^*) = 0$, with $x^* = (0, \ldots, 0)$	A.8
Schwefel 2.26 [Sch81; MY13]	$f(\vec{x}) = 418.9829n - \sum\limits_{i=1}^{n} x_i \sin\sqrt{\lvert x_i\rvert}$ $x_i \in [-500, 500]$ $f(\vec{x}^*) = 0$, with $x^* = (420, 968746, \ldots, 420, 968746)$	A.9
Penalized Schwefel 2.26 [HN20]	$f(\vec{x}) = 418.9829n - \sum\limits_{i=1}^{n} x_i \sin\sqrt{\lvert x_i\rvert} + \left\lvert \sum\limits_{i=1}^{\frac{n}{2}} x_{2i} - \sum\limits_{i=1}^{\frac{n}{2}} x_{2i-1} \right\rvert$ $x_i \in [-500, 500]$ $f(\vec{x}^*) = 0$, with $x^* = (420, 968746, \ldots, 420, 968746)$	A.10
Qing [MY13]	$f(\vec{x}) = \sum_{i=1}^{n}(x^2 - i)^2$ $x_i \in [-500, 500]$ $f(\vec{x}^*) = 0$, with $x^* = (\pm\sqrt{i}, \ldots, \pm\sqrt{i})$	A.11

Tab. A.1.: Benchmark function set No. 1

Fig. A.2.: Ackley

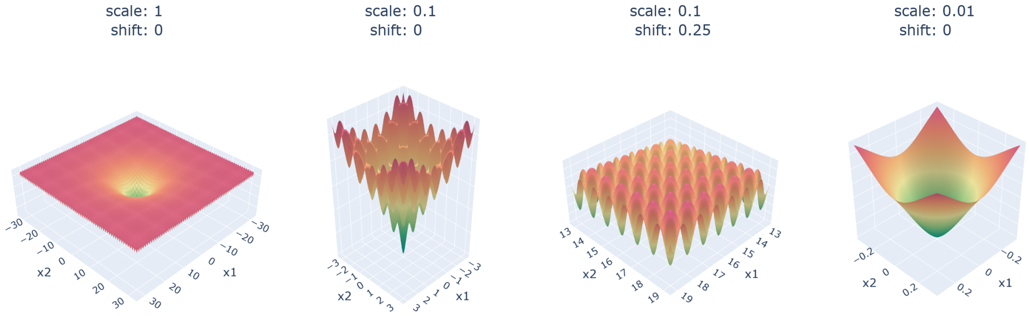

scale: 1 shift: 0
scale: 0.1 shift: 0
scale: 0.1 shift: 0.25
scale: 0.01 shift: 0

Fig. A.3.: Eggholder

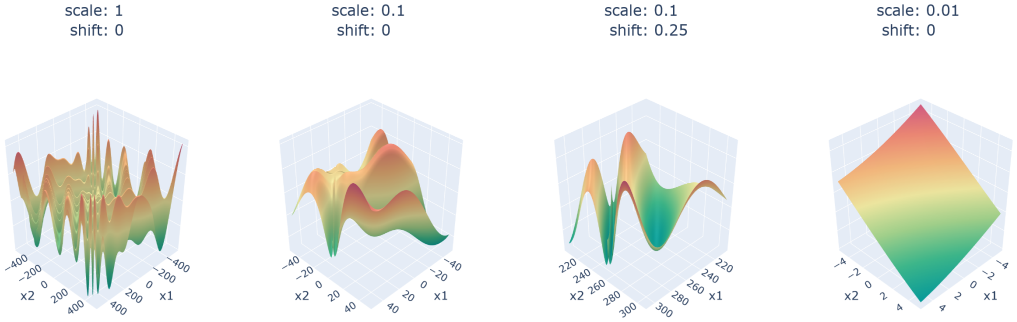

scale: 1 shift: 0
scale: 0.1 shift: 0
scale: 0.1 shift: 0.25
scale: 0.01 shift: 0

Fig. A.4.: Griewank

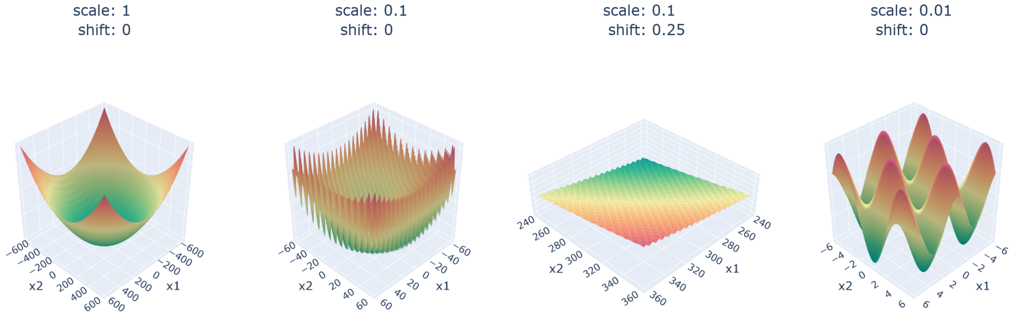
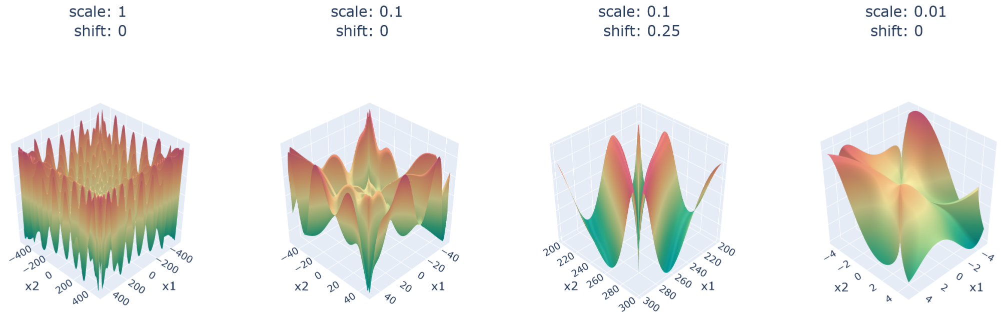

scale: 1 shift: 0
scale: 0.1 shift: 0
scale: 0.1 shift: 0.25
scale: 0.01 shift: 0

Fig. A.5.: Rana

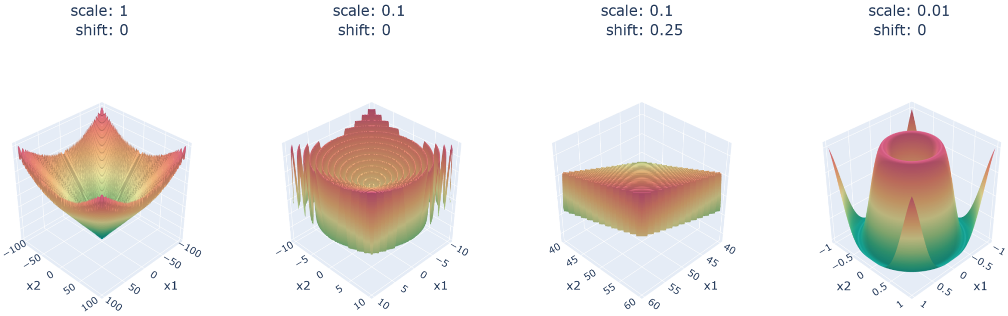

scale: 1
shift: 0

scale: 0.1
shift: 0

scale: 0.1
shift: 0.25

scale: 0.01
shift: 0

Fig. A.6.: Salomon

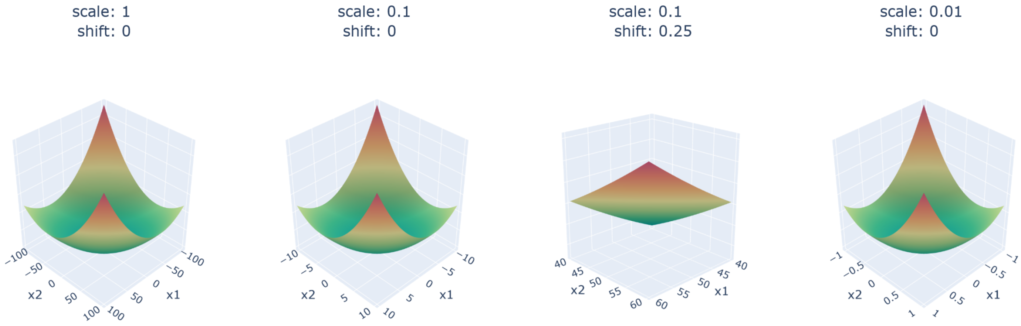

scale: 1
shift: 0

scale: 0.1
shift: 0

scale: 0.1
shift: 0.25

scale: 0.01
shift: 0

Fig. A.7.: Sargan

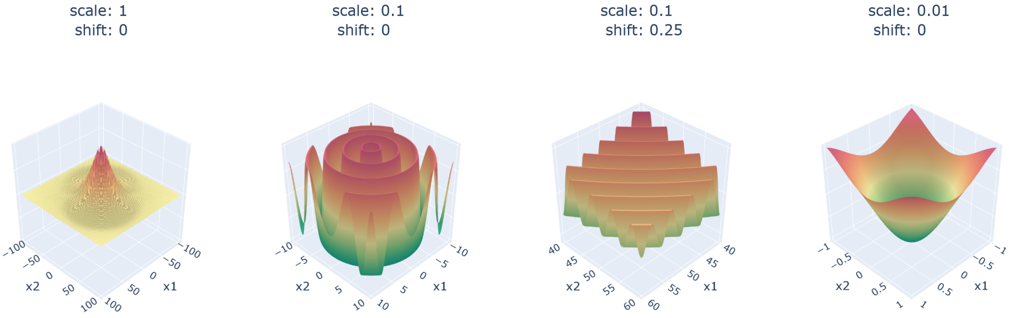

scale: 1
shift: 0

scale: 0.1
shift: 0

scale: 0.1
shift: 0.25

scale: 0.01
shift: 0

Fig. A.8.: Schaffer F6

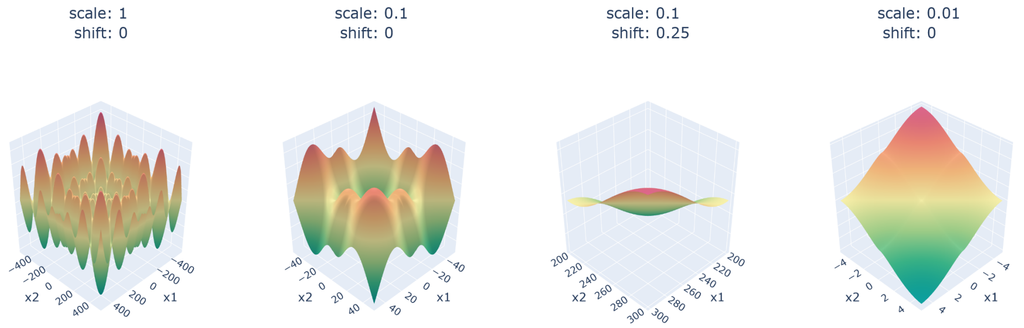

scale: 1
shift: 0

scale: 0.1
shift: 0

scale: 0.1
shift: 0.25

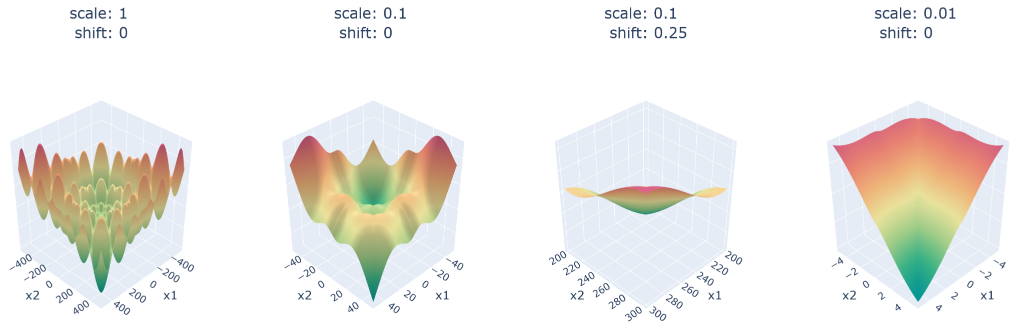
scale: 0.01
shift: 0

Fig. A.9.: Schwefel 2.26



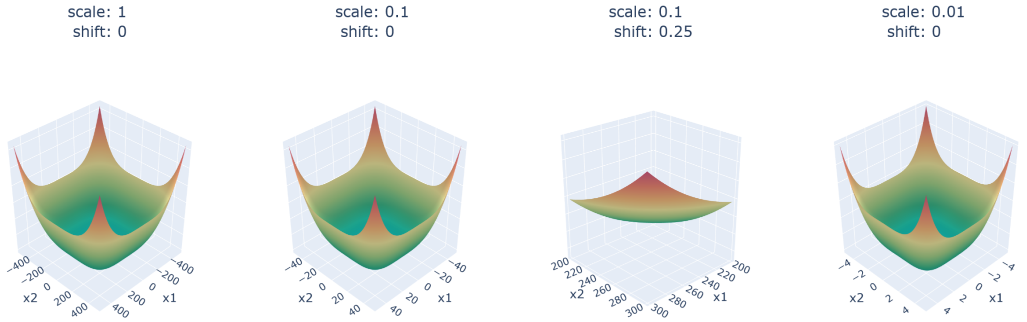
scale: 1 shift: 0 | scale: 0.1 shift: 0 | scale: 0.1 shift: 0.25 | scale: 0.01 shift: 0

Fig. A.10.: Penalized Schwefel 2.26



scale: 1 shift: 0 | scale: 0.1 shift: 0 | scale: 0.1 shift: 0.25 | scale: 0.01 shift: 0

Fig. A.11.: Qing



scale: 1 shift: 0 | scale: 0.1 shift: 0 | scale: 0.1 shift: 0.25 | scale: 0.01 shift: 0

A.1.2 Function set 2

function	**Definition, Domain and global optimum** $f(\vec{x}^{*})$	**figure no.**
Alpine 2 [MY13]	$f(\vec{x}) = -\prod_{i=1}^{n} \sqrt{x_i} sin(x_i)$ $x_i \in [0, 10]$ $f(\vec{x}^{*}) = 2.808^n$, with $x^* = (7.917, \ldots, 7.917)$	A.12
Drop Wave [PS22]	$f(\vec{x}) = 1 - \frac{1+cos\left(12\sqrt{\sum_{i=1}^{n} x_i^2}\right)}{0.5\sum_{i=1}^{n} x_i^2+2}$ $x_i \in [-5.12, 5.12]$ $f(\vec{x}^{*})$ = 0, with $x^* = (0, \ldots, 0)$	A.13
Happy Cat [BF12; PS22]	$f(\vec{x}) = \left[(\lvert\lvert\mathbf{x}\rvert\rvert^2 - n)^2\right]^{\alpha} + \frac{1}{n}\left(\frac{1}{2}\lvert\lvert\mathbf{x}\rvert\rvert^2 + \sum_{i=1}^{n} x_i\right) + \frac{1}{2}$ $x_i \in [-20, 20]$ $f(\vec{x}^{*})$ = 0, with $x^* = (-1, \ldots, -1)$	A.14
Levy [LM05]	$f(\vec{x}) = sin^2(\pi w_1) + \sum_{i=1}^{d-1}(w_i - 1)^2[1 + 10sin^2(\pi w_i + 1)]$ $+(w_d - 1)^2[1 + sin^2(2\pi w_d)]$ $x_i \in [-10, 10]$ and $w_i = 1 + \frac{x_i - 1}{4}$ for all $i = 1, \ldots, d$ $f(\vec{x}^{*})$ = 0, with $x^* = (1, \ldots, 1)$	A.15
Michalewicz function [Al-15]	$f(\vec{x}) = -\sum_{i=1}^{n}(sin(x_i) * sin(\frac{ix_i^2}{\pi})^{2m})$ $x_i \in [0, \pi]$ and $m = 10$ $f(\vec{x}^{*}) \approx -1.8$, with $x^* \approx (1.8, 1.6)$ for $n = 2$	A.16

function	**Definition, Domain and global optimum** $f(\vec{x}^*)$	**figure no.**
Mishra 11 [MY13]	$f(\vec{x}) = \left(\frac{1}{n}\sum_{i=1}^{n}\lvert x_i\rvert - \left(\prod_{i=1}^{n}\lvert x_i\rvert\right)^{\frac{1}{n}}\right)^2$ $x_i \in [-10, 10]$ $f(\vec{x}^*)$ = 0, with $x^* = (0, \ldots, 0)$	A.17
Rastrigin [Li+13]	$f(\vec{x}) = 10n + \sum_{i=1}^{n}(x_i^2 - 10cos(2\pi x_i))$ $x_i \in [-5.12, 5.12]$ $f(\vec{x}^*)$ = 0, with $x^* = (0, \ldots, 0)$	A.18
Rosenbrock [MY13]	$f(\vec{x}) = \sum_{i=1}^{n}(100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2)$ $x_i \in [-30, 30]$ $f(\vec{x}^*)$ = 0, with $x^* = (1, \ldots, 1)$	A.19
Sphere Function [MY13]	$f(\vec{x}) = \sum_{i=1}^{n}(x_i^2)$ $x_i \in [-100, 100]$ $f(\vec{x}^*)$ = 0, with $x^* = (0, \ldots, 0)$	A.20
Wavy Function [MY13]	$f(\vec{x}) = 1 - \frac{1}{n}\sum_{i=1}^{n} cos(kx_i) * exp(\frac{-x_i^2}{2})$ $x_i \in [-2\pi, 2\pi]$ and $k = 50$ $f(\vec{x}^*)$ = 0, with $x^* = (0, \ldots, 0)$	A.21
Weierstrass Function [MY13; Fin+; PS22]	$f(\vec{x}) = \sum_{i=1}^{n}(\sum_{k=0}^{k_{max}}(a^k * \cos(2\pi b^k(x_i + 0.5))))$ $\quad -n * \sum_{k=0}^{k_{max}}(a^k \cos(\pi b^k))$ $x_i \in [-5, 5]$ and $\quad a = 0.5, \quad b = 3, \quad k_{max} = 11$ $f(\vec{x}^*)$ = 0, with $x^* = (0, \ldots, 0)$	A.22
Whitley [MY13]	$f(\vec{x}) = \sum_{i=1}^{n}\sum_{j=1}^{n}\Big(\frac{(100(x_i^2 - x_j)^2 + (1 - x_j)^2)^2}{4000} - \cos(100(x_i^2 - x_j)^2$ $\quad +(1 - x_j)^2) + 1\Big)$ $x_i \in [-10.24, 10.24]$ $f(\vec{x}^*)$ = 0, with $x^* = (1, \ldots, 1)$	A.23

function	**Definition, Domain and global optimum $f(\vec{x}^*)$**	**figure no.**
Xin-She Yang 1 [PS22]	$f(\vec{x}) = \left(\sum_{i=1}^{n}(\lvert x_i \rvert)\right) * exp\left(-\sum_{i=1}^{n} sin(x_i^2)\right)$ $x_i \in [-2\pi, 2\pi]$ $f(\vec{x}^*) = 0$, with $x^* = (0, \ldots, 0)$	A.24

