TU WIEN Informatics

A Multi Objective Discrete Particle Swarm Optimization Algorithm for Mobile Offloading

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Software Engineering & Internet Computing

eingereicht von

Lukas Leitzinger, BSc.
Matrikelnummer 01327140

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ. Prof. Dr. Ivona Brandic
Mitwirkung: Dr. Vincenzo De Maio

Wien, 22. November 2024

Lukas Leitzinger	Ivona Brandic

TU WIEN Informatics

A Multi Objective Discrete Particle Swarm Optimization Algorithm for Mobile Offloading

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Software Engineering & Internet Computing

by

Lukas Leitzinger, BSc.
Registration Number 01327140

to the Faculty of Informatics

at the TU Wien

Advisor: Univ. Prof. Dr. Ivona Brandic
Assistance: Dr. Vincenzo De Maio

Vienna, November 22, 2024

Lukas Leitzinger | Ivona Brandic

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Lukas Leitzinger, BSc.

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.
Ich erkläre weiters, dass ich mich generativer KI-Tools lediglich als Hilfsmittel bedient habe und in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt. Im Anhang „Übersicht verwendeter Hilfsmittel“ habe ich alle generativen KI-Tools gelistet, die verwendet wurden, und angegeben, wo und wie sie verwendet wurden. Für Textpassagen, die ohne substantielle Änderungen übernommen wurden, habe ich jeweils die von mir formulierten Eingaben (Prompts) und die verwendete IT-Anwendung mit ihrem Produktnamen und Versionsnummer/Datum angegeben.

Wien, 22. November 2024

Lukas Leitzinger

Danksagung

Zunächst möchte ich meinen Betreuern Univ. Prof. Dr. Ivona Brandic und Dr. Vincenzo De Maio meinen tiefsten Dank aussprechen. Ihre Geduld und Ihr konstruktives Feedback über die vielen Jahre, die diese Arbeit in Anspruch genommen hat, waren von unschätzbarem Wert und haben diese Arbeit ermöglicht.

Ein besonderer Dank gilt meiner Frau Julia. Du hast mich stehts durch deine Liebe und dein Glaube an mich motiviert, vor allem in so manchen langen Nächten. Ohne dich wäre dieser Erfolg nicht möglich gewesen.

Meinen Eltern, Maria und Helmuth, danke ich von Herzen für ihre bedingungslose Unterstützung. Ihr habt mich stets in meinem Studium bestärkt, an mein Potenzial geglaubt und mir alle Möglichkeiten eröffnet, meine Ziele zu erreichen.

Abschließend möchte ich meinen Freunden, Studien- und Arbeitskollegen, sowie allen, die mich auf diesem Weg unterstützt haben, danken.

Danke, dass Ihr diese Reise möglich gemacht habt.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my advisors Univ. Prof. Dr. Ivona Brandic and Dr. Vincenzo De Maio for their patience, guidance and constructive feedback throughout the many years it took to complete this work.

To my wife, Julia, thank you for your love, encouragement and belief in my abilities which kept me being motivated. Thank you for standing by me through the long hours and the countless challenges.

To my parents, Maria and Helmuth, who supported my studies, believed in my potential and provided me with every opportunity to succeed.

Finally, I would like to thank my friends, study and working colleagues, and everyone who helped me during this journey.

Thank you all for assisting me in achieving my goals.

Kurzfassung

Die steigenden Anforderungen von mobilen Applikationen in Bezug auf Rechenleistung sowie die hohen Erwartungshaltungen der Nutzer an Energieeffizienz erfordern innovative Strategien wie Mobile Edge Cloud Offloading (MECO). Diese Arbeit stellt einen Multi-Objective Discrete Particle Swarm Optimization (MDPSO)-Algorithmus vor, welcher für die Optimierung von möglichen Deployments in MECO-Umgebungen entwickelt wurde. Ziel des Algorithmus ist die Berechnung von effizienten und ausgeglichenen Offloading-Strategien der einzelnen Applikationsaufgaben auf die gegebenen Rechenknoten, mit Berücksichtigung der in Konflikt stehenden Optimierungsziele von Applikationslaufzeit, Batterielaufzeit und Nutzerkosten.

Der MDPSO-Algorithmus erweitert klassische Particle-Swarm-Optimization-Ansätze (PSO) mit dem diskreten Aufgaben-Mapping der MECO-Deployments. Dazu wurden ein Mechanismus zur Darstellung und Geschwindigkeitsberechnung der Partikel, eine Archivierungsstrategie für die besten Lösungen sowie eine Auswahlregel dieser globalen Optima mit Bezug auf die Optimierungsziele integriert. Zur Validierung des Algorithmus wurde ein Edge-Simulation-Framework verwendet, mit dem realistische MECO-Szenarien, bestehend aus verschiedenen Infrastrukturkonstellationen sowie typischen mobilen Applikationen, dargestellt als gerichtete azyklische Graphen (DAGs), modelliert wurden.

Der verwendete Algorithmus wurde anhand standardisierter Qualitätsindikatoren wie Hypervolume und Generational Distance evaluiert und mit den etablierten Multi-Objective-Optimization-Algorithmen (MOO) NSGA-II, NSGA-III, MOCell und SPEA2 verglichen. Die Ergebnisse zeigen, dass der MDPSO-Algorithmus besonders in MECO-Szenarien mit mittlerer bis hoher Komplexität durch eine hohe Konvergenz und Diversität der Lösungen überzeugt. Gleichzeitig wurde das Potenzial zur Verbesserung der Rechenleistung und Effizienz im Vergleich zu bestehenden Algorithmen bei weniger komplexen Szenarien identifiziert.

Abstract

The increasing computational demands of modern mobile applications, along with user expectations for energy efficiency and low latency, require advanced strategies, like mobile edge cloud offloading (MECO). This thesis proposes a Multi-Objective Discrete Particle Swarm Optimization (MDPSO) algorithm designed to optimize the deployment of computational tasks in MECO environments. By addressing the conflicting objectives of application runtime, battery life and user costs, the proposed algorithm aims to deliver efficient and balanced task offloading strategies.

The MDPSO algorithm is designed to combine traditional particle swarm optimization (PSO) techniques with discrete task mapping by incorporating a particle encoding mechanism for the MECO deployment problem, a leader archive strategy as well as an objective-based global best selection rule. To validate the applied algorithm, an edge simulation framework was used to model the edge and cloud computing environments of real-world MECO scenarios with directed acyclic graphs (DAGs) to represent mobile applications.

The proposed algorithm is evaluated and compared against state-of-the-art multi-objective optimization algorithms such as NSGA-II, NSGA-III, MOCell and SPEA2, by using standardized performance metrics like hypervolume and generational distance. The results demonstrate the MDPSO's ability to find well-distributed Pareto-optimal solutions. In medium- to high-complexity MECO scenarios, the proposed MDPSO outperforms the reference algorithms in terms of convergence and diversity. However, it also shows possible improvements in computational efficiency and performance in simpler problem instances compared with established algorithms.

Contents

CHAPTER 1

Introduction

Over the past few years, smartphones have become essential in almost every part of our daily lives - changing how we communicate, work, and entertain ourselves. As smartphones evolve to support more demanding applications like augmented reality, natural language processing tasks and real-time gaming, also the mobile users expectations on those heavy computation applications increases drastically over the last years [1].

At the same time, consumers focus even more on the battery life of mobile devices. In a recent online survey over 80% of the participants answered they prefer battery life over better computing performance when buying a new smartphone [81]. This growing focus on battery performance outlines a challenge: while the hardware of devices continuously improves, the battery technology cannot keep up with the rapid growth in computational requirements. Addressing this gap requires additional, innovative solutions that go beyond hardware improvements, aiming to optimize the energy consumption of applications without compromising user experience.

One promising approach to mitigate the energy challenges of mobile applications is to offload the computation tasks to a remote infrastructure. This approach, originally proposed in the 1990s, allows mobile devices to transfer resource-intensive tasks to external servers, thereby conserving local energy and enabling higher processing capabilities [78]. Since then, the research area of mobile offloading and mobile cloud computing (MCC) is gaining more and more popularity and has been introduced as a potential technology for mobile devices to reduce energy consumption by offloading computation-intensive parts of an application to a remote cloud computing infrastructure [31].

Dependency on the cloud brings new challenges, especially latency, which can have a negative impact on the user experience in real-time applications such as AR and gaming. To address latency, Mobile Edge Cloud Offloading (MECO) combines cloud and edge computing, bringing computation closer to the user and creating a heterogeneous

infrastructure that enhances performance and energy efficiency for latency-sensitive applications [26].

1.1 Problem statement

In order to address all these aspects, the main challenge in MECO is to determine which tasks of the mobile application are needed to offload in general, where to offload them and how to balance the main factors of runtime, energy consumption and user costs. Given the variety of tasks and computing resources, this requires an intelligent deployment strategy.

In this scenario of MECO, a mobile application can be seen as a set of independent tasks with different requirements, where each task can be executed locally on the mobile device or can be offloaded to either cloud or edge nodes. The mapping of an application's tasks to a computational node for execution is known as a deployment. Depending on the applications' requirements and the available infrastructure, there can be multiple possible deployments for the mobile application. These solutions can be evaluated by (1) applications' runtime, (2) mobile device battery lifetime and (3) costs for the user [26].

These objectives usually influence each other, for example, offloading tasks can lead to a better battery lifetime but, at the same time, increase the user costs. Due to the conflicting nature of these objectives, there is not one single ideal solution but a set of trade-off solutions. Finding optimal deployments for mobile offloading, including runtime, battery lifetime and user costs, constitutes an optimization problem with multiple objectives.

1.2 Contribution and Methodological Approach

There are several methods for solving an optimization problem with multiple objectives. Evolutionary algorithms provide a widespread and robust approach. A commonly known subcategory is genetic algorithm (GA), which is based on information transmission using crossover, mutation and selection operators. However, GA has high computational complexity when applying these specified operators and selecting target points [75]. Hence, Particle Swarm Optimization (PSO) as a population-based algorithm was chosen for this work. PSO relies on Swarm Intelligence (SI) and is a very efficient alternative to GA, where only the most optimistic particles share information with others [57].

The primary contribution of this thesis lies in the design, implementation and evaluation of an enhanced multi-object particle swarm optimization algorithm tailored to the MECO deployment problem. In PSO, particle variables are typically based on real numbers, allowing straightforward calculations of the particle position and velocity. Since a solution to the deployment problem is a concrete mapping of tasks to infrastructure nodes, the particles need to be discrete variables. Therefore, a key challenge in the algorithm design is to adapt the core principles of PSO, like particle encoding, continuous position and

velocity updates as well as the social influence within a swarm, into discrete task-to-node mappings suitable for the three-dimensional object space of MECO.

To ensure that the proposed Multi-Objective Discrete Particle Swarm Optimization (MDPSO) algorithm is rigorously tested, validated across various offloading scenarios and evaluated against established MOO algorithms, the approach of this thesis combines the following methodologies:

Multi-Objective Optimization: In multi-objective optimization, the aim is not to optimize a single objective, but rather a vector of objectives. This vector often includes conflicting goals that cannot be fully optimized simultaneously [29]. For MECO, the deployment optimization problem involves three primary objectives: maximizing battery life and minimizing runtime as well as user costs. These three objectives interfere with each other, as reducing runtime can increase battery consumption, and minimizing costs may lead to higher latency or energy usage.

Hence, the solution is not a single optimal configuration but a set of trade-off solutions known as a Pareto set [61] which consists of non-dominated solutions with none being superior among across the others. The Pareto set's quality is evaluated and can be compared using several indicators, including Hypervolume [97], General Distance [87] and ϵ-indicator [99] which provide insights into different aspects of the resulting Pareto front approximation, such as convergence or diversity.

Directed Acyclic Graphs Scheduling: Applications in computational offloading often take the form of Directed Acyclic Graphs (DAGs), with each node representing a task and each edge representing task dependencies. In computational offloading, applications are often represented as Directed Acyclic Graphs (DAGs), where each node is a task and each edge represents task dependencies. DAGs provide a structured way to model complex workflows, making them suitable for scientific and mobile application scheduling [95]. In MECO as well as more general mobile cloud computing scenarios, this DAG structure allows tasks to be scheduled on local, edge, or cloud resources depending on dependencies, resource availability, and optimization goals [2].

In this thesis, the mobile applications used for evaluation — Navigator, Face Recognizer, Antivirus, and Chess — are represented as DAGs. De Maio et al. [26] described and used this modeling approach to ensure that simulations reflect real-world mobile applications with different dependency structures and computational needs. Given a DAG and available resources, the algorithm schedules tasks in a way that respects dependencies and optimizes the placement of each task to achieve the best possible trade-off among the objectives [26].

Mobile Edge Cloud Offloading Simulation: Testing offloading algorithms within MECO requires a realistic modeling of cloud and edge infrastructures. Edge computing introduces additional complexity, as it combines different resources with varying performance levels and network conditions [5]. To emulate these

conditions, the simulations are conducted by using SLEIPNIR[1], an extended framework that models mobile cloud infrastructures, including cloud and edge resources and communication links between mobile devices and nodes [13].