Tab. A.2.: Benchmark function set No. 2

Fig. A.12.: Alpine No. 2

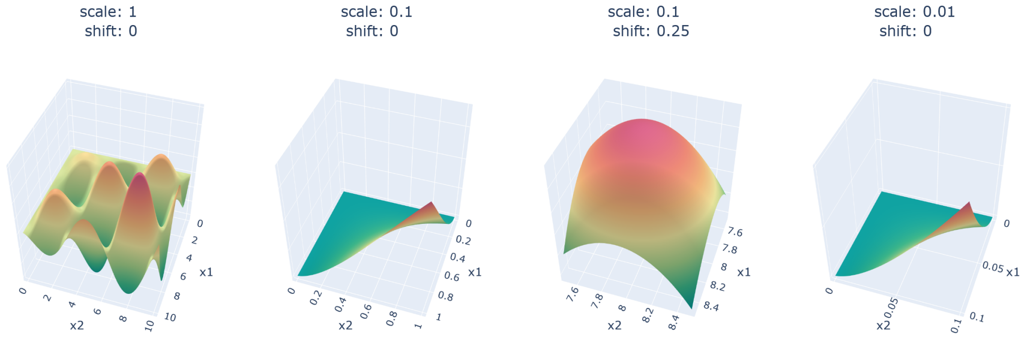


Fig. A.13.: Drop Wave

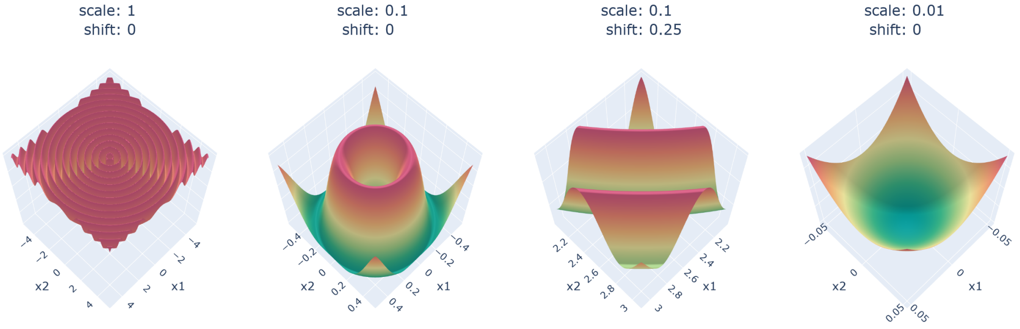


Fig. A.14.: Happy Cat

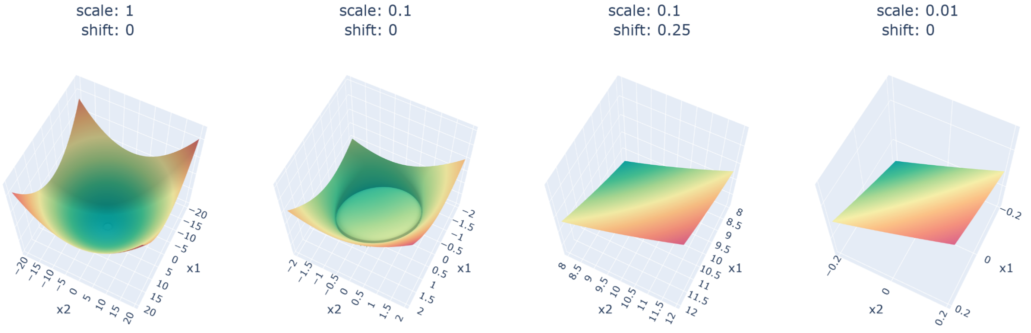


Fig. A.15.: Levy

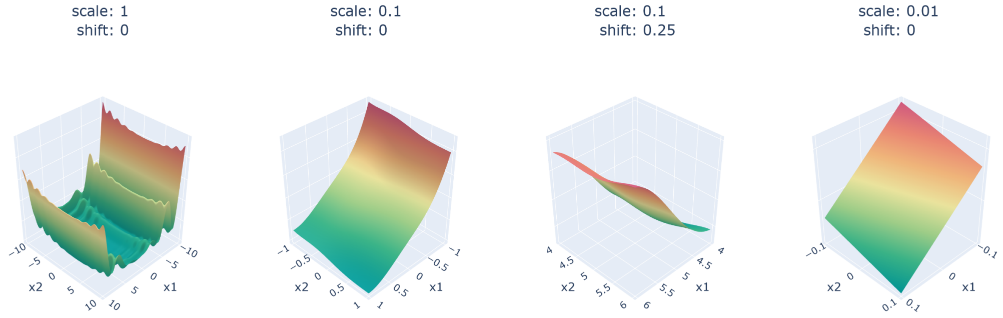


Fig. A.16.: Michalewicz

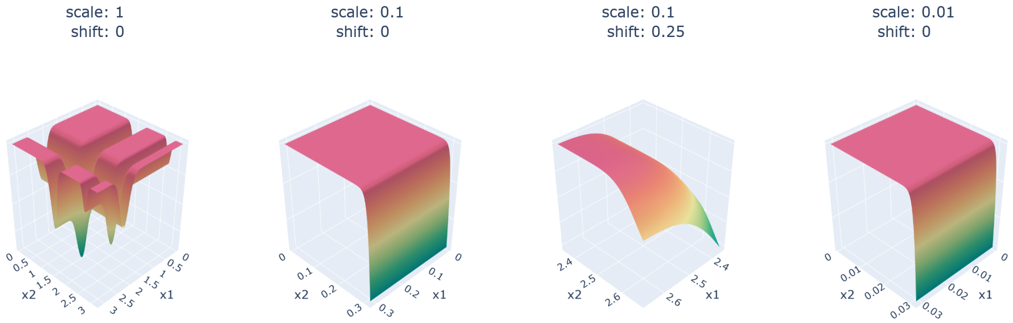


Fig. A.17.: Mishra 11

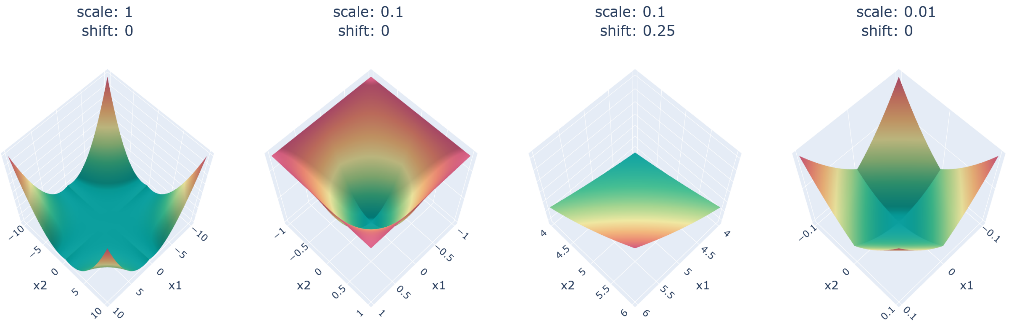


Fig. A.18.: Rastrigin

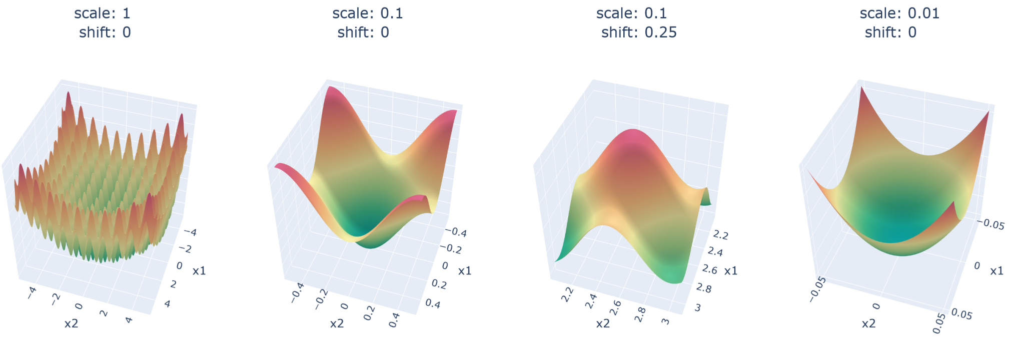


Fig. A.19.: Rosenbrock

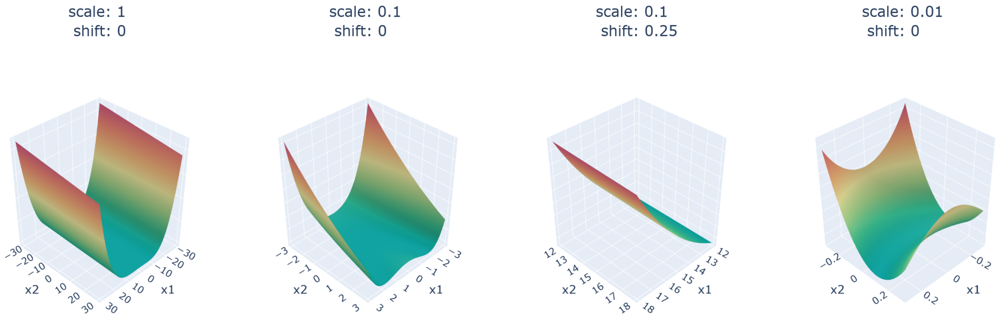

scale: 1 shift: 0 | scale: 0.1 shift: 0 | scale: 0.1 shift: 0.25 | scale: 0.01 shift: 0

Fig. A.20.: Sphere

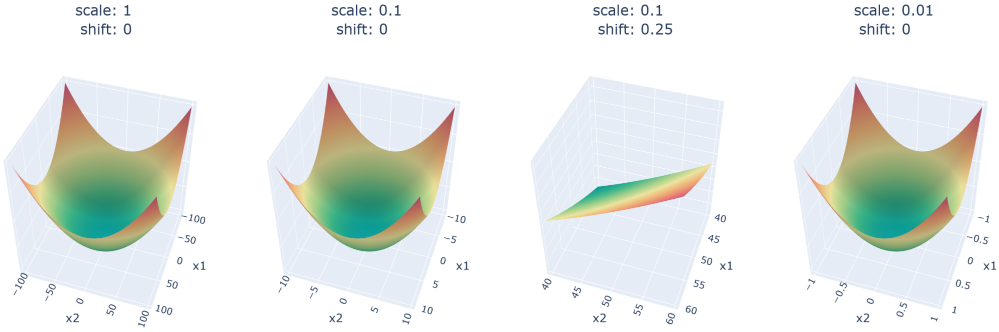

scale: 1 shift: 0 | scale: 0.1 shift: 0 | scale: 0.1 shift: 0.25 | scale: 0.01 shift: 0

Fig. A.21.: Wavy function

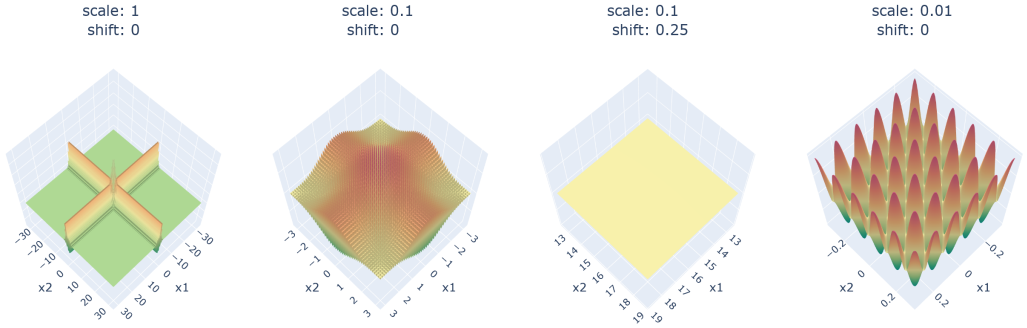

scale: 1 shift: 0 | scale: 0.1 shift: 0 | scale: 0.1 shift: 0.25 | scale: 0.01 shift: 0

Fig. A.22.: Weierstrass

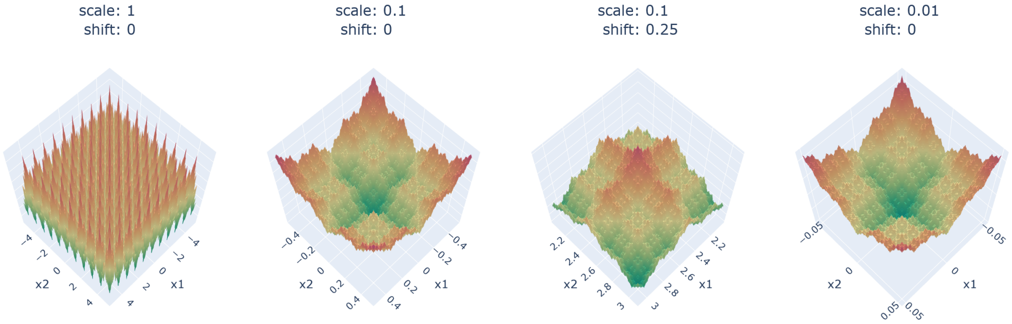

scale: 1 shift: 0 | scale: 0.1 shift: 0 | scale: 0.1 shift: 0.25 | scale: 0.01 shift: 0

Fig. A.23.: Whitley

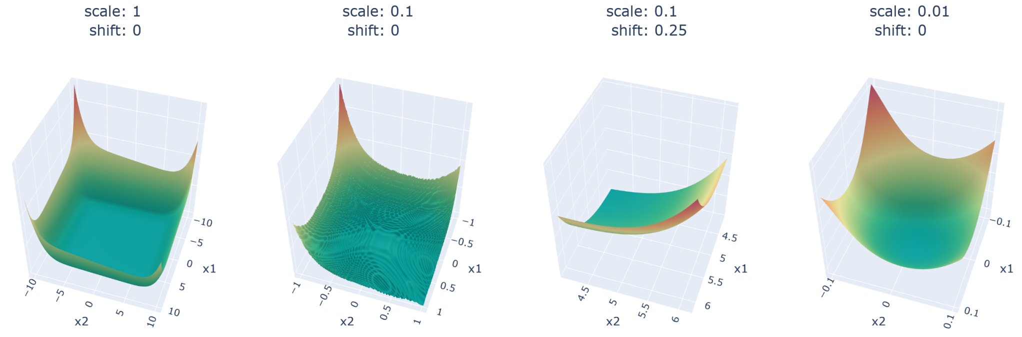
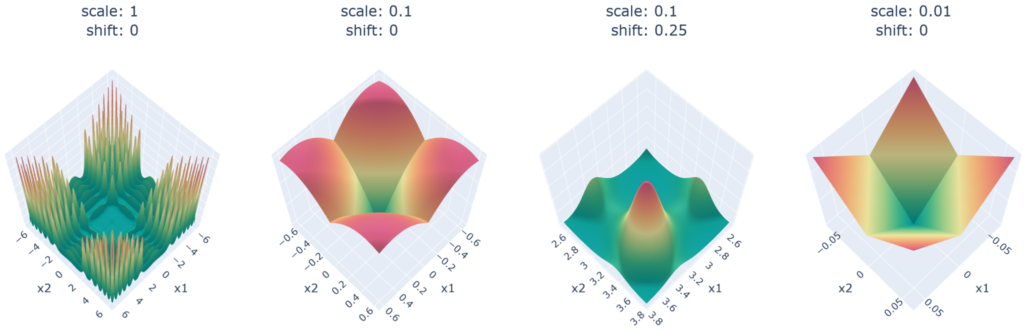


Fig. A.24.: Xin-She Yang 1



A.2 Algorithm Setup

The general principles of the two algorithms in use are described in the respective chapters, namely section 4.2.1 and section 4.2.2. For both algorithms, several parameters must be set appropriately. The focus of this work is only on one hyperparameter, namely the communication topology. Since the goal is to evaluate optimization runs with different communication topologies, reasonable default values for other parameters were determined in advance, but no extensive parameter tuning was performed. The analysis of the interactions of the topology with other parameters is left for future work. The default settings are presented below for both algorithms.

A.2.1 COHDA

A major advantage of COHDA is that hardly any hyperparameters need to be specified. In the implementation for continuous global optimization problems used here, only the so-called reaction time and the choice of the local optimizer remain. The reaction time or sleeping time is a short period of time during which the agent remains idle between two iterations of its perceive - decide - act phases. During this time, new messages may arrive from other agents. These messages are processed together in the subsequent perceive phase. In [Hin14] Hinrichs showed that such an intermediate idle phase has a positive effect on the performance of the algorithm. This is especially the case if the agents are slightly asynchronous due to different reaction times, i.e. with increased inter-agent variation. A similar direction was taken in the study of Bremer et al. [BL19]. By explicitly adding laziness to the agent behavior, i.e. by randomly delaying the decision phase, they were able to show an improvement in optimization performance in many cases. Since inter-agent variation is expected to have positive effects, but a high reaction time also increases the simulation time considerably, a compromise had to be found. In all optimizations performed with COHDA in this thesis, the agents choose the reaction time after each iteration as a random value between 0.05 and 0.15 seconds.

Second, the local optimizer must be selected. In the chosen setting, this optimizer has to optimize only two decision variables within their feasible bounds for a given objective function. Therefore, a variety of pre-implemented global optimization methods can be used. In a preliminary study, several optimizers from the python library SciPy [Vir+20] were tested. The *simplicial homology global optimization* (SHGO) [ESF18] showed the best tradeoff between runtime and solution quality. The local optimizers have their own hyperparameters. For SHGO, the default parameters of the SciPy library have been used mainly, with only minor adjustments in a few places. A final adaptation of COHDA is to limit the simulation time to 50 minutes (chosen based on the number of agents). Since the simulations run on a single machine, this corresponds to about one minute of CPU time per agent. In a real distributed environment, they would run in parallel, but there would be communication overhead. Without a fixed timeout, some of the simulation runs for particularly difficult problems may otherwise take an extremely long time. However, due to the anytime property of COHDA, complete solutions are available shortly after the start of the optimization. In the case of a timeout, the best solution that has been found so far will be taken as the result of the negotiation. Table A.3 gives an overview of all parameters required for COHDA in this variant and the values chosen.