This simulation framework supports configurations that consider both 3G and Wi-Fi network links, modeling variations in bandwidth and latency to reflect real-world network inconsistencies. Additionally, environmental factors such as connection unreliability are simulated through randomized Quality of Service (QoS) settings, which include bandwidth, latency, and reliability measures [13]. For mobile devices, the energy consumption model defined in De Maio et al. [26] is applied in order to enable realistic calculations of power usage for both computation and offloading. This environmental setup ensures that the MDPSO algorithm is evaluated under conditions that mirror real-world MECO deployments.

Monte Carlo Simulations and Statistical Analysis: Monte Carlo simulations are employed to evaluate the MDPSO algorithm across a diverse array of offloading scenarios, testing its adaptability and robustness under varying conditions. By performing a large number of independent simulations using different randomly generated samples of possible offloading configurations - representing different applications, infrastructure conditions and network variabilities - the proposed algorithm is compared against standard algorithms like NSGAII [29], NSGAIII [54], SPEA2 [61] and MOCell [94].

The resulting solution sets are then compared with each other by using various mathematical quality indicators in order to provide comparable insights into the algorithm's performance. To provide a solid basis for the interpretation of these indicator values, different statistical tests (including the Wilcoxon test and Friedman test) are applied in order to check the statistical significance with respect to the number of independent executions.

1.3 Structure of the thesis

This thesis is organized into the following chapters. Chapter 2 introduces the fundamental concepts such as cloud and edge computing, mobile offloading, multi-objective optimization and particle swarm optimization. It provides the theoretical foundation which is necessary to understand the used challenges and methods.

In Chapter 3, the state-of-the-art in multi-objective optimization is reviewed. It provides an overview of prominent algorithms, current MOO development frameworks used in literature and edge simulation tools, establishing the context for the research contributions of this thesis.

Chapter 4 introduces the primary contribution of the thesis: the Multi-Objective Discrete Particle Swarm Optimization (MDPSO) algorithm tailored for MECO scenarios. It

[1] https://github.com/vindem/sleipnir

defines the problem and outlines key components such as the leader archive, global best selection and particle encoding, followed by the presentation of the final algorithm.

Chapter 5 focuses on the performance evaluation of the proposed algorithm using simulations of various problem instances and comparing them against common MOO reference algorithms. After describing the evaluation approach, comparing quality indicators and statistical analysis, the results are compared and discussed.

The last Chapter 6 summarizes the research findings and outlines potential directions for future work.

CHAPTER 2

Fundamentals

This chapter outlines the fundamental concepts that are relevant to understand the optimization challenges of Mobile Edge Cloud Offloading (MECO). First, an overview of **Cloud Computing** as well as **Edge Computing** is given. **Mobile Offloading** is then described as a technique to improve battery life and performance on mobile devices. Afterwards, **Multi-Objective Optimization** is explained as a methodology to optimize MECO deployments, describing different aspects such as Pareto optimality, solution constraints and performance metrics for evaluations. Finally, the core principles of **Particle Swarm Optimization** are presented as an effective algorithm class for solving these optimization problems with multiple objectives.

2.1 Cloud Computing

Cloud computing is a technology which enables the delivery of computing services such as servers, storage, databases, networking and software over the internet, which is often referred to as the *cloud*. It allows users to access these resources on-demand, eliminating the need for managing physical hardware and infrastructure themselves.

This technology has become as essential to our daily lives as smartphones by providing the powerful computing resources which are no longer fully available on the users' local devices such as desktops, tablets or gaming consoles. The *cloud* in this context represents an abstract infrastructure with several core properties: scalability, reliability, fault tolerance and high availability. These features enable users to dynamically utilize computing resources and storage based on their needs [37].

2.1.1 Core Characteristics of Cloud Computing

The cloud's infrastructure offers a flexible and on-demand resource pool for tasks such as data processing, storage and computing, which ensure remote access to the users'

resources. Hence, the core characteristics include the following features.

Scalability and Flexibility: Users can scale their resource usage up or down depending on their demand. This makes it ideal for businesses that experience variable workloads [37].

Reliability and Availability: Cloud providers often ensure high levels of uptime through redundancy and geographic distribution, which boosts reliability.

Abstraction and Virtualization: Through virtualization, cloud providers abstract the hardware layer, allowing multiple users to share resources in a way that appears isolated to each user. This model improves efficiency and resource usage [18].

2.1.2 Cloud Service Models

Cloud computing is broadly categorized into three main service models, ranked by increasing levels of abstraction:

Infrastructure as a Service (IaaS): As the name suggests, in this service model, technical infrastructure or virtualized hardware is provided as a service. Here, users lease essential computing resources, such as servers, storage and networks, without managing the physical infrastructure by themselves. Users gain the flexibility to install and manage their own operating systems and applications which makes it suitable for those who require a high level of control over their environment [62]. IaaS users enjoy flexibility, however, they are responsible for managing everything from the operating system level up, including patches and upgrades.

Platform as a Service (PaaS): PaaS provides a higher level of abstraction than IaaS by delivering a ready-to-use development platform. Users can build and deploy applications without managing the underlying infrastructure. This service typically includes tools for development, testing and deployment [18]. PaaS examples such as Google App Engine, Heroku and Microsoft Azure App Services simplify the development process as they provide a managed environment for creating and running applications. Users are responsible for their application code and configurations, while the provider manages the servers, storage and networking.

Software as a Service (SaaS): SaaS represents the highest level of abstraction, where users access fully developed applications over the internet, which is accessible directly from web browsers or apps without local installation [37]. Familiar SaaS examples include Gmail, Google Workspace, Salesforce, and Microsoft Office 365. They reduce the need for maintenance and infrastructure, by offering quick access to applications on any device with internet connectivity.

2.1.3 Cloud Deployment Models

Cloud services are deployed through various models to suit different privacy and management needs:

Public Cloud: These cloud services are accessible to anyone and typically hosted by third-party providers such as AWS, Google Cloud, and Microsoft Azure. Public clouds are known for their scalability and cost-efficiency but may share resources across multiple users, leading to potential concerns regarding data privacy [18].

Private Cloud: In scenarios requiring strict data control, a private cloud provides dedicated resources exclusively for a single organization. Private clouds are often hosted on-premise and ensure data remains within organizational boundaries to meet regulatory or security requirements. Many businesses, especially in the financial and governmental sectors, rely heavily on private clouds in order comply with stringent data protection policies [18].

Hybrid Cloud: Combining public and private cloud elements, hybrid clouds allow data and applications to move between environments. This flexibility is useful for businesses in need of secure, private storage for sensitive data and scalable resources for less sensitive operations. Hybrid models enable optimized infrastructure usage in order to balance costs and security [37].