A.2.2 IDICE

The IDICE algorithm is a combination of the island model and co-evolutionary algorithms. In addition, the local optimization specifically implemented is Differential Evolution (DE). Therefore, several parameters have to be selected, some of which are related to the characteristics of the island model, some to DE, and some to their combination in the spatially distributed setup. The response time between iterations and the 50-minute timeout are handled in the same way as for COHDA.

parameter	**description**	**value**
reaction time	idle time between the "act" step of an iteration and the "perceive" step of the next iteration	random value between 0.05 and 0.15 seconds
local optimizer	optimization algorithm that is applied in each "decide" step to choose new values for the agent's own decision variables	*simplicial homology global optimization* (SHGO)
sampling method	sampling method applied by SHGO	*simplicial* (default by SciPy) - "provides the theoretical guarantee of convergence to the global minimum in finite time" [Vir+20]
n	"Number of sampling points used in the construction of the simplicial complex" [Vir+20]	200 (100 is default by SciPy)
iters	"Number of iterations used in the construction of the simplicial complex" [Vir+20]	5 (1 is default by SciPy)
timeout	timeout of an optimization run	50 minutes (wall-clock time)

Tab. A.3.: Applied parameter Setting for COHDA

In contrast to COHDA in combination with SHGO, it is not possible to resort to default parameters. Therefore, a small preliminary study on parameter tuning was carried out. An experimental setup for assigning parameters was created using Latin Hypercube and tested using 3 composite spaces per benchmark function. Depending on the weighting of solution quality and convergence time, different configurations are advantageous. As a result of the preliminary study, two configurations were selected, both of which are listed in table A.4. The normalized error and the normalized computation time as a weighted sum were used as selection criteria. Config 1 puts more emphasis on solution quality (6:4), while Config 2 puts more emphasis on fast computation time (4:6).

Figure A.25 shows the performance of both configurations for several benchmark functions, also compared to COHDA. For each benchmark function, the first 3 composite spaces were used and optimizations were performed with a *static* small-world topology ($k = 24$, $p = 0.5$) and simplified setups of *decrease* ($k = 50$, $p = 0.5$) and *increase* ($k = 2$, $p = 0.5$). In each case there was a fast ($\alpha = 0.7$) and a slow ($\alpha = 0.05$) adaptation variant. Normalization was performed separately for each composite space. A min-max scaling was used for all occurring values. The

parameter	**description**	**config 1**	**config 2**
reaction time	idle time between the "act" step of an iteration and the "perceive" step of the next iteration	random value between 0.05 and 0.15 seconds	
$iter_{min}$	minimal number of iterations each agent must perform	18	37
μ	number of iterations between migrations	6	10
k	k-best solutions are migrated	10	12
m	population size on each island	10	30
ρ	how many samples should be generated in the splicing operation	equal to number of decision variables	
$strategy$	mutation strategy in DE	rand2bin	currentto-best1exp
$iter_{max}$	number of iterations without improvement after which the agent requests termination	231	284
cf	convergence factor: after $cf * \mu$ iterations of the DE solver the rounding factor for self-retrieved candidates is reduced by one decimal place	45	7
timeout	timeout of an optimization run	50 minutes (wall-clock time)	

Tab. A.4.: Applied parameter Setting for IDICE; The parameters' functionality is described in more detail in section 4.2.2 in the relevant steps.

figure shows that the performance of the different heuristics or configurations varies depending on the benchmark function and thus probably depends on the problem characteristics.

COHDA is usually more efficient in terms of the number of search executions and the number of messages. But the two IDICE configurations are quite competitive, especially in terms of the error rate, where they perform significantly better in several benchmarks. In addition, the difference between the two configurations becomes apparent. Config 1 requires more resources but yields better results. Config 2, on the other hand, is slightly less resource-intensive, but usually finds sufficiently good solutions. Both setups should be suitable for investigating the influence of the communication topology, since they show different performance in different benchmark functions.

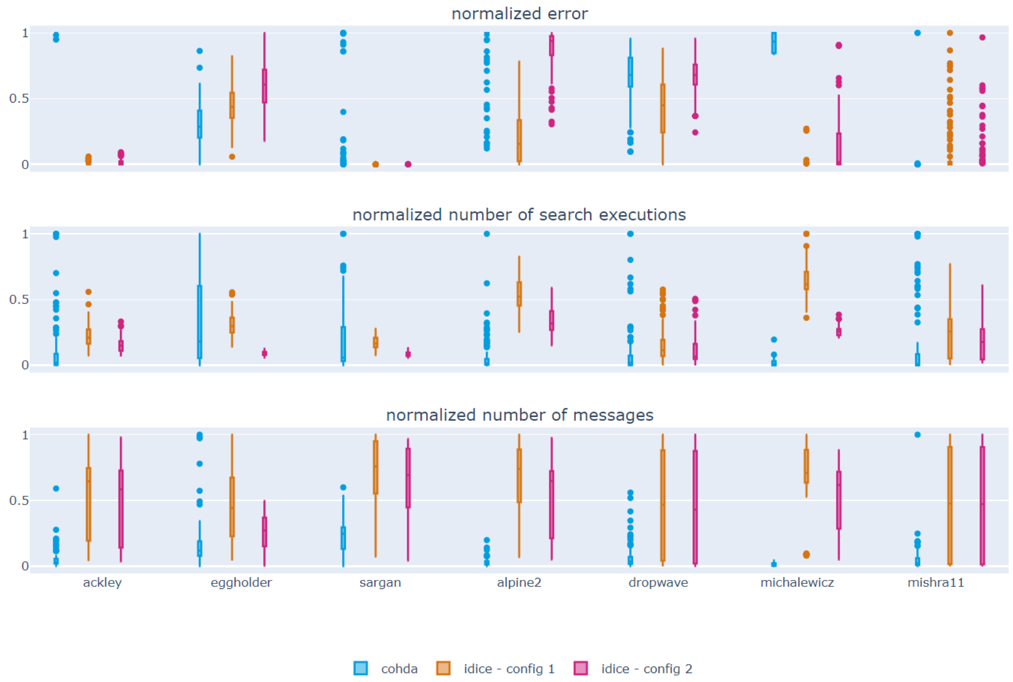


Fig. A.25.: Comparison of performance of IDICE configurations and COHDA on a subset of benchmark functions

B

Appendix B

B.1 Parameter Tuning for distributed FLA

In chapter 3, the techniques for the computation of FLA metrics have been divided into different *feature groups* depending on the sampling techniques used for their computation in the different phases of the distributed FLA. Below, a parameter tuning for distributed computation is performed for each of these *feature groups*. The tunable parameters and value ranges are listed in table 3.1. Centrally computed FLA metrics are always used for reference.

B.1.1 Distributed sensitivity analysis

To fine-tune the distributed calculation of the sensitivity indices, 5 composite spaces were selected from each of the 10 benchmark functions in set number 1 (appendix A.1.1), resulting in 50 problem instances (per dimension). The calculation was performed using the different parameter combinations from table 3.1. The global calculation using SAlib [IUH22; HU17] serves as a reference. Thus, across all decision variables, the RMSE can be computed for both the first order and total order effects:

$$\boldsymbol{S_{i}}_{RMSE} = \sqrt{\frac{1}{k}\sum_{i=1}^{k}(S_i - S_{i-ref})^2} \tag{B.1}$$

$$\boldsymbol{S_{ti}}_{RMSE} = \sqrt{\frac{1}{k}\sum_{i=1}^{k}(S_{ti} - S_{ti\ ref})^2} \tag{B.2}$$

where S_i and S_{ti} are the distributedly computed indices, S_{i-ref} and S_{ti-ref} the indices computed by SAlib and k the number decision variables. The experiments were performed with 10 and 100 decision variables to investigate the impact on different system sizes.

In the following, first the effect of the initial sampling parameters and then the effect of the local FLA are examined. In each case, the sampling method and the number of samples are varied. Figure B.1 shows the results for the combination of different initial sampling methods and number of samples. There is no discernible trend in either the error rates of the two indices, nor in the calculation duration. This could be due to the small number of variables (2) that each agent holds in this configuration. When considering larger dimensions, a drop in the error rate can be observed for more than 32 initial samples. To investigate whether this trend continues, even larger values for the initial sample size were tested for the 100-dimensional problems, namely 256 and 512. Figure B.2 shows the results. For sample sizes greater than 32, a reduction in large error values resp. outliers in the violin plots for the errors in the total order indices is noticeable. In addition, the probability density functions become significantly flatter between 64 and 128 for most samplers, indicating a higher probability of even smaller errors with more than 64 samples. The positive trend is not continued with even more samples. The error rates of first order indices seem not be significantly influenced by the initial sampling. The computation time has been omitted from this plot, as it was not affected by the initial sampling.

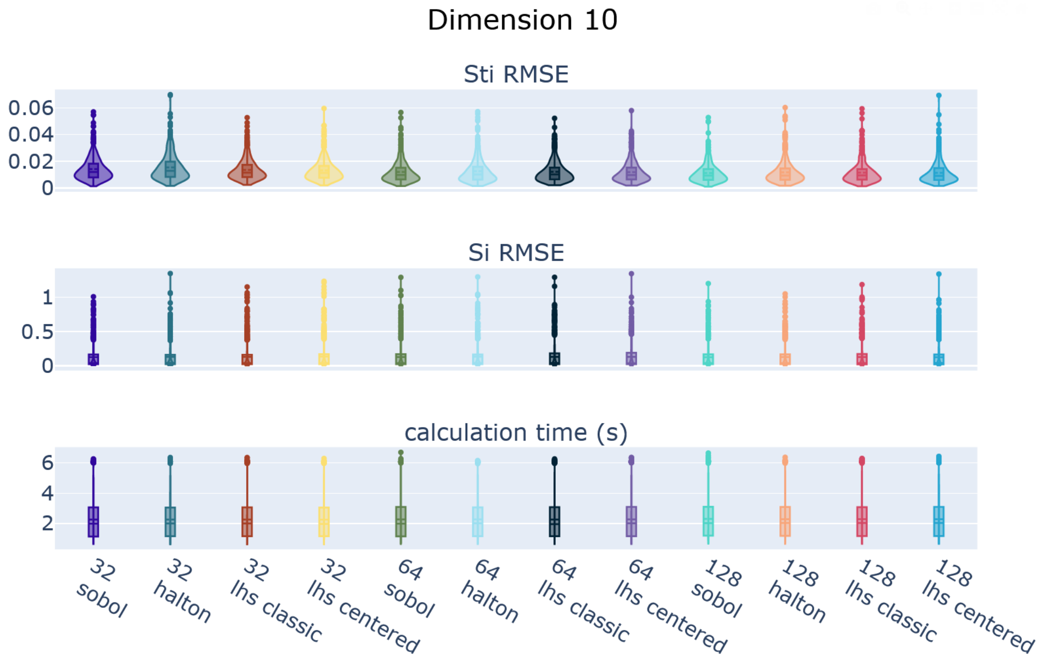


Fig. B.1.: Violin plots showing the effects of the **initial sampling** method and the number of samples on the error rates of the indices and the computation time for dimension 10

In contrast to the initial sampling, the local FLA showed larger impact on both, errors and computation time. Figure B.3 shows the results for 10 dimensions. The errors

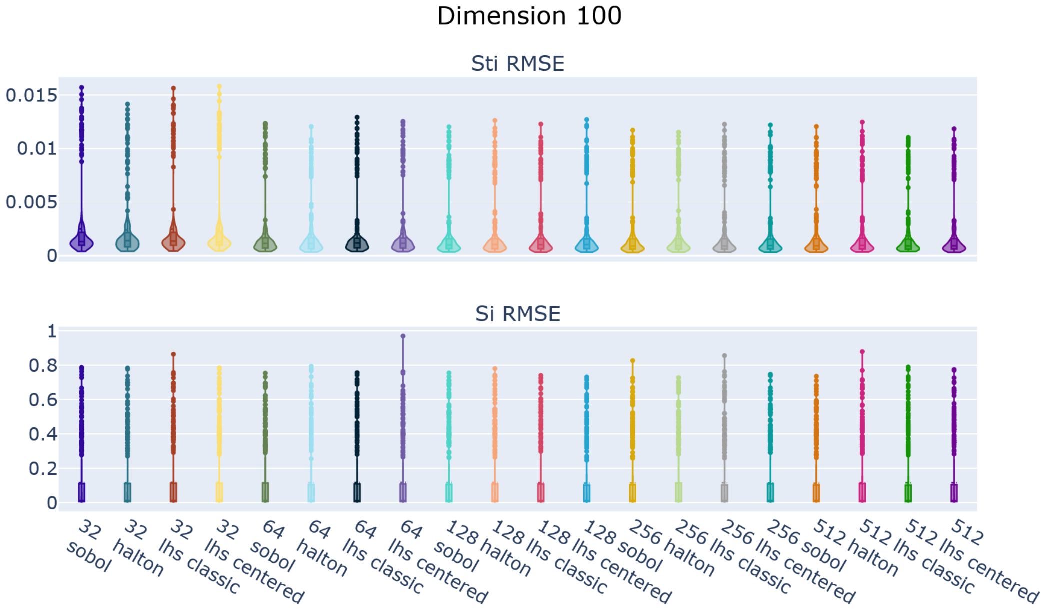


Fig. B.2.: Violin plots showing the effects of the **initial sampling** method and the number of samples on the error rates of the indices and the computation time for dimension 100 with extended number of initial samples

for both indices show a clear downward trend as more samples are used, while at the same time the computation time increases. If a worst case calculation time of 6 seconds is acceptable for the use cases, the largest possible number of samples would surely be chosen for a small number of dimensions. No clear favorite can be identified in the sampling method. They seem to have little influence on the calculation time. For the total order indices, the LHS variants perform slightly better, for the first order indices Halton.

However, the problem instances with 100 dimensions present a different result. Figure B.4a shows the results for all combinations of sampling techniques and the number of samples. Here the runs with Halton sampling produced inferior results. Removing the runs with Halton and those with 264 samples leaves the results in fig. B.4b. The two LHS variants perform equally well. Again, the clear relationship between the number of samples and the reduction of errors is evident. In order to select a suitable number of samples, it is necessary to evaluate which calculation time is still appropriate and to choose the largest possible number accordingly. The appropriate computation time is of course dependant on the use case. When considering the computation times displayed in fig. B.4b, it must be taken into account that the computation in the experimental setup was not performed in parallel, since all agents were running on one process core (no computer with 100 cores available). The time required for communication in spatially distributed systems was in turn not needed. The computation times are therefore a rather rough estimate and are only intended to serve as an indicator of the increased computational effort. Another

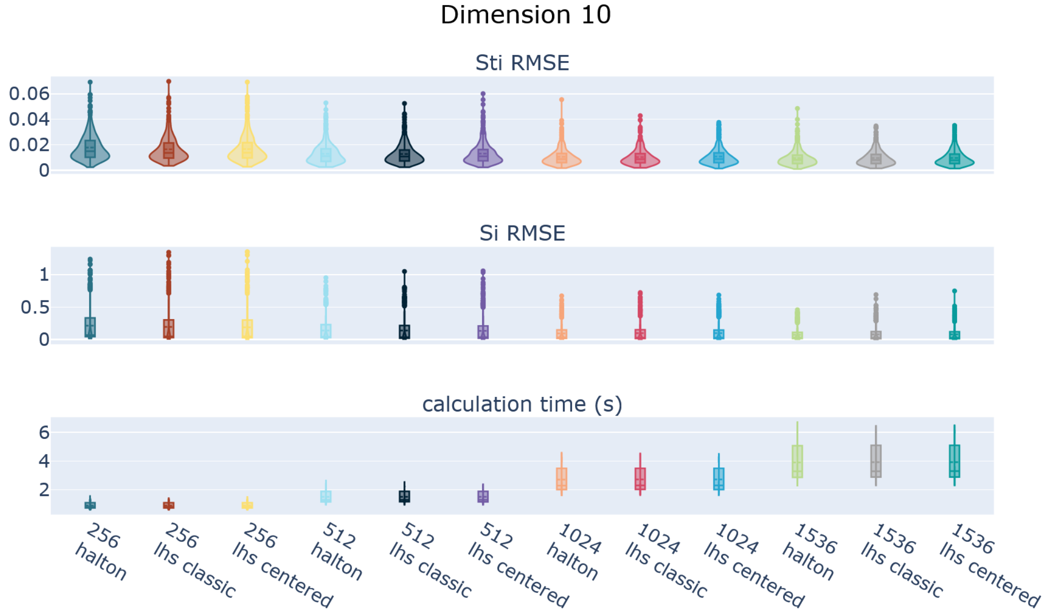


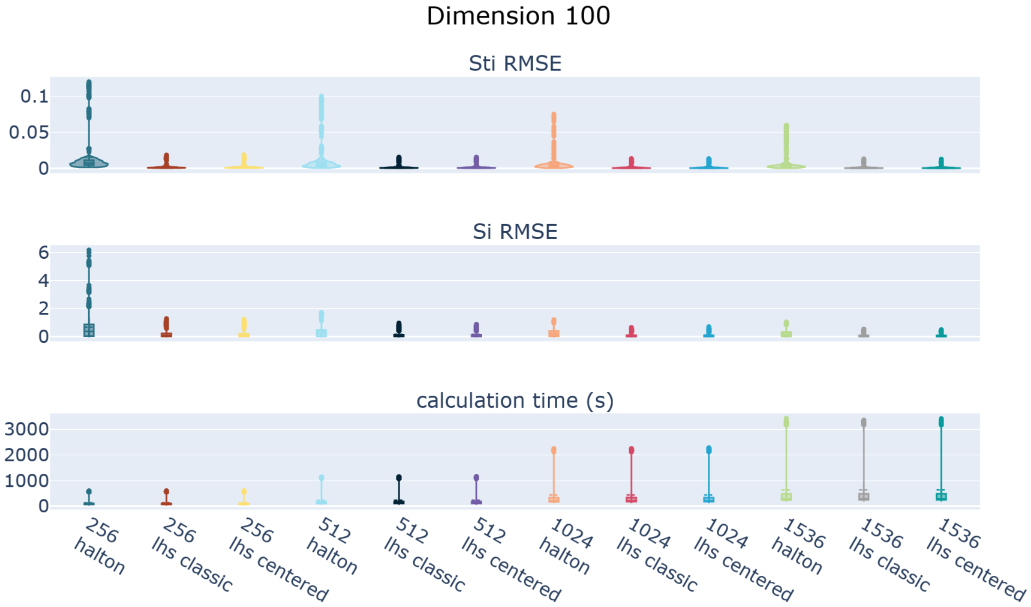
Fig. B.3.: Violin plots showing the effects of the sampling method and the number of samples for local FLA on the error rates of the indices for dimension 10

important aspect at this point are also the computational resources. The devices in the field on which the agents run have only limited capacities. The multiplication of large matrices, as required for the calculation of the sensitivity indices, can be overstraining for such a device. Especially if the calculations for the FLA is not the only parallel running task of the agent.

Figure B.5 shows the same results as fig. B.4b, but instead of the distribution as violin plots uses scatter plots over all problem instances (x-axis). Once again, this shows that a higher number of samples reduces the errors for both indices. However, it also shows that the error size is influenced most by the problem itself. All problem instances located in the upper point cloud with the largest errors at S_{ti} are composite spaces of Salomon. All problem instances in the point cloud with the highest errors at S_i are composite spaces of Salomon and Schaffer F6. The same holds for the computation times. The extreme Points in fig. B.5c all belong to composite spaces of the Sargan function. Excluding Sargan, fig. B.5d is retrieved.

In summary, the parameter evaluation showed that initial sampling has only a minor impact on the accuracy of the indices and the computational effort with the current setup. From a number of 128 samples on, no significant improvement was observed. However, the parameterization for local FLA shows a clear correlation between the number of samples and the quality of the results as well as the computational effort. Especially for higher dimensional problems, there is also a clear difference between different sampling techniques. Table B.1 gives an overview of the findings and the chosen values for the parameters used in the further course of this work.

(a) Full set of results. Errors for runs with Halton sampling stand out.



(b) Reduced result set excluding runs with Halton sampling and the ones with 256 samples

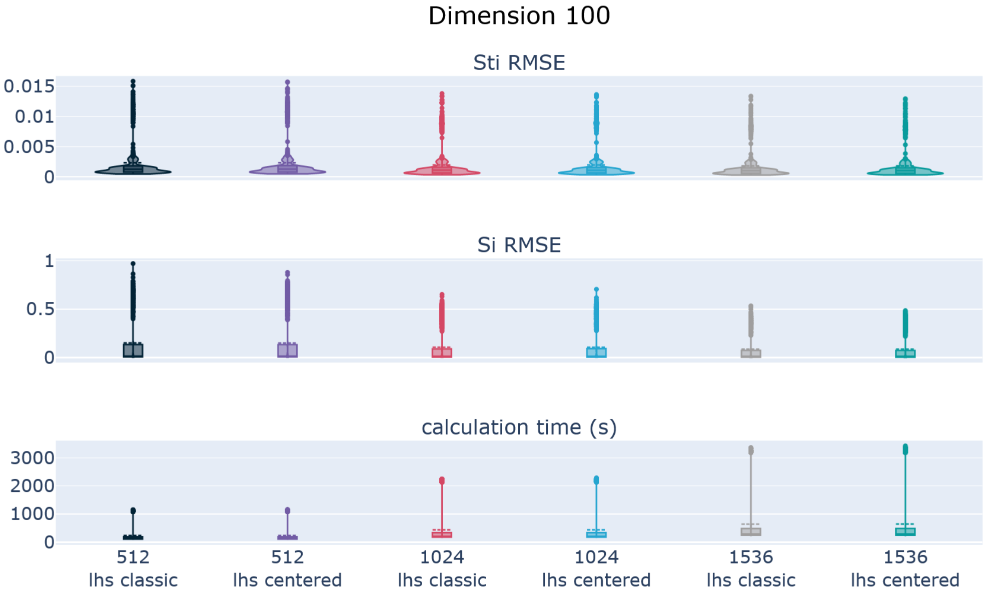
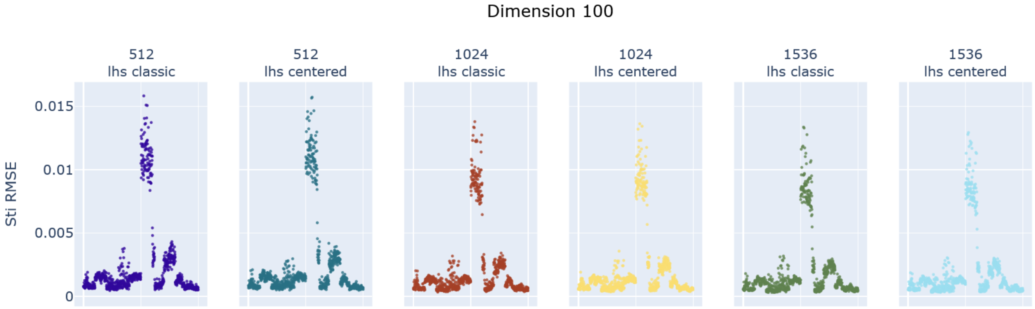
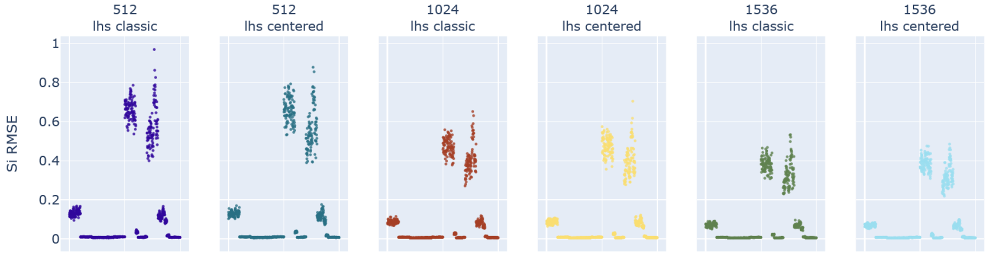


Fig. B.4.: Violin plots showing the effects of the sampling method and the number of samples for local FLA on the error rates of the indices for dimension 100.

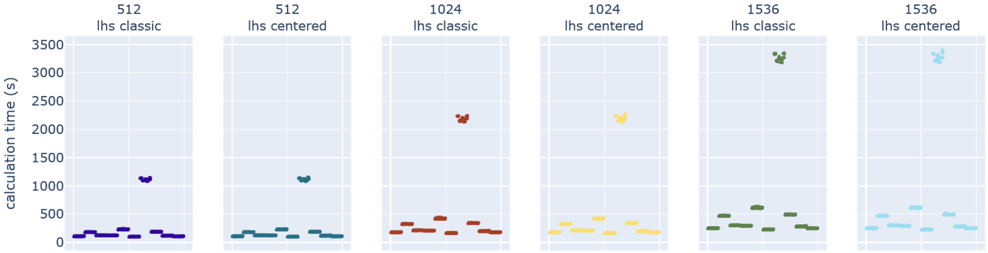
(a) S_{ti}



(b) S_i



(c) calculation time



(d) calculation time without sargan

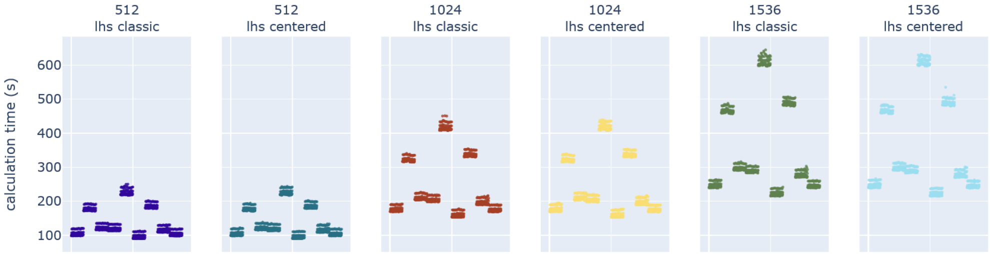


Fig. B.5.: Scatter plots showing how sampling methods and the number of samples for local FLA perform across the used problem instances for dimension 100.

B.1.2 Distributed computation of fitness distributions

The parameter tuning for fitness distribution based metrics follows the same setup as for sensitivity based metrics (see appendix B.1.1 and table 3.1). First the effect of the initial sampling parameters and then the effect of the local FLA are examined. In each case, the sampling method and the number of samples are varied. For reference, the calculation of the metrics is additionally performed centrally. Since the mean value over the metrics calculated for the subspaces should be close to the centrally calculated metric over all search spaces, the deviation from this reference value is used below as a criterion for parameter selection:

$$\begin{aligned} \text{deviation fitness variance:} \quad & \boldsymbol{\mu_2(y)_{dev}} && = |\mu_2(y)_{mean} - \mu_2(y)_{ref}| \\ \text{deviation state variance:} \quad & \boldsymbol{\mu_2(||d||)_{dev}} && = |\mu_2(||d||)_{mean} - \mu_2(||d||)_{ref}| \\ \text{deviation mean state distance:} \quad & \boldsymbol{\mu(||d||)_{dev}} && = |\mu(||d||)_{mean} - \mu(||d||)_{ref}| \end{aligned} \tag{B.3}$$

Results

Figure B.6 and fig. B.7 show the results of varying the initial sampling on 10 D and 100 D problemin instances. The initial sampling does not appear to have a significant impact on the deviation from the centrally calculated metrics, Thus, the parameters proposed for the distributed sensitivity analysis can be adopted.

Whereas, the parameters of local FLA do show an impact on the quality of the results of the metrics as depicted in fig. B.8 and fig. B.9. As expected, the computational effort increases with the number of samples. Interestingly, this increase was considerably smaller for centered LHS in the case of 100-dimensional problem instances. The effects of varying parameters on the deviations of the three fitness and state distribution based metrics differ and therefore need to be considered individually.

Regarding the deviations of **fitness variances**, a clear downwards trend is visible for 10 dimensions. In the higher dimensional case this is not observable, but the deviations decrease compared to the lower dimensional case. It is noteworthy that some of the setups with just 265 samples exhibit the smallest deviation. The sampling method itself does not seem to have major influence on the variances, although for dimension 10 Sobol sampling shows slightly better results. In general, however, the deviations in this metric tend to be small, and some variation is to be expected due to the stochastic nature of the sampling process.

The **deviation of the state variances** shows a different picture. In general, the violin plots rather resemble box plots, since most values show small deviation and some

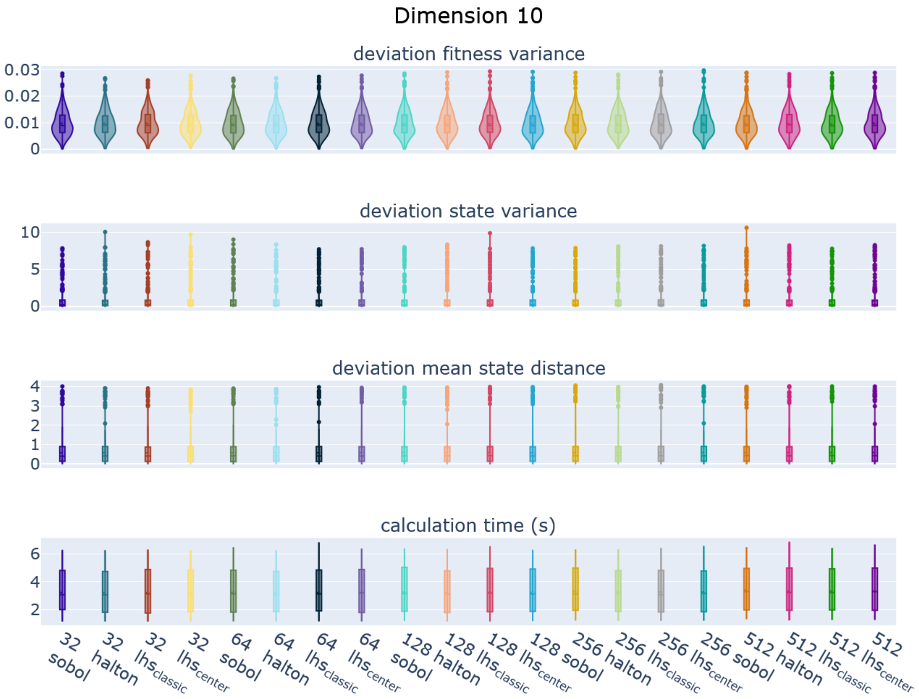


Fig. B.6.: Violin plots showing the impact of the **initial sampling** methods and the number of samples on the deviation of fitness and state variance based metrics from centrally computed counterparts for dimension 10.

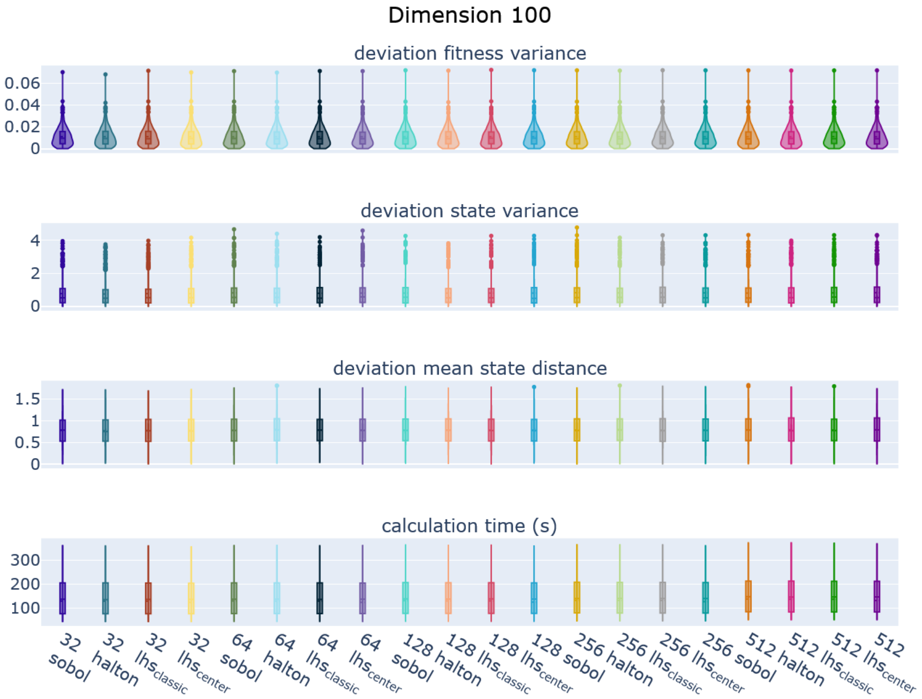


Fig. B.7.: Violin plots showing the impact of the **initial sampling** methods and the number of samples on the deviation of fitness and state variance based metrics from centrally computed counterparts for dimension 100.

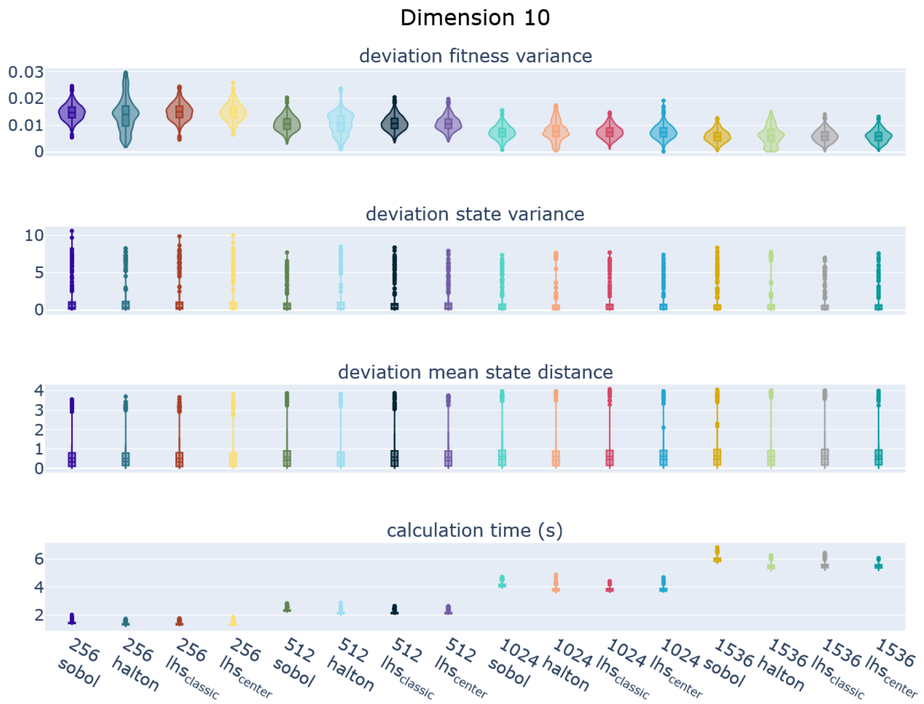


Fig. B.8.: Violin plots showing the impact of the sampling method and the number of samples for **local FLA** on the deviation of fitness and state variance based metrics from centrally computed counterparts for dimension 10.

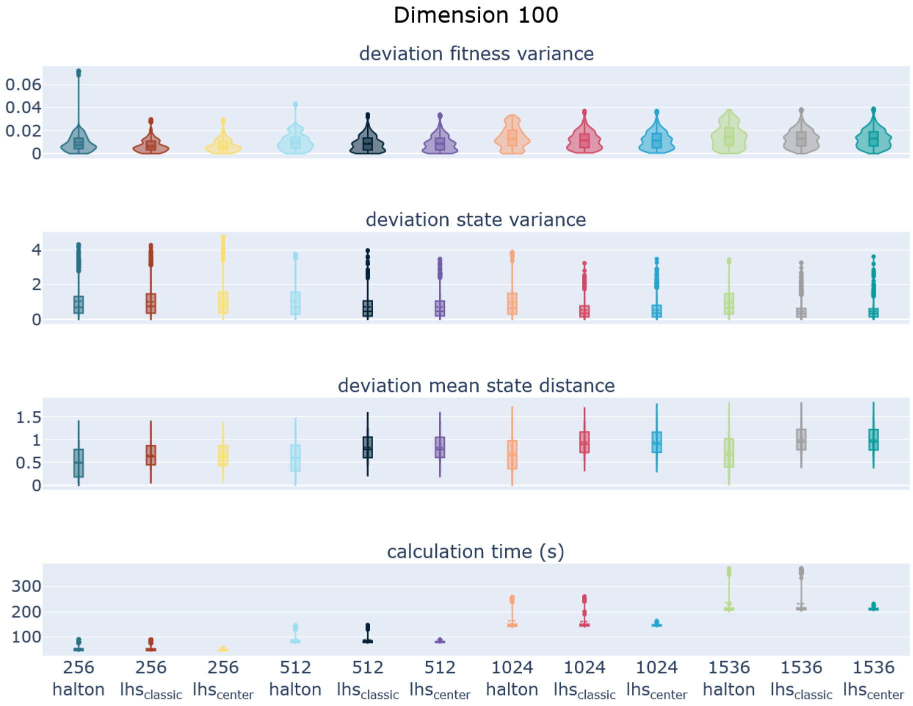


Fig. B.9.: Violin plots showing the impact of the sampling method and the number of samples for **local FLA** on the deviation of fitness and state variance based metrics from centrally computed counterparts for dimension 100.

outliers dominate the impression. However, when looking at the 100 dimensional runs, with an increase in the number of samples, a decrease in the height of the boxes and a decrease in the number and magnitude of the outliers becomes apparent. Here, the Halton sampling compares poorly.

The **deviation of the mean state distances** shows hardly any difference for the different parameterizations in the 10-dimensional setups. For the 100-dimensional setups, the number of samples again has no discernible effect. For the sampling method, on the other hand, differences are noticeable, with Halton Sampling producing slightly better results.

The following can therefore be stated as an overall summary: The parameters for initial sampling showed no major impact on the deviations. Thus, parameters can be set according to requirements of other FLA metrics. For local FLA, the number of samples had a positive effect on the deviations of some metrics, but also resulted in a higher computational effort. This increased effort seems do be less relevant when centered LHS is used. In summary, centered LHS with 1024 samples seems to be a reasonable compromise, which is also consistent with the setting that is most appropriate for distributed sensitivity analysis.