2.2 Edge Computing

As desribed in the previous section, cloud computing provides flexible access to resources and computing power in centralized data centers over the internet. However, even via a fast and reliable internet connection, due to the physical distance to this remote servers, for some real-world scenarios the latency can be a limiting factor, which can have a negative impact on the performance [80].

Edge computing has emerged as a decentralized model, which addresses this gap by bringing computation and data storage closer to the sources, reducing latency and therefore the need to transfer data to remote centralized servers.

2.2.1 Core Characteristics of Edge Computing

In contrast to cloud computing, edge computing is characterized by following features:

Proximity to Data Sources: By processing data locally or close to the device, instead of transferring data over long distances, delays and response time is minimized [36]. This makes it ideal for applications and scenarios where quick response times are required, as in autonomous driving, video streaming, cloud gaming or augmented reality [45].

Bandwidth Efficiency: The amount of data transmitted over networks is reduced, which saves bandwidth as well as costs by processing data locally and eventually only transferring filtered or aggregated data. This can be immensely helpful in scenarios which involves high data volumes, like video surveillance or smart city sensors [45].

Location Awareness: Edge systems are able to collect and process data based on geographic location without needing to transfer them into the cloud for transportation applications or utility management [80].

Scalability for IoT Devices: Many different deployed edge nodes can manage the processing load of locally connected devices, eventually avoiding an overload for a central server, which allows scalable solutions for growing IoT ecosystems [45].

2.2.2 Architecture of Edge Computing

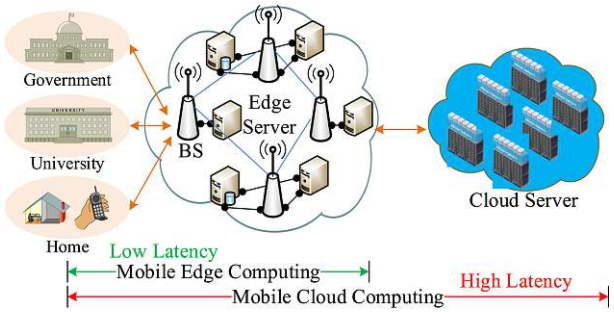


Figure 2.1: The overview of the MEC and MCC architecture [36]

Building on the capabilities of cloud computing, the edge computing architecture consists of following components, as can be seen in Figure 2.1:

- **Edge Devices** like sensors, cameras and mobile phones are responsible for generating data and sometimes offload tasks [36].
- **Edge Nodes** are local processing units (often at base stations, micro data centers or other network infrastructure) processing tasks as well as handling aggregation, analysis and storage of data, reducing transmission load and speeding up response times [80].
- **Central Cloud Servers** can be used for offloading of tasks which requires high computation power or storage by edge devices or even edge nodes. This hybrid setup combines the benefits of edge processing and the vast resources of cloud computing [45].

2.2.3 Challenges and Considerations in Edge Computing

However, beside its benefits, the operating of a distributed edge infrastructure entails the following considerations:

Infrastructure Costs: In contrast to the cost efficient nature of centralized cloud data centers, both the initial deployment and ongoing maintenance of edge nodes can be costly [45]. Furthermore, unlike the centralized control in cloud data centers, the efficient allocation of computational resources across edge nodes is more difficult to manage, which results in possible underutilized hardware or in bottlenecks which can lead to violations of service level agreements [80].

Data Management and Consistency: In addition to the resource utilization, providing needed data synchronization as well as data consistency across decentralized nodes can be challenging. Especially when it comes to real-world scenarios where the devices are mobile and switch between different networks [36]

Security and Privacy: Even though security can be enhanced due to reducing data transfers, decentralized data processing can also involve vulnerabilities. Therefore, enhanced security measures at all edge nodes is essential to protect user privacy, which can be complex due to its decentralized infrastructure [80].

2.3 Mobile Offloading

As mentioned in the introduction, mobile offloading is a promising approach to overcome hardware limitations for mobile devices. Its basic idea is to pass resource-intensive tasks to servers and let them handle the computation and data processing to relieve the mobile device from the load [18]. The concept originates from mobile cloud computing (MCC), solely considered to offload single tasks to centralized data centers with high computation power, but has evolved to prioritize latency and bandwidth, especially required in real-time applications like gaming or augmented reality [63]. As described in Section 2.2, edge computing addresses these challenges by allowing mobile devices to offload tasks to nearby edge nodes instead of distant data centers, which lowers the latency and input lag on the one hand and results in enhanced user experience on the other [66].

2.3.1 Mobile Edge Cloud Offloading

Mobile Edge Cloud Offloading (MECO) offers a hybrid approach by utilizing centralized cloud servers as well as nearby edge nodes when it comes to offloading of tasks. This provides the flexibility to respond to the tasks' requirements, such as use edge nodes where minimal latency and offload to cloud nodes, when computationally intensive operations needs to be done. Keeping this flexibility in mind, MECO introduces the following key considerations for optimizing its effectiveness when it comes todeciding on where and what to offload and if offloading at all [26]:

- **Runtime** of the task is a primary factor, as the decision of whether to offload it — and if yes, where to offload it — directly impacts the application's responsiveness and overall user experience.
- **Battery life** of the mobile device can be extended, as offloading reduces the computational load on mobile devices. However, also the offloading itself with its communication and data transfer to the nodes consumes considerable energy.
- **User Costs** incurred by renting computation power or other service fees needs to be carefully managed to ensure that the offloading process remains economically viable for end users. Additionally, utilizing edge nodes is usually more expensive than computation power of cloud servers, due to its decentralized nature and complex maintenance [45].

2.3.2 Types of Offloading

Besides the categorization in MECO based on the offloading destination, such as utilizing cloud or edge nodes, mobile offloading in general follows three main approaches based on the execution model, visualized in Figure 2.2:

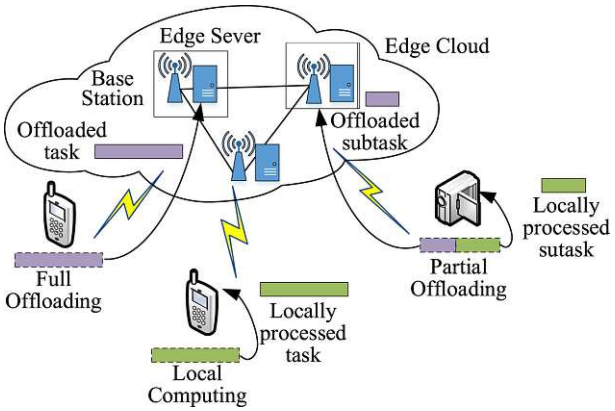


Figure 2.2: Offloading types in mobile edge computing [36]

- **Full Offloading** describes the execution of the entire workload on a remote node, limiting the needed communication and data transfer between the mobile device and node to a minimum. This approach is suitable for high-resource tasks, where the device lacks the overall capacity to execute any portion locally [36]. One example is live video processing, a scenario highlighted by Naouri et al. [66], where the computational load exceeds the processing capacity of local devices.
- **Partial Offloading** allows splitting of workloads between the mobile device and edge or cloud nodes. This approach provides flexibility to jointly execute locally on the device and remote processing on edge or cloud nodes, which can optimize energy costs and runtime by offloading only the most resource-intensive tasks [66].