B.1.3 Distributed computation of structure exploring metrics

The parameter tuning of the SE-FLA metrics involves the same parameters as the previous two feature groups, namely the sampling method and the number of samples, for the initial sampling and the local FLA. In addition, the random walks for the local FLA are repeated a number of times to compensate for random effects. The number of repetitions is therefore another relevant parameter (see table 3.1). The figures B.10,B.11, B.12, B.13, B.14 and B.15 show the deviation for the centrally and distributedly calculated FEM with different parameter combinations for the initial sampling and the local FLA. The FEM is taken as exemplary metric, but the impression remains the same for all SE-FLA metrics. In addition, for the 100-dimensional problems, only a reduced set of parameter values (e.g., the largest and smallest number of samples) was used for the calculation after no effect was observed for the 10-dimensional problems. The influence of the parameters concerning the sampling is very small with respect to the calculated values. The deviations are almost equal, since the calculated values differ only by a few decimal places. Consequently, the violin plots look virtually identical. The initial sampling method and number of samples in fig. B.10 and B.11 do not even influence the computational effort. Regarding the local FLA, the sampling method is also negligible. The number of samples, however, and the number of repetitions of the entire computation procedure affect the computational effort, as shown in figures B.12, B.13, B.14 and

B.15. Since they do not seem to have influence on the calculated values, the most sensible conclusion is to choose the smallest number in each case and thus the least required computational effort.

Since the repetition of the overall computation procedure increases the computational effort immensely and 30 repetitions did not show different results than 10 repetitions, the experiments where repeated with even less repetitions in the distributed computation, namely 5 and 1 as shown in fig. B.15. A possible explanation is that, based on the recombined samples of the other agents, many random walks have already been performed in different versions of the local search space. Thus, further repetitions of the random walks are unlikely to add further value.

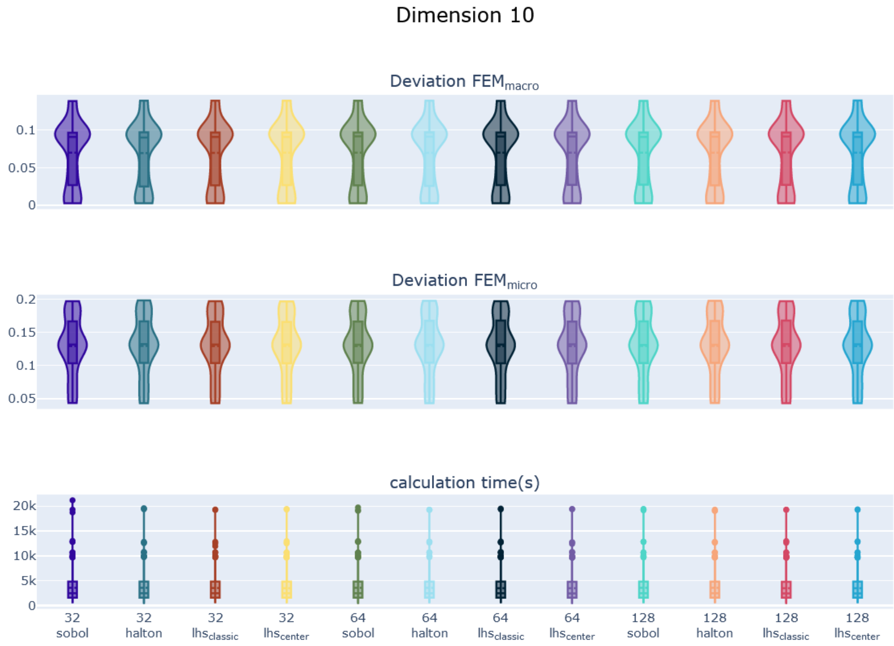


Fig. B.10.: Violin plots showing the impact of the **initial sampling** methods and the number of samples on the deviation of FEM from centrally computed counterpart for dimension 10.

In order to obtain a more comprehensive impression of the distributed computation of the SE-FLA metrics, the values for a composite space in 10-D of each benchmark function are shown in figures B.16 and B.17. These are scatter plots in which a separate data point is plotted for each combination of sampling parameters (sampling method and number of samples for both phases and number of repetitions). However, in most cases these points are so close together that only one point is visible. Three calculations with 30 repetitions were performed for the centrally calculated reference points. FEM and PIC as measures for ruggedness and modality are in all cases slightly underestimated. The SEM as measure of smoothness is also underestimated on micro scale but mostly overestimated on macro scale. The distributed calculation of the dispersion metric always results in a lower values, which indicate single funnel landscapes. It seems reasonable that the global shape of the landscape is harder to

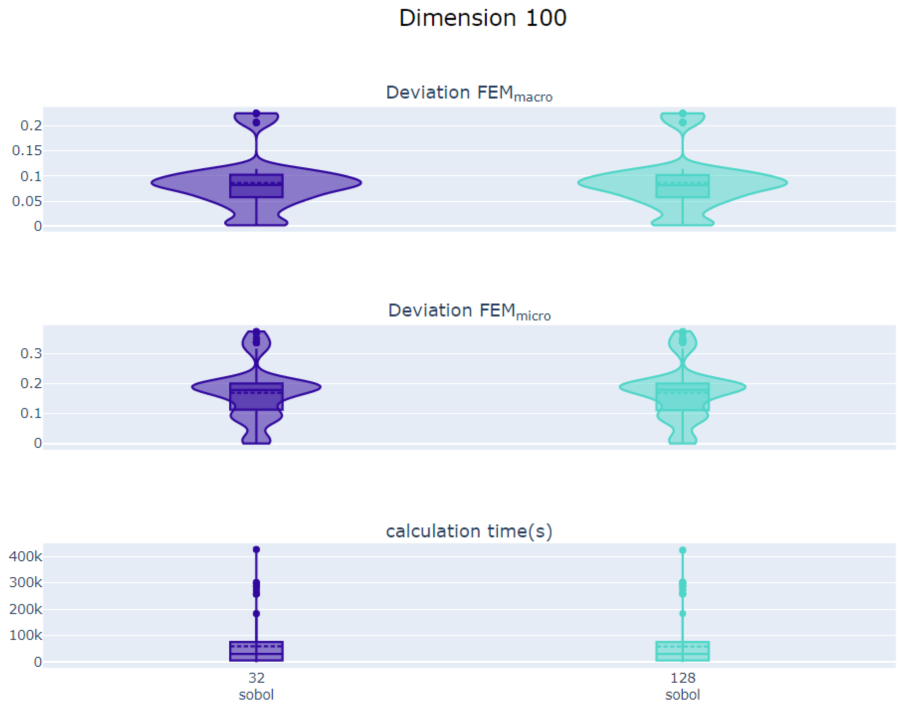


Fig. B.11.: Violin plots showing the impact of the **initial sampling** methods and the number of samples on the deviation of FEM from centrally computed counterpart for dimension 100.

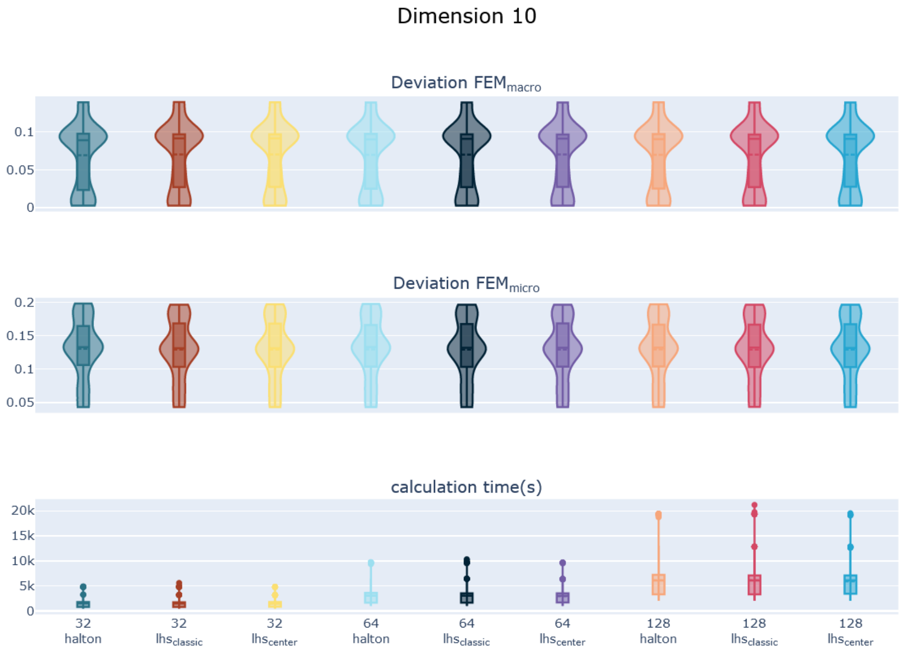


Fig. B.12.: Violin plots showing the impact of the sampling method and the number of samples for **local FLA** on the deviation of FEM from centrally computed counterpart for dimension 10.

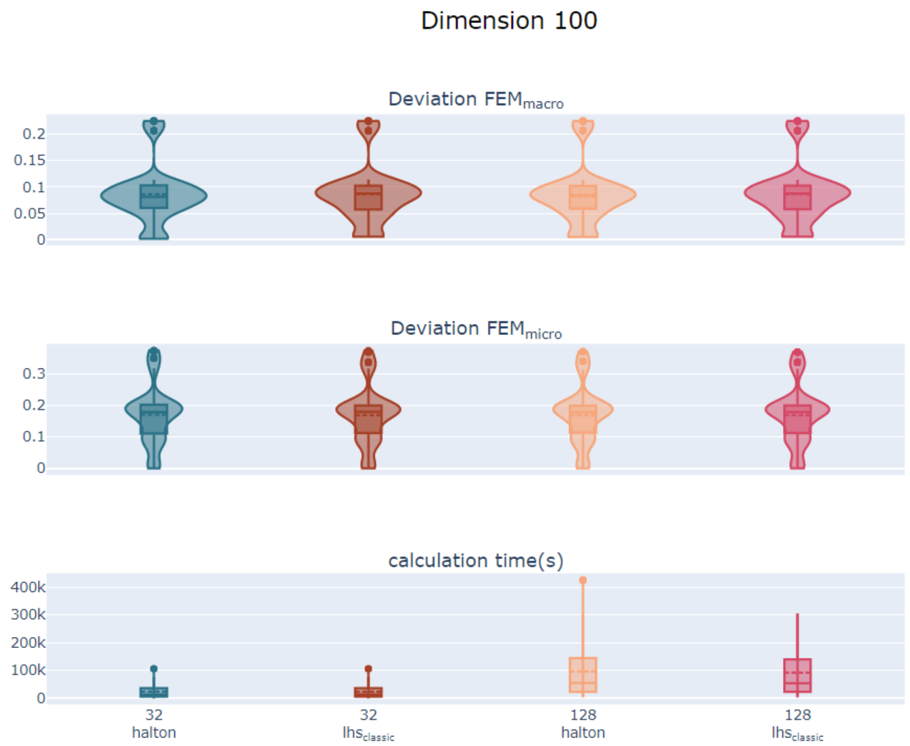


Fig. B.13.: Violin plots showing the impact of the sampling method and the number of samples for **local FLA** on the deviation of FEM from centrally computed counterpart for dimension 100.

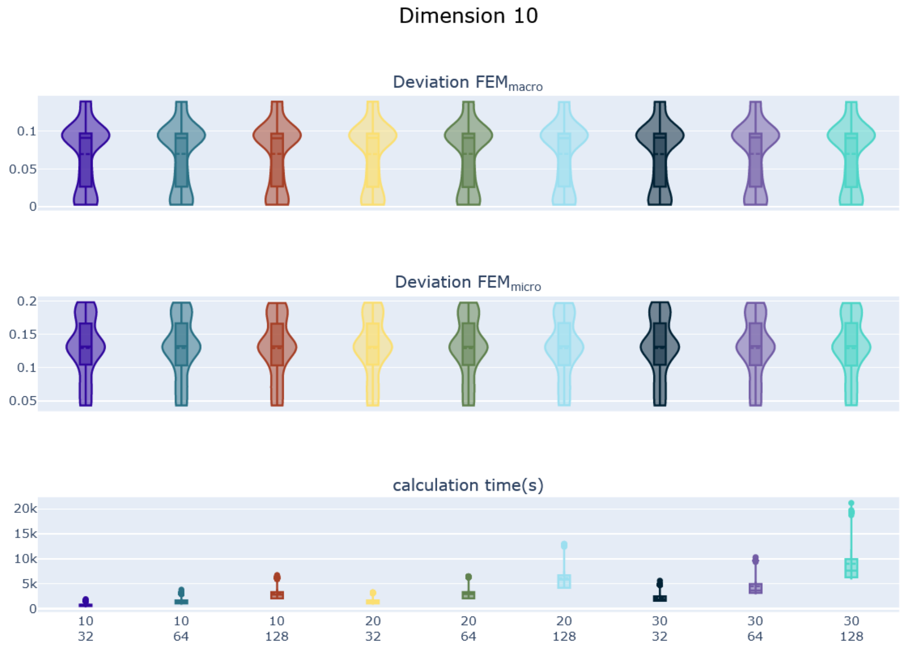


Fig. B.14.: Violin plots showing the impact of the the number of samples for **local FLA** and the number of repetitions of the local FLA on the deviation of FEM from centrally computed counterpart for dimension 10.

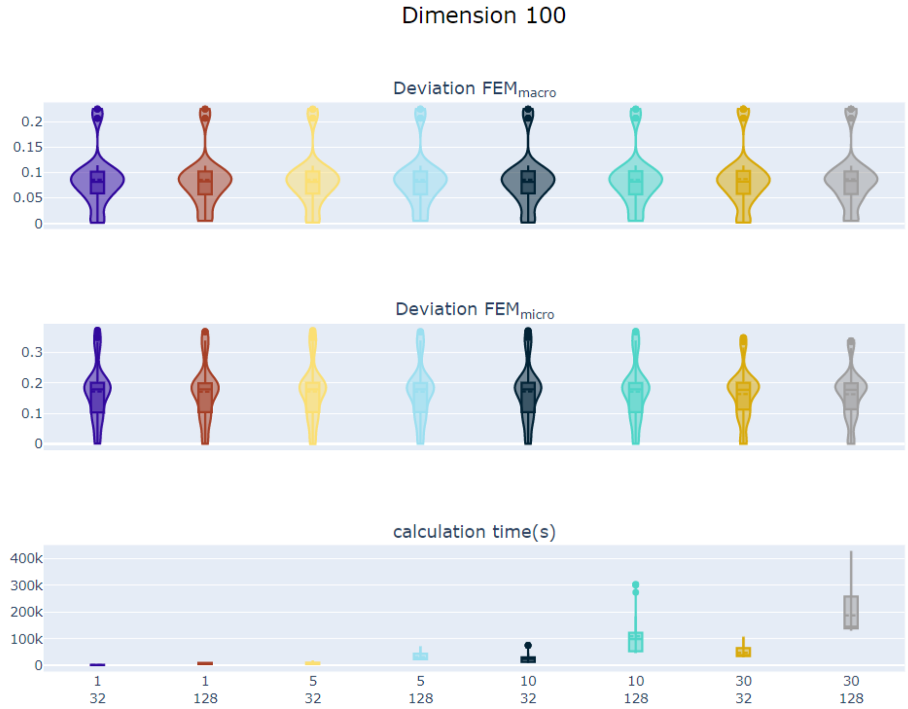
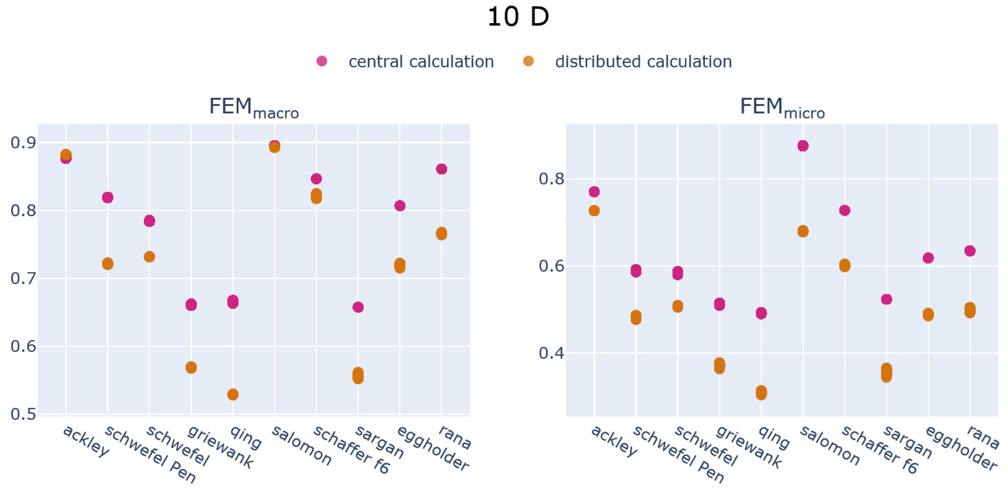


Fig. B.15.: Violin plots showing the impact of the the number of samples for **local FLA** and the number of repetitions of the local FLA on the deviation of FEM from centrally computed counterpart for dimension 100.

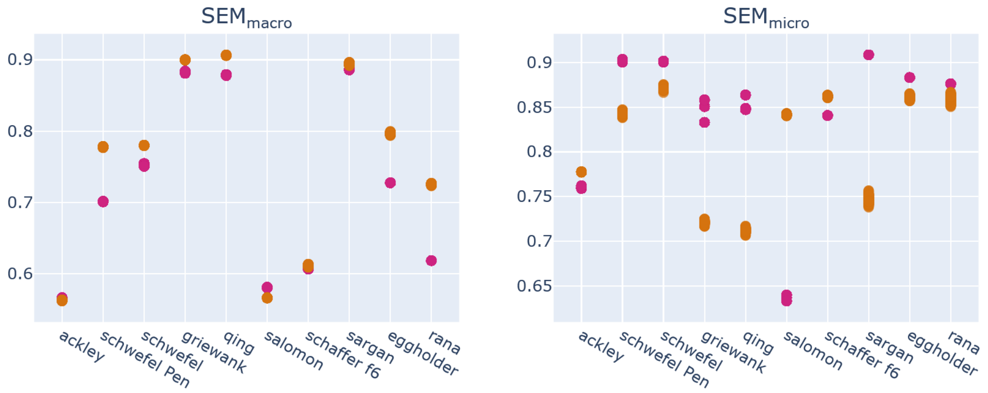
perceive in the local calculation due to the limited perspective. The gradient-based metrics are very close to the centrally calculated values for most functions. The exceptions are Salomon, Ackley and Schaffer F6, where the values are estimated to be significantly higher in the distributed calculation. These are the same functions for which the first order effects in the distributed and the central calculation deviate strongly from each other. The results for 100-D problem instance show the same picture.

In general, the deviations are similar for each metric when compared across multiple problem instances (i.e., composite spaces). This could indicate that they properly represent the distributed view of the problem or that there is a relatively constant bias compared to the central computation. Either way, this seems to provide comparability among problem instances, suggesting suitability for the metrics' actual purpose, namely, problem-specific parameterization of distributed optimization heuristics.

(a) FEM as measures for ruggedness on micro and macro scale (maximum step size of 1% and 10% of the search space)



(b) SEM as measures for smoothness on micro and macro scale (maximum step size of 1% and 10% of the search space)



(c) PIC as measures for modality on micro and macro scale (maximum step size of 1% and 10% of the search space)

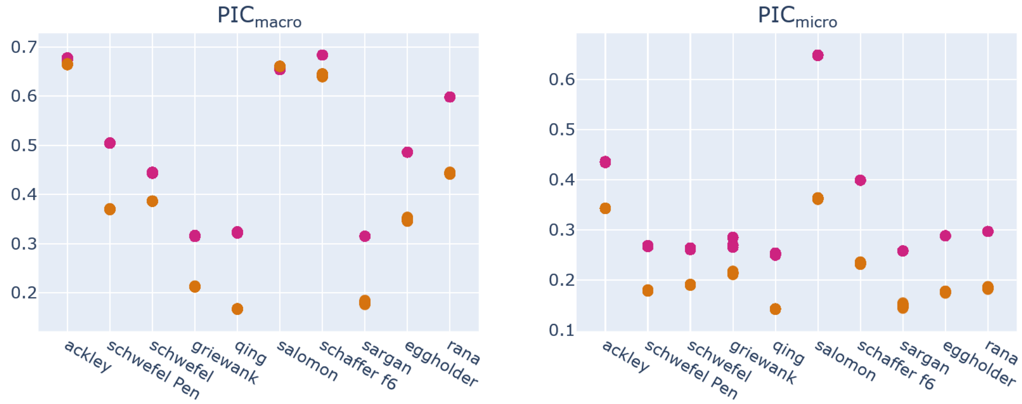
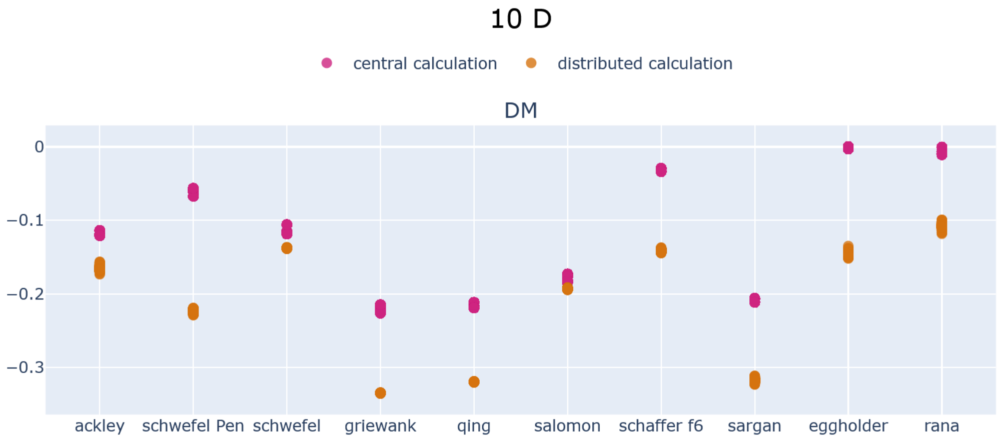


Fig. B.16.: Comparison of central and distributed calculation of information theoretic measures for 10-D problem instances

(a) Dispersion metric that indicates the presence of funnels



(b) gradient measures

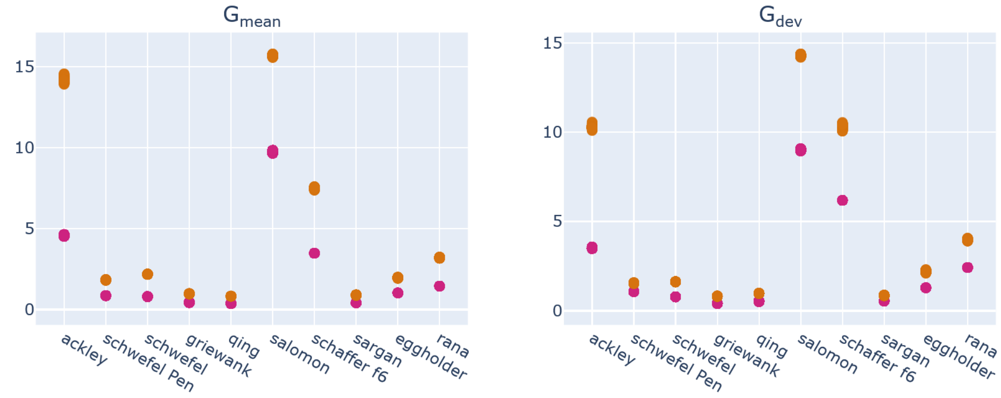


Fig. B.17.: Comparison of central and distributed calculation of gradient based measures and the dispersion metric for 10-D problem instances

B.1.4 Final parameter setup

Table B.1 summarizes the findings on the impact of the parameters and the consequently chosen values for each parameter.

phase	**parameter**	**findings**	**chosen value**
initial sampling	1.1: sampling method	**SA-FLA**: No detectable influence for small number of decision variable per agent. Thus any sampling method (sobol, halton, LHS) may be used **FSD-FLA**: No detectable influence on deviation from centrally computed reference values **SE-FLA**: No detectable influence on deviation from centrally computed reference values	LHS classic
	1.2: number of samples	**SA-FLA**: 128 is sufficient, but larger numbers do not increase computational times significantly and could be taken as well if beneficial for other metrics **FSD-FLA**: No detectable influence on deviation from centrally computed reference values **SE-FLA**: No detectable influence on deviation from centrally computed reference values	128

phase	**parameter**	**findings**	**chosen value**
local FLA	2.1: sampling method	**SA-FLA**: For a small number of samples, Halton sampling yields slightly better results for first order indices. For larger problem dimensions, the two LHS variants considerably outperform Halton. **FSD-FLA**: Different sampling methods have slight advantages or disadvantages for different metrics and problem dimensions. In terms of deviations from the reference values, no method stands out in particular. In terms of computational effort, LHS centered is clearly preferable, especially for larger system sizes. **SE-FLA**: No detectable influence on deviation from centrally computed reference values	Halton for 10-D LHS centered for 100-D
	2.2: number of samples	The more samples, the better the results, but at the cost of computational effort. Largest possible value should be chosen depending on the use case. **SA-FLA**: 1024 is sufficient **FSD-FLA**: 1024 is sufficient **SE-FLA**: No detectable influence on deviation from centrally computed reference values but on the computational effort. Thus, the lowest number is appropriate.	1024 for LDS 32 for SE-FLA
	2.3: number of repetitions	**SE-FLA**: No detectable influence on deviation from centrally computed reference values but on the computational effort. Thus, the lowest number is appropriate.	10

Tab. B.1.: Overview of the findings about the impact of parameters and the chosen values for FLA features

B.2 Correlation Analysis of distributed FLA metrics

In section 3.7, the correlation of all distributed FLA metrics was calculated using the Spearman correlation coefficient and clustered by applying the UPGMA algorithm [Mül11]. Figures B.18, B.19, and B.20 show, for each cluster, the values of the included metrics across all problem instances in shared scatter plots. For clusters 9 - 14, the correlation of the clustered metrics is clearly visible, as the relation of the values between different problem instances is very similar. For cluster number 15, high spikes occur for several functions (*Alpine 2, Drop Wave, Mishra 11, Rastrigin and Xin She Yang 1*). First, these extremely high values for the state variance metrics show that for these functions the differences between different subsearch spaces in a composite space are large (high standard deviations). Second, both the variances and the mean values within the subsearch spaces for the mean distances of points with similar fitness are large. Overall, this indicates very heterogeneous setups. If these functions are omitted from the analysis (fig. B.20b), it is easier to see that all 4 metrics show spikes for the same problems, but at different magnitudes. For the *Sargan* function, the increase of $\mu(||d||)\ mean$ is strikingly high compared to the other metrics. This could be due to the fact that *Sargan* is a bowl shaped function with high edges and similar fitness values can therefore be very far apart. At the same time, its shape changes little when the subspaces are created, which makes the heterogeneity comparatively small. Moreover, $\mu(||d||)\ mean$ is the only one of the 4 metrics that is based solely on mean values and does not include variance or standard deviation. Nevertheless, the 4 metrics show an overall similar behavior when considering all problem instances.

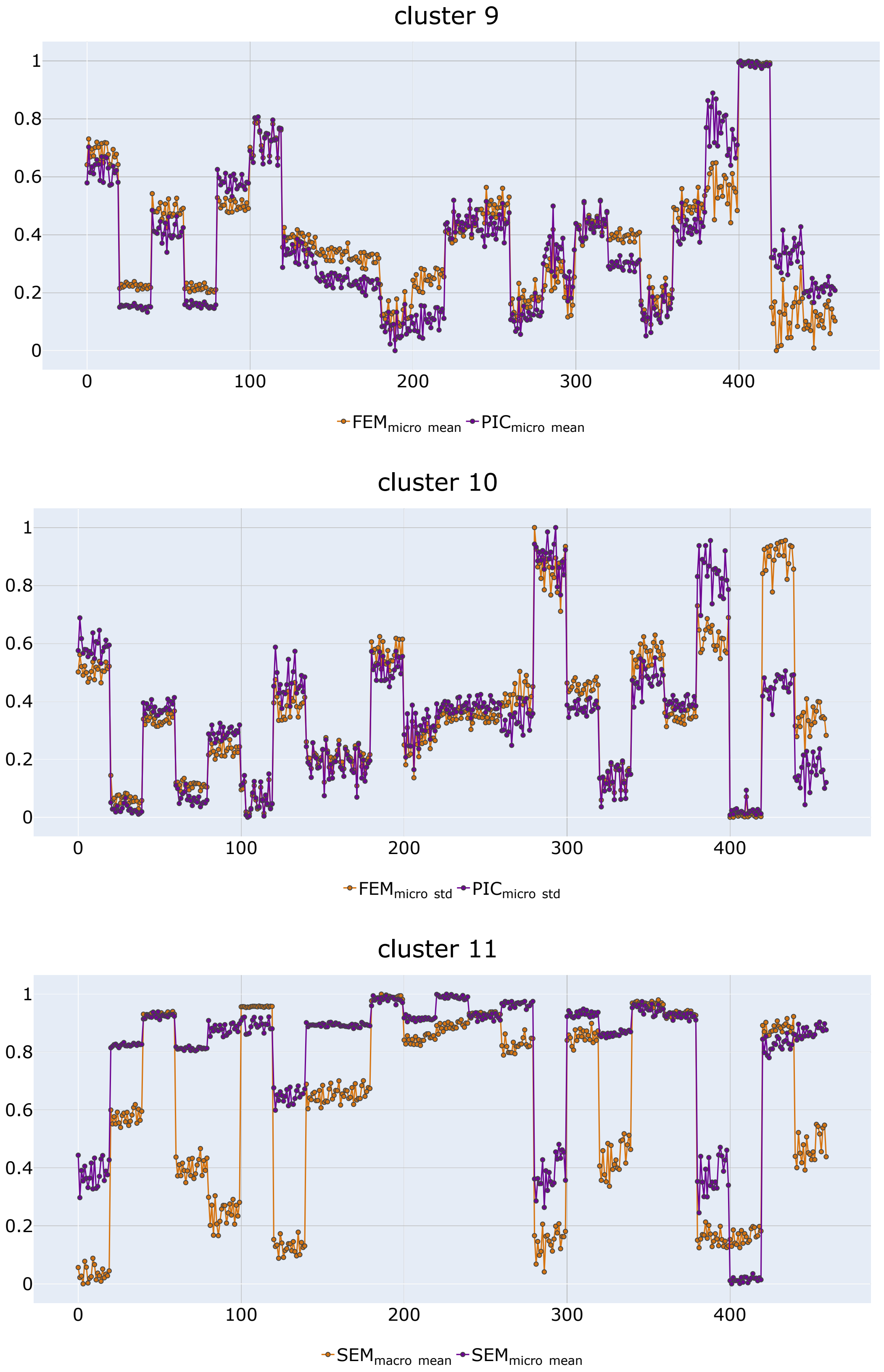


Fig. B.18.: Clusters 9 - 11: values of clustered metrics across problem instances

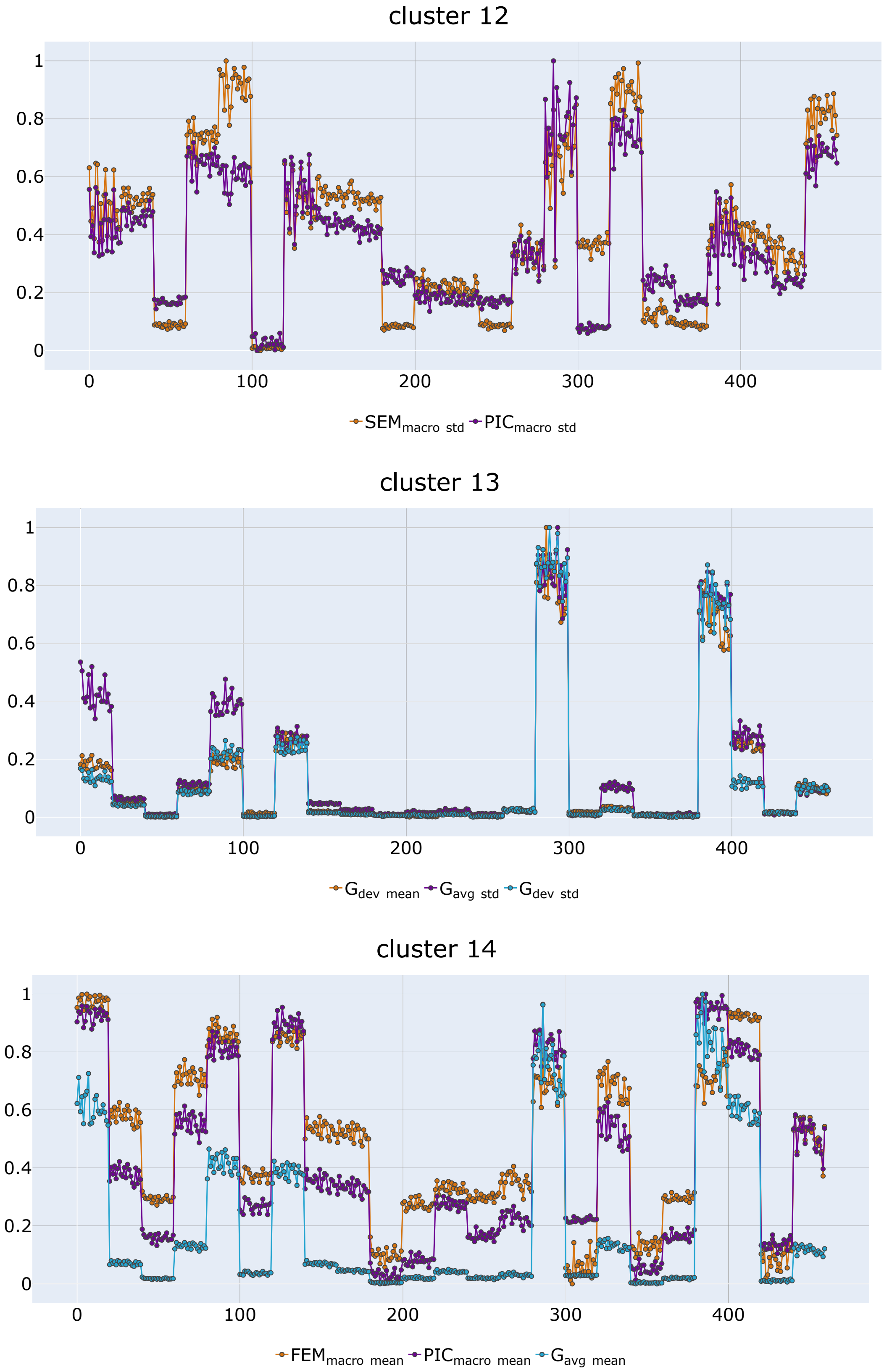
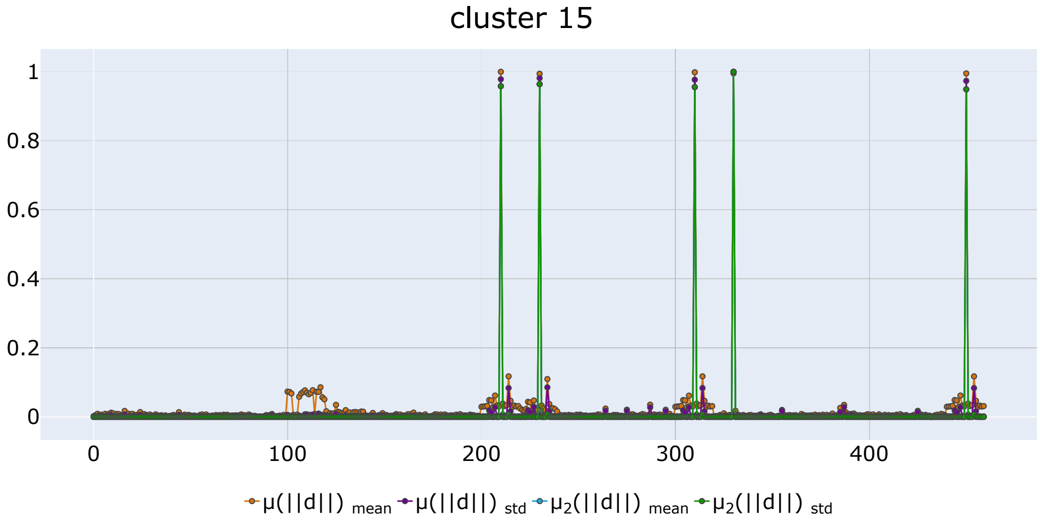


Fig. B.19.: Clusters 12 - 14: values of clustered metrics across problem instances

(a) Full set of problem instances



(b) reduced set of problem instances (and rescaled)

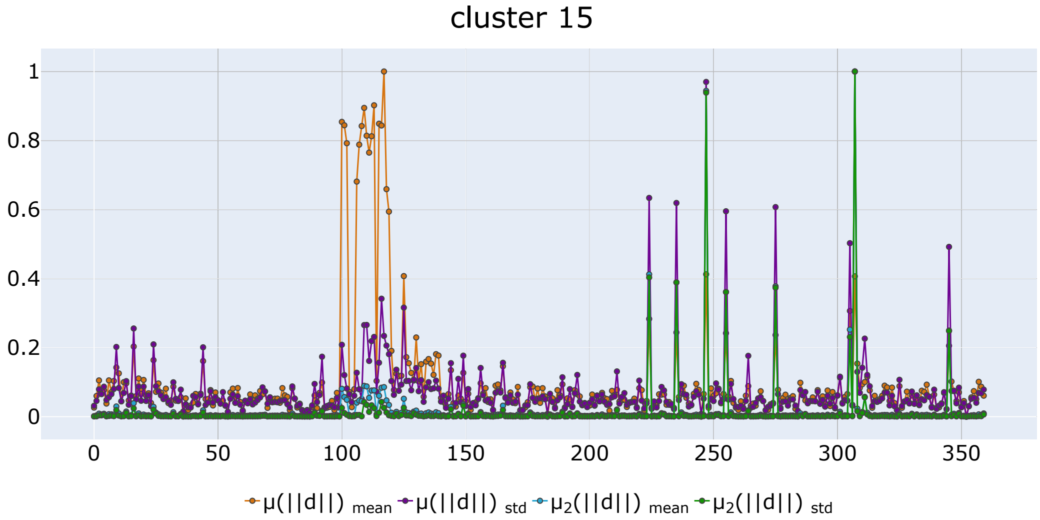


Fig. B.20.: Cluster 15: values of clustered metrics across problem instances

Appendix C

C

This part of the appendix presents additional data and figures related to chapter 6 and thus to the evaluation of the importance of features or the evaluation of the predictive ability of ML models.