At the same time it adds more complexity due to task dependencies on each other [36].

- **Local Computation**, where all tasks are executed directly on the mobile device, does not consider offloading, but it may be a necessary fallback due to factors such as poor network quality or sufficient computational capacity on the device [36].

2.3.3 Key Components of Mobile Offloading

To ensure an efficient resource utilization and task delegation in real-world mobile offloading, a deployment strategy needs to be computed while considering the application's structure as well as the infrastructure capabilities. De Maio et al. proposed a model of an offloading engine including following key components, visualized in Figure 2.3:

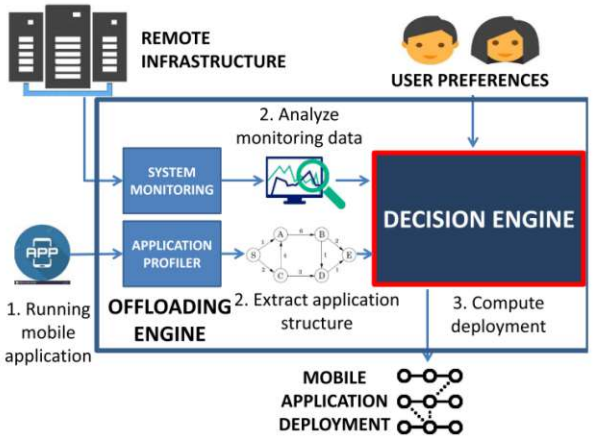


Figure 2.3: Offloading model [26]

Application Profiler analyzes the structure of a mobile application in smaller tasks as well as identifying their dependencies between each other, such as the need what needs to be executed first [26]. In addition, tasks are classified in their requirements, like resource needs or whether it is unoffloadable due to the use of local device components, e.g. camera or GPS [66].

System Monitor gathers real-time information about the remote MEC infrastructure, including available cloud and edge nodes as well as their resources [26]. Also insights on network conditions are relevant. For example, given a bad connection with low bandwidth or high latency, the processing of tasks could slow down or even fail. Ongoing monitoring of these parameters can be used to dynamically adapt to changes in network conditions or resource capacities [66].

Decision Engine is the core, where all collected data from the system monitor, application profile and potential user preferences are analyzed and a deployment strategy

is computed. This can be done by optimization algorithms which calculate valid mapping of tasks to computational nodes and trying to optimize multiple objectives of runtime, battery life and user costs, as mentioned above [26].

2.4 Multi-Objective Optimization

Multi-objective optimization (MOO) focuses on solving problems that involve multiple, often conflicting objectives, which means that no single solution can simultaneously optimize all objectives. Instead, MOO aims to find a set of trade-off solutions that balance these objectives effectively. These trade-off solutions form what is known as the Pareto-optimal set or Pareto front, where each solution represents a balance where no objective can improve without worsening at least one other objective [29]. This fundamental concept makes MOO critical in fields like engineering, finance, and logistics, where various goals must be met simultaneously [42].

In a typical multi-objective optimization problem, we are given an n-dimensional vector of decision variables, $\vec{x} = [x_1, x_2, \ldots, x_n]$, and m objective functions, $f_1(\vec{x}), f_2(\vec{x}), \ldots, f_m(\vec{x})$, each mapping $\vec{x}$ to a scalar value, $f_i(\vec{x}) : \mathbb{R}^n \to \mathbb{R}$, which needs to be minimized or maximized [22]. Mathematically, it can be expressed as:

$$\text{minimize/maximize}\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), \ldots, f_m(\vec{x})], \tag{2.1}$$

where $\vec{f}(\vec{x})$ represents the vector of m objective functions. For example, if $f_1(\vec{x})$ needs to be minimized and $f_2(\vec{x})$ maximized, it can be expressed as $\text{minimize}\vec{f}(\vec{x}) = [f_1(\vec{x}), -f_2(\vec{x}))]$. The solution space consists of feasible solutions that satisfy constraints, if any, within which the optimization process seeks the Pareto-optimal set [22].

2.4.1 Pareto Optimality and the Pareto Front

The concept of Pareto optimality is central in MOO, where a solution is deemed "Pareto optimal" if there is no other solution that can improve one objective without worsening at least one other objective. This results in a set of non-dominated solutions, collectively known as the Pareto front. For instance, in scheduling problems, optimizing for both cost and time, lead to various solutions where some are faster but more expensive, while others are slower but cheaper, representing points on the Pareto front [32].

The visual in Figure 2.4 helps to illustrate the structure of a MOO problem. The decision space (on the left) consists of the variables x_1 and x_2, representing possible configurations or solutions for a problem. The objective space (on the right) is mapped from the decision space through objective functions $f_1(x)$ and $f_2(x)$, representing the outcomes or performance measures which are aimed to be optimized. In the decision space, infeasible regions are shaded, indicating solutions that do not satisfy constraints. Solutions in the objective space can be classified as either dominated (e.g., points C, D, F) or non-dominated (points A, B). Dominated solutions (in white) are inferior to at least one solution, meaning they could be improved in at least one objective without compromising

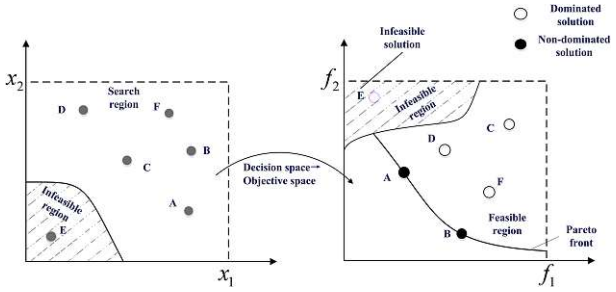


Figure 2.4: Relationship between the design space and the objective space and solution definition of a two-objective problem [24]

the other. Non-dominated solutions (in black) lie on the Pareto front, indicating optimal trade-offs. A solution is considered Pareto optimal if it is not dominated by any other solution, meaning that improving one objective would degrade another. Points on this front represent optimal trade-offs for the given objectives. This front is critical for decision-making, as it provides a range of balanced solutions without one dominating another [24].