C.1 Correlation Analysis

In section 6.2.2, a correlation analysis was performed between the distributedly computed FLA metrics and the performance metrics. This analysis also distinguished between different CTVs.

C.1.1 Interpretation of correlation coefficients

The following tables C.1, C.2, C.3 and C.4 list the proportionally highest positive and negative correlations and discuss the resulting implications. A bold marking of the heuristic name indicates a relatively high coefficient for the metric compared to other coefficients for that heuristic.

v_{cv} - **COHDA (-0.32)** - IDICE (-0.18)	Coefficient of variation in variable sensitivity $\uparrow f(x)_{norm}$ (lower solution quality) correlates with $\downarrow v_{cv}$: degree of influence by the variables becomes more balanced $\downarrow f(x)_{norm}$ (higher solution quality) correlates with $\uparrow v_{cv}$: degree of influence by the variables becomes more unevenly distributed
$SEM_{macro_{mean}}$ - **COHDA (-0.31)** - **IDICE (-0.39)**	mean value of smoothness over all sub-search spaces $\uparrow f(x)_{norm}$ (lower solution quality) correlates with $\downarrow SEM_{macro_{mean}}$: lower average smoothness $\downarrow f(x)_{norm}$ (higher solution quality) correlates with $\uparrow SEM_{macro_{mean}}$: higher average smoothness

Tab. C.1.: Proportionally high **negative** correlation for $f(x)_{norm}$; A bold marking of the heuristic indicates a relatively high coefficient for the metric compared to other coefficients for the heuristic. The numbers in parentheses are the average correlation coefficients for the FLA metric and the heuristic.

C.1.2 P-Values

During the correlation analysis in section 6.2.2, the p-values for the correlation coefficients were calculated. These p-values are an indication of how likely it is that the value of the correlation coefficient could have been obtained by chance. They are plotted in heatmaps similar to the heatmaps plotted in section 6.2.2 with the correlation coefficients, in figures C.1, C.2, C.3a and C.4.

C.2 Permutation Importance

In section 6.2.3, the permutation importance of the FLA metrics and CTV parameters was examined in relation to the prediction of the performance metrics. Tables C.5, C.6, C.7 and C.8 show the permutation importance scores used to generate the bar charts in section 6.2.3.

C.3 ML Model Parameters

The Random Forest regression models in section X were tuned with a random search for hyper parameter optimization. Table C.9 shows the final parameter settings used for the analyses in section 6.3.2 and section 6.3.3. The final parameter settings for section 6.3.4, chosen after applying noise to a subset of the training data, are shown in table C.10.

metric	**implication**
$\boldsymbol{DM_{mean}}$ - **COHDA (0.26)** - **IDICE (0.49)**	Dispersion metric mean $\uparrow f(x)_{norm}$ (lower solution quality) correlates with $\uparrow DM_{mean}$: more likely multi-funnel landscapes $\downarrow f(x)_{norm}$ (higher solution quality) correlates with $\downarrow DM_{mean}$: more likely single-funnel landscapes
$\boldsymbol{DM_{std}}$ - COHDA (0.17) - **IDICE (0.55)**	Dispersion metric standard deviation $\uparrow f(x)_{norm}$ (lower solution quality) correlates with $\uparrow DM_{std}$: greater heterogeneity with respect to global shapes across subsearch spaces $\downarrow f(x)_{norm}$ (higher solution quality) correlates with $\downarrow DM_{std}$: lower heterogeneity with respect to global shapes across subsearch spaces
$\boldsymbol{SEM_{macro_{std}}}$ - COHDA (0.11) - **IDICE (0.53)**	Standard deviations of smoothness and modality on macro scale $\uparrow f(x)_{norm}$ (lower solution quality) correlates with $\uparrow SEM_{macro_{std}}$: greater heterogeneity with respect to smoothness and modality across subsearch spaces $\downarrow f(x)_{norm}$ (higher solution quality) correlates with $\downarrow SEM_{macro_{std}}$: lower heterogeneity with respect to smoothness and modality across subsearch spaces
$\boldsymbol{FEM_{macro_{mean}}}$ - **COHDA (0.32)** - IDICE (0.266)	Mean values for ruggedness and modality on macro scale combined with average gradients $\uparrow f(x)_{norm}$ (lower solution quality) correlates with $\uparrow FEM_{macro_{mean}}$: greater mean ruggedness, modality and average gradients $\downarrow f(x)_{norm}$ (higher solution quality) correlates with $\downarrow FEM_{macro_{mean}}$: lower mean ruggedness, modality and average gradients

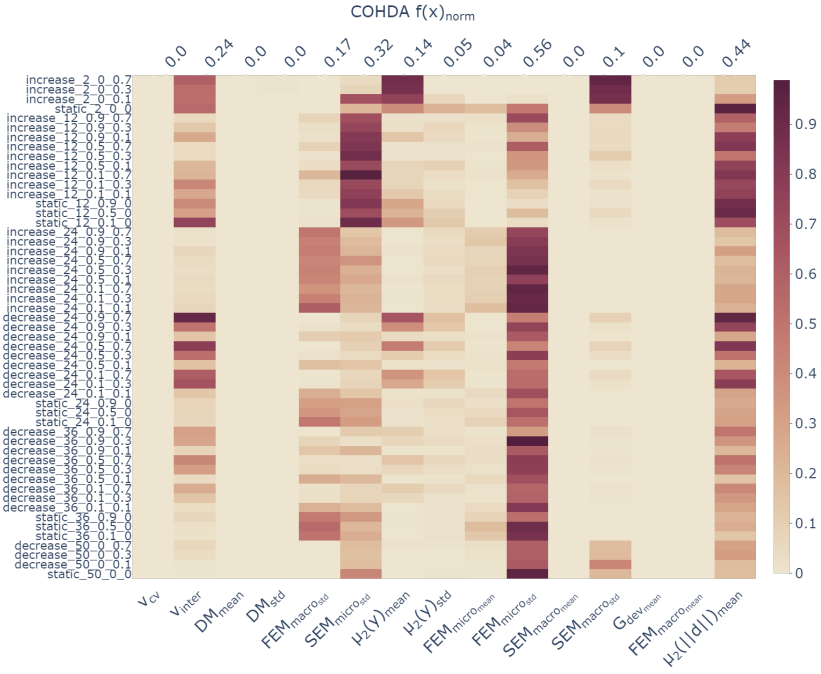
Tab. C.2.: Proportionally high **positive** correlation for $f(x)_{norm}$; A bold marking of the heuristic indicates a relatively high coefficient for the metric compared to other coefficients for the heuristic. The numbers in parentheses are the average correlation coefficients for the FLA metric and the heuristic.

$\boldsymbol{FEM_{macro_{mean}}}$ - COHDA - *decrease* - $\alpha = 0.7$ (0.40)	Mean values for ruggedness and modality on macro scale combined with average gradients $\uparrow |SE|_{norm}$ higher computational effort correlates with $\uparrow FEM_{macro_{mean}}$: greater mean ruggedness, modality and average gradients $\downarrow |SE|_{norm}$ lower computational effort correlates with $\downarrow FEM_{macro_{mean}}$: lower mean ruggedness, modality and average gradients
$\boldsymbol{v_{cv}}$ - IDICE - *decrease* (0.41)	Coefficient of variation in variable sensitivity $\uparrow |SE|_{norm}$ higher computational effort correlates with $\uparrow v_{cv}$: degree of influence by the variables becomes more unevenly distributed $\downarrow |SE|_{norm}$ lower computational effort correlates with $\downarrow v_{cv}$: degree of influence by the variables becomes more balanced
$\boldsymbol{v_{inter}}$ - IDICE - *decrease* - $k \neq 50$ (0.40)	Degree of variable interaction $\uparrow |SE|_{norm}$ higher computational effort correlates with $\uparrow v_{inter}$: higher degree of variable interaction $\downarrow |SE|_{norm}$ lower computational effort correlates with $\downarrow v_{inter}$: lower degree of variable interaction (towards separability)
$\boldsymbol{\mu(||d||)}$ - IDICE - *decrease* - $k \neq 50$ (0.41)	State distance and variance $\uparrow |SE|_{norm}$ higher computational effort correlates with $\uparrow \mu(||d||)$: higher state distances $\downarrow |SE|_{norm}$ lower computational effort correlates with $\downarrow \mu(||d||)$: lower state distances

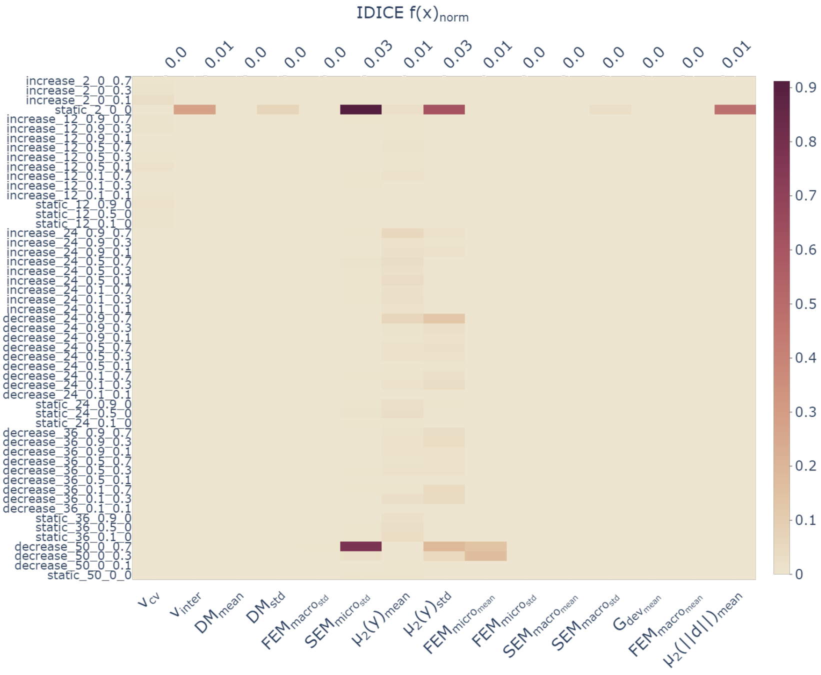
Tab. C.3.: Proportionally high **positive** correlation for $|SE|_{norm}$; The numbers in parentheses are the maximum correlation coefficients for the FLA metric and the annotated variants of the exchange topology in combination with the heuristic.

$\boldsymbol{v_{cv}}$ - COHDA - *decrease* - $k \neq 50$ - $\alpha = 0.7$ (-0.47)	Coefficient of variation in variable sensitivity $\uparrow \lvert SE \rvert_{norm}$ higher computational effort correlates with $\downarrow v_{cv}$: degree of influence by the variables becomes more balanced $\downarrow \lvert SE \rvert_{norm}$ lower computational effort correlates with $\uparrow v_{cv}$: degree of influence by the variables becomes more unevenly distributed
$\boldsymbol{FEM_{micro_{std}}}$ - COHDA - *decrease* (-0.44)	standard deviation over all sub-search spaces of ruggedness and modality on micro sale $\uparrow \lvert SE \rvert_{norm}$ higher computational effort correlates with $\downarrow FEM_{micro_{std}}$: lower heterogeneity of ruggedness and modality on micro sale $\downarrow \lvert SE \rvert_{norm}$ lower computational effort correlates with $\uparrow FEM_{micro_{std}}$: higher heterogeneity of ruggedness and modality on micro sale
$\boldsymbol{FEM_{macro_{mean}}}$ - IDICE - *decrease* (-0.38)	Mean values for ruggedness and modality on macro scale combined with average gradients $\uparrow \lvert SE \rvert_{norm}$ higher computational effort correlates with $\downarrow FEM_{macro_{mean}}$: lower mean ruggedness, modality and average gradients $\downarrow \lvert SE \rvert_{norm}$ lower computational effort correlates with $\uparrow FEM_{macro_{mean}}$: greater mean ruggedness, modality and average gradients
$\boldsymbol{FEM_{micro_{mean}}}$ - IDICE - *decrease* (-0.45)	Mean values for ruggedness and modality on micro scale $\uparrow \lvert SE \rvert_{norm}$ higher computational effort correlates with $\downarrow FEM_{micro_{mean}}$: lower mean ruggedness and modality on micro scale $\downarrow \lvert SE \rvert_{norm}$ lower computational effort correlates with $\uparrow FEM_{micro_{mean}}$: greater mean ruggedness and modality on micro scale

Tab. C.4.: Proportionally high **negative** correlation for $|SE|_{norm}$; The numbers in parentheses are the maximum correlation coefficients for the FLA metric and the annotated variants of the exchange topology in combination with the heuristic.

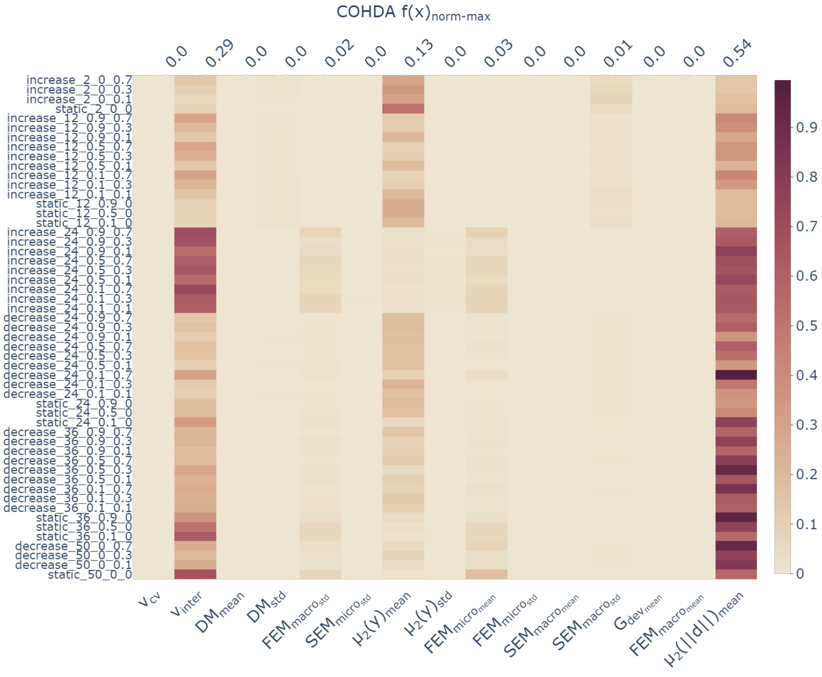


(a) Probability value for spearman coefficients with $f(x)_{norm}$ for COHDA

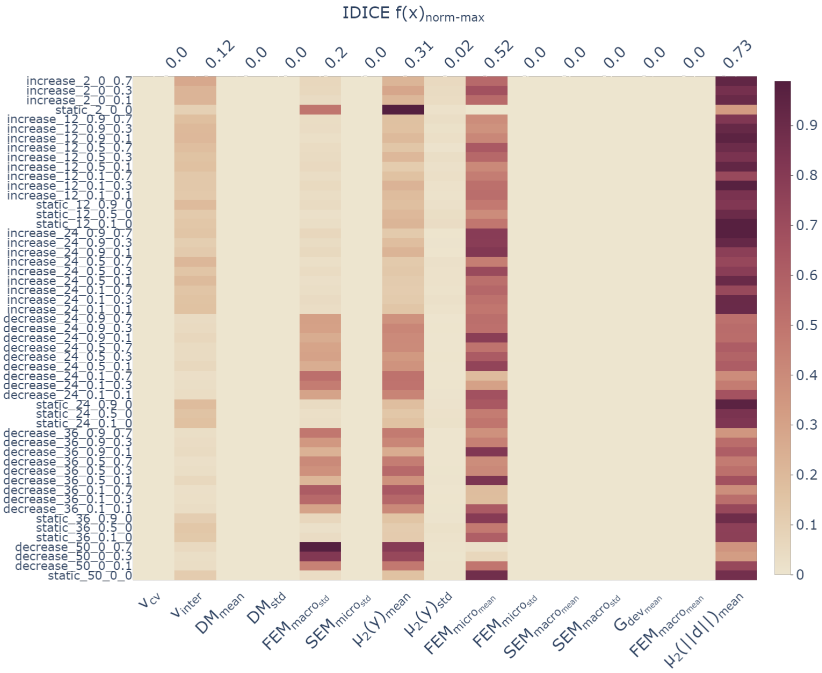


(b) Probability value for spearman coefficients with $f(x)_{norm}$ for IDICE

Fig. C.1.: Probability value for spearman correlation coefficients for the FLA metrics with $f(x)_{norm}$ computed separately for each exchange topology variant and across all problem instances.

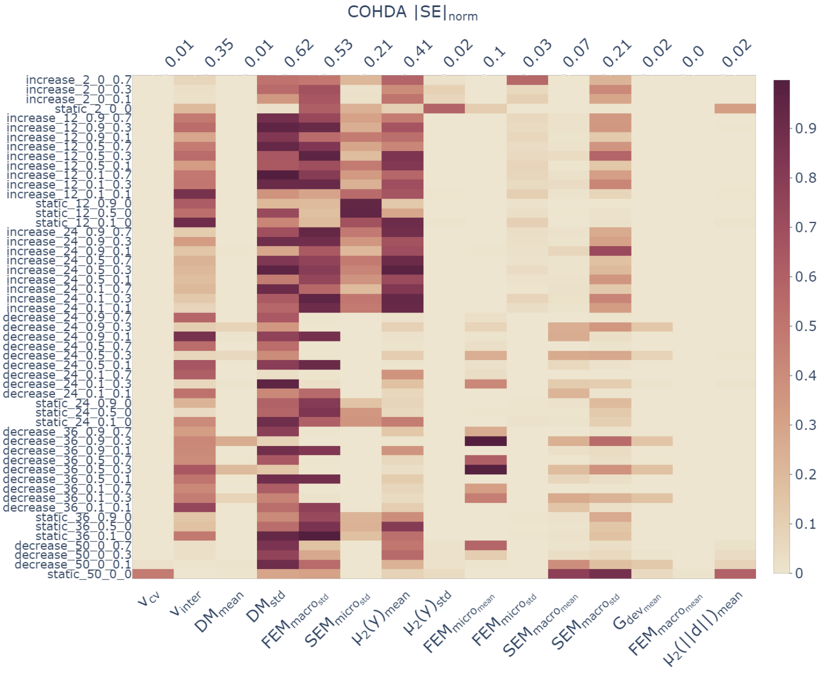


(a) Probability value for spearman coefficients with $f(x)_{norm-max}$ for COHDA

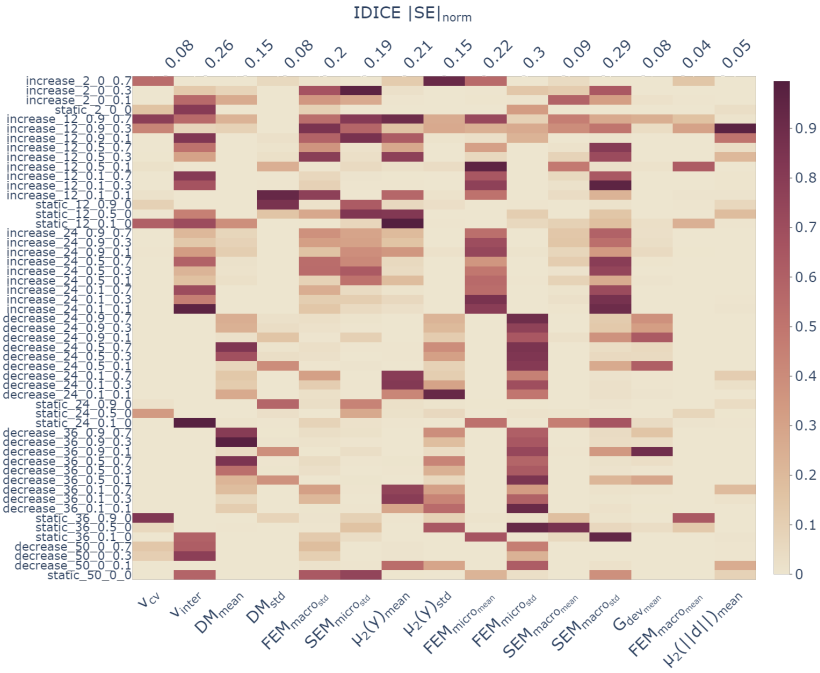


(b) Probability value for spearman coefficients with $f(x)_{norm-max}$ for IDICE

Fig. C.2.: Probability value for spearman correlation coefficients for the FLA metrics with $f(x)_{norm-max}$ computed separately for each exchange topology variant and across all problem instances.