2.4.2 Solution Constraints

In multi-objective optimization (MOO), constraints play a critical role in defining feasible solutions by limiting the decision space and shaping the structure of the objective space, as represented in Figure 2.4.

Constraints are essential in ensuring that the resulting solutions can also be used in real-world applications, where physical, technical and operational limitations are typical. For example, in sustainable energy systems, constraints related to energy efficiency, emissions, and operational limits ensure that optimized solutions contribute positively to environmental goals [24]. Similarly and more relevant to this thesis, the whole group of discrete optimization problems needs constraints to check the applicability, such as QoS constraints in cloud computing task scheduling [85] or capacity constraints in mobile offloading, validate if a deployment of an application on an infrastructure satisfies capacity boundaries, such as available RAM, storage and cores on the computational nodes [26].

Following approaches are used in multi-objective optimization to handle constraints:

Penalty Methods: Penalty-based methods add penalty terms to the objective function, trying to to also consider the solution's degree of constraint violation. Therefore, solutions which are outside of the feasible region will be punished and hence are less attractive. In other words, searching in other areas needs to be conducted again. The impact of penalties does not need to be static, but also can be dynamically adjusted over time based on solution feasibility [28].

Constraint-Dominance Mechanisms: In this approach, constraints are directly included into the Pareto dominance computation. It always prefers or prioritizes

feasible solutions over infeasible ones. However, when comparing infeasible solutions with each over, the degree of constraint violation needs to be considered [22]. This means in this case, the infeasible solution which has a lower level of constraint violation is dominating the other one [28].

ϵ-Constraint Method: A complete different approach in MOO is the ϵ-constraint method. It does not simply add terms to the objective function but reformulates all objectives except one – suitable when one objective is preferred by the decision maker [64]. A MOO problem with k objectives $f_1, f_2, \ldots, f_k$ is converted by selecting one objective as main focus (f_1), while the others $f_2, \ldots, f_k$ are converted into constraints:

$$\min f_1(x) \quad \text{subject to} \quad f_j(x) \leq \epsilon_j, \quad j = 2, \ldots, k, \tag{2.2}$$

where ϵ_j represents the threshold for each secondary objective, which is iteratively varied [64].

Hybrid Approaches: As the name indicates, hybrid approaches combine multiple constraint-handling methods within the algorithm. For example, popular algorithms like NSGA-II [29] make use of both penalty functions and constraint-domination mechanism. This ensures that feasible solutions rank higher, but infeasible solutions with a low degree of violations are also considered, resulting in a better exploration of the Pareto front [29].

2.4.3 Classification of MOO Algorithms and Methods

According to Cui et al. [24], the diverse landscape of MOO algorithms can be categorized by the used multi-objective trade-off methods as well as by its source of inspiration.

These trade-off optimization approaches include *(1) A priori methods*, transforming the MOP into a single-objective problem, *(2) Interactive methods*, incorporating preferences of decision makers throughout the optimization process, *(3) Pareto-dominated methods*, maintaining a diverse set of Pareto-optimal solutions without aggregating objectives, and *(4) New dominance approaches*, introducing modified definitions of dominance for flexibility in selecting solutions beyond strict Pareto optimality [24].

The classification by inspiration on the other hand, visualized in Figure 2.5, includes following:

- **Biology-Inspired Algorithms** are based on activities and behaviors which can be found in nature. They are broadly sub-divided into evolution based algorithms, also known as Evolutionary Algorithms (EA), and swarm based algorithms. Where EA are stochastic search methods, inspired by natural selection and evolution from nature, including Genetic Algorithms (GA), Differential Evolution (DE), Evolutionary Programming (EP), Evolutionary Strategy (ES), Harmony Search Algorithm

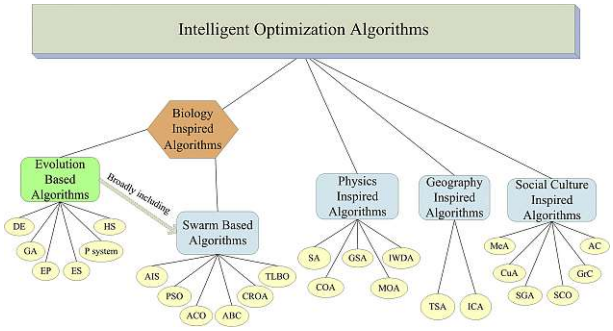


Figure 2.5: Classification of optimization algorithms by its source of inspiration [24]

(HS) and Membrane Computing (P system) [24]. Well-known representatives of GA are for example NSGA-II and SPEA2 which are also relevant in Chapter 5 as reference algorithms.

Swarm based algorithms on the other hand try to mimic the collective behaviors of social organisms like birds, bees, or ants. Besides Particle Swarm Optimiatzion (PSO), mostly relevant for this thesis and more detailed described in 2.5, they also consist of the more exotic algorithms such as Artificial Immune System (AIS), Ant Colony Optimization (ACO), Artificial Bee Colony (ABC), Coral Reef Optimization algorithm (CROA) or Teaching-Learning Based Optimization (TLBO) [24].

- **Physics-Inspired Algorithms** uses mechanisms which try to leverage principles of physics and imitate physical laws in nature. For example Simulated Annealing (SA) mimics the cooling process of metals to find global optima and Gravitational Search Algorithms (GSA) using gravitational forces to guide through the search space. Beside these two common algorithms there are three more crucial ones, namely Chaotic Optimization Algorithm (COA), Magnetic Optimization Algorithm (MOA) and Intelligent Water Drops Algorithm (IWDA) [24].

- **Geography Inspired Algorithms** try to simulate spatial and territorial behaviors. Including Tabu Search (TS), which explores the search space by a sequence of moves, and Imperialistic Competition Algorithm (ICA), which models competition among empires for colonies, whose amount determines the power [24].

- **Social Culture Inspired Algorithms** are based on the principles and mechanisms observed in human societies and cultural interactions. Among others, they include Memetic Algorithm (MeA), combining evolutionary mechanisms with local search strategies as well as Granular Computing (GrC), which mimics human thought processes by dividing information into granules for hierarchical problem-solving [24].

2.4.4 Performance Metrics for Evaluation

As described above, there are many diverse strategies and algorithms for multi objective optimization, which occur to have different strengths for different types of optimization problems. Additionally, it needs to be kept in mind that new algorithms which will be proposed in literature must always be compared to the others.

When it comes to evaluating and comparing the proposed solution sets, MOO brings the complexity of balancing multiple, often conflicting objectives. Additionally, in contrast to single objective optimization where single solutions can be compared, whole sets of approximated Pareto fronts need to be evaluated.