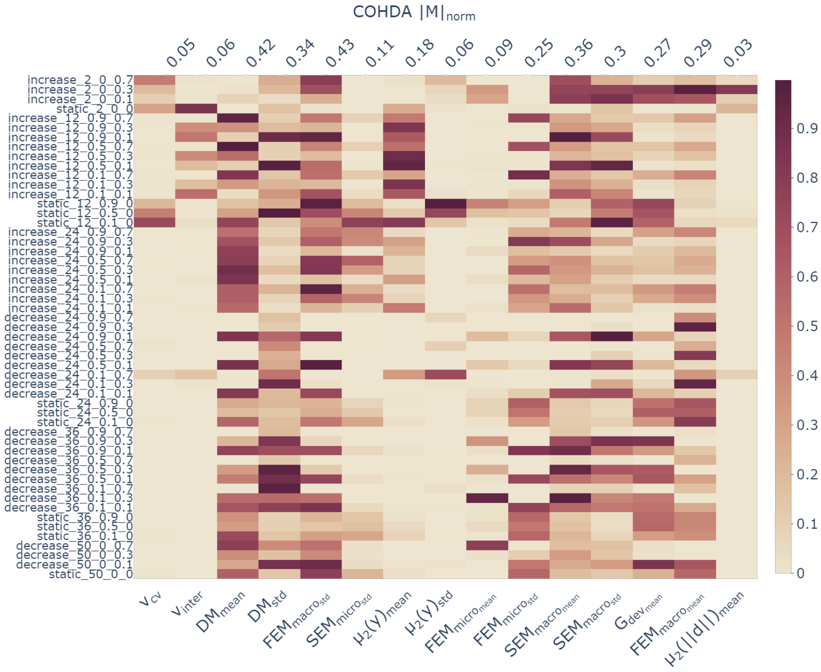


(a) Probability value for spearman coefficients with $|SE|_{norm}$ for COHDA

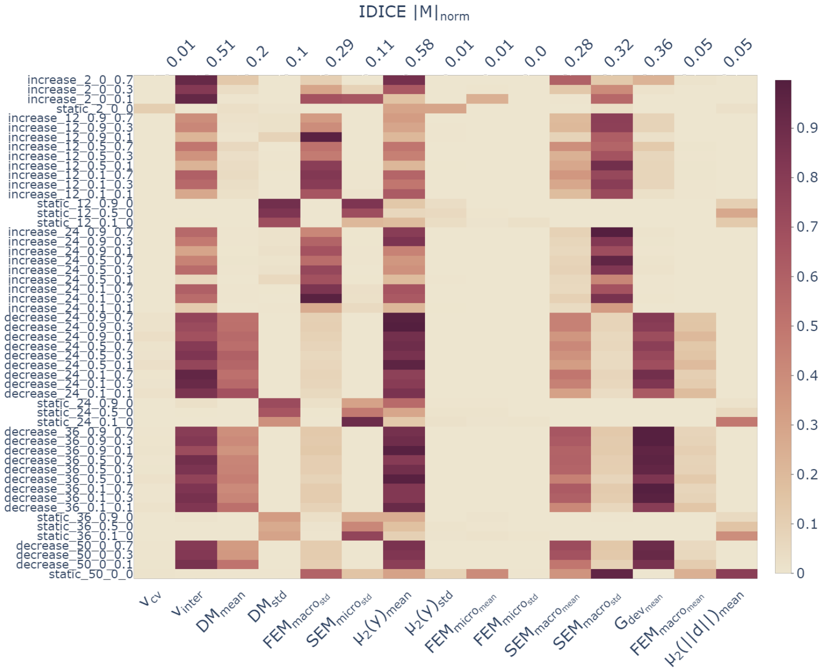


(b) Probability value for spearman coefficients with $|SE|_{norm}$ for IDICE

Fig. C.3.: Probability value for spearman correlation coefficients for the FLA metrics with $|SE|_{norm}$ computed separately for each exchange topology variant and across all problem instances.



(a) Probability value for spearman coefficients with $|M|_{norm}$ for COHDA



(b) Probability value for spearman coefficients with $|M|_{norm}$ for IDICE

Fig. C.4.: Probability value for spearman correlation coefficients for the FLA metrics with $|M|_{norm}$ computed separately for each exchange topology variant and across all problem instances.

feature		$f(x)_{norm}$			$f(x)_{norm-max}$	
	base MSE	MSE mean	MSE std (±)	base MSE	MSE mean	MSE std (±)
v_{cv}	0.0211	0.0262	0.0000	0.0005	0.0035	0.0000
v_{inter}	0.0211	0.0253	0.0001	0.0005	0.0005	0.0000
DM_{mean}	0.0211	0.1829	0.0006	0.0005	0.0016	0.0000
DM_{std}	0.0211	0.0490	0.0003	0.0005	0.0007	0.0000
$FEM_{macro_{std}}$	0.0211	0.0218	0.0000	0.0005	0.0005	0.0000
$SEM_{micro_{std}}$	0.0211	0.0231	0.0000	0.0005	0.0007	0.0000
$\mu_2(y)_{mean}$	0.0211	0.0286	0.0001	0.0005	0.0566	0.0001
$\mu_2(y)_{std}$	0.0211	0.0269	0.0001	0.0005	0.0007	0.0000
$FEM_{micro_{mean}}$	0.0211	0.0551	0.0003	0.0005	0.0006	0.0000
$FEM_{micro_{std}}$	0.0211	0.0348	0.0001	0.0005	0.0008	0.0000
$SEM_{macro_{mean}}$	0.0211	0.0276	0.0001	0.0005	0.0017	0.0000
$SEM_{macro_{std}}$	0.0211	0.1234	0.0005	0.0005	0.0005	0.0000
$G_{dev_{mean}}$	0.0211	0.0216	0.0000	0.0005	0.0005	0.0000
$FEM_{macro_{mean}}$	0.0211	0.0258	0.0001	0.0005	0.0010	0.0000
$\mu_2(\|\|d\|\|)_{mean}$	0.0211	0.0260	0.0000	0.0005	0.0007	0.0000
α	0.0211	0.0227	0.0001	0.0005	0.0006	0.0000
p	0.0211	0.0212	0.0000	0.0005	0.0005	0.0000
k	0.0211	0.0263	0.0001	0.0005	0.0013	0.0000
$mode$	0.0211	0.0262	0.0001	0.0005	0.0005	0.0000
$mode\,\&\,\alpha$	0.0211	0.0278	0.0001	0.0005	0.0007	0.0000
$\alpha\,\&\,k$	0.0211	0.0274	0.0000	0.0005	0.0012	0.0000
$initial\,topology$	0.0211	0.0269	0.0001	0.0005	0.0013	0.0000
$mode\,\&\,k$	0.0211	0.0301	0.0000	0.0005	0.0012	0.0000
$mode,\ \alpha\,\&\,k$	0.0211	0.0311	0.0001	0.0005	0.0013	0.0000
$topo\,variant$	0.0211	0.0318	0.0001	0.0005	0.0013	0.0000

Tab. C.5.: Retrieved MSE with permuted features for COHDA

feature	$\lvert SE\rvert_{norm}$			$\lvert M\rvert_{norm}$		
	base MSE	MSE mean	MSE std (±)	base MSE	MSE mean	MSE std (±)
v_{cv}	0.0093	0.0112	0.0000	0.0114	0.0163	0.0000
v_{inter}	0.0093	0.0100	0.0000	0.0114	0.0322	0.0002
DM_{mean}	0.0093	0.0106	0.0000	0.0114	0.0267	0.0001
DM_{std}	0.0093	0.0101	0.0000	0.0114	0.0252	0.0001
$FEM_{macro_{std}}$	0.0093	0.0093	0.0000	0.0114	0.0135	0.0000
$SEM_{micro_{std}}$	0.0093	0.0096	0.0000	0.0114	0.0118	0.0000
$\mu_2(y)_{mean}$	0.0093	0.0107	0.0000	0.0114	0.0119	0.0000
$\mu_2(y)_{std}$	0.0093	0.0094	0.0000	0.0114	0.0116	0.0000
$FEM_{micro_{mean}}$	0.0093	0.0112	0.0000	0.0114	0.0141	0.0000
$FEM_{micro_{std}}$	0.0093	0.0223	0.0002	0.0114	0.0142	0.0000
$SEM_{macro_{mean}}$	0.0093	0.0100	0.0000	0.0114	0.0119	0.0000
$SEM_{macro_{std}}$	0.0093	0.0117	0.0001	0.0114	0.0131	0.0000
$G_{dev_{mean}}$	0.0093	0.0119	0.0000	0.0114	0.0158	0.0000
$FEM_{macro_{mean}}$	0.0093	0.0157	0.0001	0.0114	0.0113	0.0000
$\mu_2(\lvert\lvert d\rvert\rvert)_{mean}$	0.0093	0.0092	0.0000	0.0114	0.0117	0.0000
α	0.0093	0.0195	0.0001	0.0114	0.0174	0.0000
p	0.0093	0.0095	0.0000	0.0114	0.0124	0.0000
k	0.0093	0.0236	0.0002	0.0114	0.0263	0.0001
$mode$	0.0093	0.0568	0.0004	0.0114	0.0247	0.0001
$mode\,\&\,\alpha$	0.0093	0.0662	0.0004	0.0114	0.0288	0.0001
$\alpha\,\&\,k$	0.0093	0.0314	0.0002	0.0114	0.0291	0.0001
$initial\,topology$	0.0093	0.0243	0.0002	0.0114	0.0280	0.0001
$mode\,\&\,k$	0.0093	0.0586	0.0002	0.0114	0.0310	0.0001
$mode,\ \alpha\,\&\,k$	0.0093	0.0662	0.0002	0.0114	0.0324	0.0001
$topo\,variant$	0.0093	0.0664	0.0002	0.0114	0.0339	0.0001

Tab. C.6.: Retrieved MSE with permuted features for COHDA

feature	$f(x)_{norm}$			$f(x)_{norm-max}$		
	base MSE	MSE mean	MSE std (±)	base MSE	MSE mean	MSE std (±)
v_{cv}	0.0259	0.0252	0.0000	0.0012	0.0027	0.0000
v_{inter}	0.0259	0.0674	0.0003	0.0012	0.0016	0.0000
DM_{mean}	0.0259	0.0331	0.0002	0.0012	0.0017	0.0000
DM_{std}	0.0259	0.0256	0.0000	0.0012	0.0012	0.0000
$FEM_{macro_{std}}$	0.0259	0.0252	0.0000	0.0012	0.0012	0.0000
$SEM_{micro_{std}}$	0.0259	0.0271	0.0001	0.0012	0.0013	0.0000
$\mu_2(y)_{mean}$	0.0259	0.0264	0.0000	0.0012	0.0332	0.0001
$\mu_2(y)_{std}$	0.0259	0.0253	0.0000	0.0012	0.0012	0.0000
$FEM_{micro_{mean}}$	0.0259	0.0265	0.0001	0.0012	0.0013	0.0000
$FEM_{micro_{std}}$	0.0259	0.0346	0.0001	0.0012	0.0012	0.0000
$SEM_{macro_{mean}}$	0.0259	0.0329	0.0001	0.0012	0.0014	0.0000
$SEM_{macro_{std}}$	0.0259	0.0258	0.0000	0.0012	0.0012	0.0000
$G_{dev_{mean}}$	0.0259	0.0330	0.0001	0.0012	0.0012	0.0000
$FEM_{macro_{mean}}$	0.0259	0.0257	0.0000	0.0012	0.0013	0.0000
$\mu_2(\|\|d\|\|)_{mean}$	0.0259	0.0253	0.0000	0.0012	0.0012	0.0000
α	0.0259	0.0274	0.0001	0.0012	0.0012	0.0000
p	0.0259	0.0321	0.0002	0.0012	0.0012	0.0000
k	0.0259	0.0317	0.0001	0.0012	0.0012	0.0000
$mode$	0.0259	0.0299	0.0002	0.0012	0.0012	0.0000
$mode\,\&\,\alpha$	0.0259	0.0327	0.0002	0.0012	0.0012	0.0000
$\alpha\,\&\,k$	0.0259	0.0320	0.0001	0.0012	0.0012	0.0000
$initial\,topology$	0.0259	0.0359	0.0002	0.0012	0.0013	0.0000
$mode\,\&\,k$	0.0259	0.0330	0.0002	0.0012	0.0012	0.0000
$mode,\ \alpha\,\&\,k$	0.0259	0.0348	0.0002	0.0012	0.0013	0.0000
$topo\,variant$	0.0259	0.0366	0.0002	0.0012	0.0013	0.0000

Tab. C.7.: Retrieved MSE with permuted features for IDICE

feature	$\|SE\|_{norm}$			$\|M\|_{norm}$		
	base MSE	MSE mean	MSE std (±)	base MSE	MSE mean	MSE std (±)
v_{cv}	0.0161	0.0272	0.0002	0.0005	0.0007	0.0000
v_{inter}	0.0161	0.0159	0.0000	0.0005	0.0026	0.0000
DM_{mean}	0.0161	0.0181	0.0000	0.0005	0.0008	0.0000
DM_{std}	0.0161	0.0203	0.0001	0.0005	0.0007	0.0000
$FEM_{macro_{std}}$	0.0161	0.0156	0.0000	0.0005	0.0018	0.0000
$SEM_{micro_{std}}$	0.0161	0.0174	0.0000	0.0005	0.0015	0.0000
$\mu_2(y)_{mean}$	0.0161	0.0169	0.0000	0.0005	0.0011	0.0000
$\mu_2(y)_{std}$	0.0161	0.0167	0.0000	0.0005	0.0005	0.0000
$FEM_{micro_{mean}}$	0.0161	0.0223	0.0001	0.0005	0.0037	0.0000
$FEM_{micro_{std}}$	0.0161	0.0203	0.0000	0.0005	0.0057	0.0000
$SEM_{macro_{mean}}$	0.0161	0.0180	0.0000	0.0005	0.0009	0.0000
$SEM_{macro_{std}}$	0.0161	0.0185	0.0000	0.0005	0.0012	0.0000
$G_{dev_{mean}}$	0.0161	0.0177	0.0000	0.0005	0.0030	0.0000
$FEM_{macro_{mean}}$	0.0161	0.0163	0.0000	0.0005	0.0007	0.0000
$\mu_2(\|\|d\|\|)_{mean}$	0.0161	0.0156	0.0000	0.0005	0.0017	0.0000
α	0.0161	0.0214	0.0001	0.0005	0.0035	0.0000
p	0.0161	0.0314	0.0002	0.0005	0.0017	0.0000
k	0.0161	0.0237	0.0001	0.0005	0.0700	0.0003
$mode$	0.0161	0.0432	0.0002	0.0005	0.0302	0.0001
$mode\,\&\,\alpha$	0.0161	0.0488	0.0002	0.0005	0.0366	0.0001
$\alpha\,\&\,k$	0.0161	0.0266	0.0001	0.0005	0.0695	0.0004
$initial\,topology$	0.0161	0.0381	0.0002	0.0005	0.0724	0.0004
$mode\,\&\,k$	0.0161	0.0455	0.0002	0.0005	0.0744	0.0003
$mode,\ \alpha\,\&\,k$	0.0161	0.0491	0.0002	0.0005	0.0775	0.0003
$topo\,variant$	0.0161	0.0597	0.0003	0.0005	0.0801	0.0003

Tab. C.8.: Retrieved MSE with permuted features for IDICE

	parameter	$f(x)_{norm}$	$f(x)_{norm-max}$	$\lvert SE\rvert_{norm}$	$\lvert M\rvert_{norm}$
COHDA	**number of trees**	138	117	267	138
	criterion to measure the quality of a split	poisson	friedman mse	friedman mse	poisson
	max depth of trees	20	10	20	20
	minimum number of **samples** required to **split** an internal node	5	5	2	5
	minimum number of **samples** required to be at a **leaf** node	4	1	4	4
IDICE	**number of trees**	171	42	138	138
	criterion to measure the quality of a split	friedman mse	squared error	poisson	poisson
	max depth of trees	10	10	20	20
	minimum number of **samples** required to **split** an internal node	5	10	5	5
	minimum number of **samples** required to be at a **leaf** node	1	1	4	4

Tab. C.9.: Chosen parameter setups random forest regressors

	parameter	$f(x)_{norm}$	$f(x)_{norm-max}$	$\|SE\|_{norm}$	$\|M\|_{norm}$
COHDA	**number of trees**	171	138	235	235
	criterion to measure the quality of a split	squared error	friedman mse	squared error	friedman mse
	max depth of trees	20	10	70	None
	minimum number of **samples** required to **split** an internal node	5	10	10	10
	minimum number of **samples** required to be at a **leaf** node	2	4	1	4
IDICE	**number of trees**	235	138	235	235
	criterion to measure the quality of a split	friedman mse	friedman mse	friedman mse	friedman mse
	max depth of trees	None	10	None	None
	minimum number of **samples** required to **split** an internal node	10	10	10	10
	minimum number of **samples** required to be at a **leaf** node	4	4	4	4

Tab. C.10.: Chosen parameter setups random forest regressors for predicting noisy data