Over the years, researchers proposed different quality indicators and mathematical metrics, which can be categorized as followed. The indicators used in this thesis for the evaluation and the comparison of the proposed algorithm are described in detail in section 5.

Pareto Dominance-Based Indicators: Those indicators evaluate the quality of solution sets with regards to how well they approximate the true Pareto front. The most common indicator is the Hypervolume (HV), which calculates the volume of the objective space which is dominated by the solutions bounded by a reference point [98]. Other examples are the Generational Distance (GD) and Inverted Generational Distance (IGD), which calculates the average distance from each solution to the closest point on the true Pareto front (GD) and the other way around (IGD) [87].

Diversity Indicators: As the name suggests diversity indicators focus on how well the algorithm maintains solution diversity. The indicators evaluate the distribution and spread of solutions across the Pareto front. For example, the Spacing (S) indicator [79] measures the distribution of solutions across the Pareto front by calculating the variance in distances between consecutive solutions, while the Spread indicator [29] evaluates the distance between the extreme solutions and the overall diversity among solutions.

Convergence-Diversity Composite Indicators: These indicators provide a single metric value, trying to integrate aspects convergence (closeness to the true Pareto front) and diversity (distribution across the front). A well-known example is the ϵ-Indicator [58], designed to evaluate and compare Pareto front approximations based on their proximity to an ideal reference set. It measures how much a solution set needs to be shifted along each objective to dominate another set. It is particularly useful when comparing approximation sets generated by different optimization algorithms [99].

Other Indicators for Specific Aspects: In addition to evaluating the quality of the solution sets, other important quality aspects of an algorithm while computing solution sets need to be considered. For example, one indicator is the *error ratio*

which shows the proportion of solutions in the generated set that are not Pareto-optimal [55]. Another one is the computational time for an algorithm to propose a solution set. Especially for real life environments, where timing for a decision is critical, this can be a deciding factor for choosing the algorithm.

2.4.5 Handling Dynamic and Uncertain Environments

In the area of mobile edge cloud offloading, the users often physically move through the real world, resulting in a continuous change of the whole underlying infrastructure used for optimizing the possible offloading deployments. Especially, computational nodes on the edge usually benefit from the proximity of the mobile phone by low latency. Or in other words, when the change of the infrastructure is not applied to the algorithm dynamically, moving away from the node potentially resulting in drastic increase of latency and therefore making a potential calculated solution obsolete or even not valid regarding the constraints. Keeping that in mind, handling dynamic environments is critical for the use of MOO algorithms in real-world scenarios - also concerning MECO and other different fields like real-time logistics, economics or streaming data applications [8].

The primary challenge in Dynamic Multi-Objective Optimization (DMOO) is to develop algorithms not only capable to continuously use changed input information, but also efficiently track and adapt to the resulting, dynamic Pareto-optimal fronts [48]. Traditional static optimization methods, where a fixed set of solutions is searched, often do not provide the necessary mechanism to handle temporal variations effectively. Therefore, techniques such as evolutionary algorithms (EAs) and particle swarm optimization (PSO) have been adapted for DMOO by integrating memory, prediction and diversity maintenance mechanisms [49]. For example, memory-enhanced algorithms store previously discovered solutions, allowing the algorithm to use them when similar conditions repeatedly appear. Vice versa, prediction-based approaches use knowledge of previous environments to predict the evolution of objectives and their resulting Pareto fronts. A comprehensive overview of proposed algorithms designed and extended for solving DMOO can be found in [49].

Additionally, Helbig et al. [49] outlines the challenges in evaluating the performance of DMOO algorithms. Unlike traditional metrics, such as HV, GD, described in section 5.2.2, the performance measurements for DMOO always need to capture how well algorithms adapt to changes over time, such as the stability measure and wins-losses approach [49].

2.4.6 Preference Articulation and Decision Making

In many real-world situations where MOO is used, the goal is to select a single solution from a resulting Pareto front - a set of many diverse and non-dominated solutions, with different trade-offs of the conflicting objectives. However, beside picking a random solution, the decision-maker (DM) may have specific preferences for certain solution attributes or regions of the Pareto front [60].

For example, the propose mobile offloading model by De Maio et al. [26] (illustrated in Figure 2.3) introduces the user preferences as third input, next to infrastructure and application information, for the decision engine to calculate and select one valid deployment. Hence the users are provided with the ability to express their preferences, e.g. being more interested in saving battery lifetime and accepting higher costs.

These preferences can be incorporated through reference points, which represent desirable objective values that the DM bias to achieve [90]. This allows the DM to focus the search on regions of interest (ROI) rather than the entire Pareto front. There are various extensions of MOO algorithms, such as R-NSGA-II and SMPSO/RP [68], introducing reference points to align the optimization process with solutions which meet the DM's preferences.

A further discipline in this area is allowing the DM to interactively change the preferences over time in order to avoid the restart of the optimization from scratch each time preferences shift [8].

2.5 Particle Swarm Optimization

As described in 2.4.3, Particle Swarm Optimization (PSO) is a intelligent optimization algorithm classified as biology by trying to mock the social behaviors of honey bees, bird flocks and fish schools [24]. Initially proposed by Kennedy and Eberhart in 1995 [34], the basic idea behind this methodology is to use a collection of individual particles - without a central control but sharing information with each other - to explore the fitness landscape of a problem [75]. Since then it has become a widely spread research area in science literature and is used in many publication fields, including engineering, machine learning, complex systems modeling and task offloading [57].

2.5.1 Core Components of PSO

To describe the fundamental mechanics of the PSO it is necessary to understand the following key components which distinguish it from other optimization techniques such as Genetic Algorithms (GAs) [22] [46]:

Particles and Swarm: A particle is one potential solution for the given optimization problem. It is a specific combination of variable values for the objection function resulting in a concrete position within the solution space [52]. The collection or set of individual particles is commonly known as the swarm. In each iteration it represents one population of solutions. Although the particles are individual, they work and learn together to a certain extent within the swarm [46].

Position and Velocity: At each point of time, a particle consists of a current position $x_i(t)$ and a velocity $v_i(t)$. The position corresponds to a specific candidate solution while the velocity defines the direction where the particle is going within the search

space. Based on the particle's own experience and the experiences of others in the swarm, the position and the velocity is updated in each iteration [20].

Best Positions (p_{best} and g_{best}): Each particle within the swarm remembers its best discovered position, called its personal best (p_{best}). This position represents the best-known solution for that particle so far and outlines considerations for further steps. where to go next [75]. Additionally, the swarm itself also maintains the best solution found by any particle within the swarm, known as the global best position (g_{best}). In addition to the p_{best}, the g_{best} also influences the calculation of the velocity for each particle, pushing all particles towards the best-known region in the search space [57].

2.5.2 Core Principle of the particle movement

Concluding, PSO's core principle is based on the iteratively movement of the particles through the search space and influenced by the current direction $v_i(t)$, the personal best p_{best} as well as the swarms global best g_{best}. The calculation of the new velocity for the next iteration $v_i(t+1)$ and the update of the position can be explained as the following [20]:

$$v_i(t+1) = w \cdot v_i(t) + c_1 \cdot r_1 \cdot (p_{best,i} - x_i(t)) + c_2 \cdot r_2 \cdot (g_{best} - x_i(t)) \tag{2.3}$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{2.4}$$

In order to facilitate balancing the exploring new areas and the exploiting known good areas, the calculation to the new direction, as seen in equation 2.3, is driven by the following three aspects [46]:

Inertia Term ($w \cdot v_i(t)$): The inertia term maintains a particle's current direction. The inertia weight w is the main factor to control the exploring of complete new areas (when w is large). The inertia weight is often reduced over time to allow broad exploration at the beginning and focus on best found solutions towards the end [46].

Cognitive Component ($c_1 \cdot r_1 \cdot (p_{best,i} - x_i(t))$): The cognitive component forces the movement of a particle back to its personal best found position. The influence of the particle's own experience is controlled by the cognitive coefficient c_1. To avoid getting stuck in local optima, the random factor r_1 (a random number between 0 and 1) allows particles to explore slightly different paths of each iteration [20]. [20]

Social Component ($c_2 \cdot r_2 \cdot (g_{best} - x_i(t))$): The social component represents the influence of the swarm's knowledge, guiding particles in the direction of the global best position. The social coefficient c_2 manages the weight of this influence, while another random factor r_2 (a random number between 0 and 1), adds again some flexibility to the movement. This component is the main factor to PSO's social aspect (sometimes also referred to swarm intelligence), helping particles to learn from the swarm's collective success [75] [46].

2.5.3 Enhancing Exploration with Turbulences

In addition to the previously mentioned mechanism promoting diversity, some PSO variations go a step further and introduce turbulences into their iterative approach, helping particles escape local optima and improve the exploration of the search space [50]. Since Particle Swarm Optimization is inspired by the biological behavior of swarms like bird flocks (see section XY), this feature was given the appropriate name *turbulence* - originating from the unpredictable movement within the air, created when air currents of different speeds and directions meet [6].

These turbulences can prevent premature convergence by allowing particles to occasionally jump to new areas of the search space. Thus they can explore potential solutions outside their current trajectory. Turbulences are especially helpful in complex landscapes where local Pareto fronts can trap particles [50]. Turbulence functions in PSO can occur in various forms, such as the following:

Velocity Perturbations: The turbulence can be introduced by adding random noise to the velocity of particles. This noise, or perturbation, disrupts the movements which particles might otherwise follow according to cognitive and social components. Giving them - so to say - an extra push to explore other regions [50].

Random Position Adjustments: Another variant is to randomly relocate particles to complete new positions within the search space. These position adjustments prevent the swarm from becoming overly concentrated in specific areas, promoting diversity in the swarm. This type of turbulence is usually not used regularly, as large or too many position shifts can disrupt convergence [50].

Adaptive Turbulence: While turbulences are beneficial for exploration, excessive turbulence can hinder convergence and lead to a loss of focus. The adaptive turbulence strategy attaches importance to balancing the use of turbulence throughout the optimization process or based on the current state of the swarm. This can be particularly effective if, for example, most of the particles converge in a small area, hence, the algorithm can increase turbulence to promote exploration [6].

Mutation-Inspired Turbulence: Some PSO variants make use of mutation, based on the principles of genetic algorithms [76]. This means that, with a certain probability, a particle's position or velocity undergoes a random alteration. These mutation-like changes introduce diversity in a controlled manner and are especially useful in handling discrete optimization problems [88].

2.5.4 Multi Objective Particle Swarm Optimization

PSO was initially proposed to find a single optimized solution to a problem with maximizing or minimizing one objective function. However, many real life problems as well as problems in literature require multiple objectives to be optimized [52]. Therefore, it did

not took that long until the first Multi-Objective PSO (MOPSO) algorithm was proposed in 1999 [22].

By solving problems with multiple – usually conflicting – objectives, there is not one single best solution. Instead, a collection of solutions is searched, also known as the Pareto front, where each solution in the set is Pareto optimal - in other words - non-dominated by any of the objectives (for more details see section 2.4.1).

Transferring this phenomenon into PSO means that the particles do not converge into a single point but seek to approximate the Pareto front. In the best case, the result is a diverse set of solutions that offer different compromises among objectives. This fact requires MOPSO to manage multiple best solutions simultaneously rather than focusing on one single global optimum g_{best}.

2.5.5 MOPSO Archive and Leader selection

As described in 2.5.2, PSO each particle is influenced by its personal best p_{best} and the swarm's global best solution g_{best}. MOPSO replaces the singular global best solution with a collection of non-dominated solutions, called archive [76]. This archive always represents the current approximation of the Pareto front. After each iteration, the archive is updated to store only non-dominated solutions. Which means that not only newly discovered non-dominated solutions are added, but also existing archived solutions which are now dominated by a better solution will be removed from the archive.

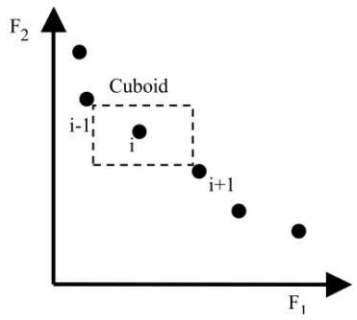


Figure 2.6: Calculation of Crowding Distance [76]

During each iteration, particles in MOPSO need to select a leader from the archive for the calculation of the velocity. Besides just a simple random selection, a more effective strategy is to be chosen based on the crowding distance. This is a measure solution density in objective space, which can promote less-crowded areas on the Pareto front as leaders. As shown in Figure 2.6, the crowding distance for each solution is computed by calculating the proximity between the solutions and their nearest neighbor which is based on sorted objective functions. To preserve solution on the edges of the Pareto front the crowding distance is assigned infinite [76].

Whenever the archive reaches its capacity, crowding distance is also used to manage which solutions are retained. In this case, solutions with the lowest crowding distance are removed to make space for new, potentially more diverse solutions. This approach helps the archive to maintain a well-spread set of non-dominated solutions, ensuring that it covers the entire Pareto front without excessive clustering [76].

CHAPTER 3

Related Work

This chapter explores the state-of-the-art approaches, tools and frameworks that are related to the research area of this thesis. First, an overview of **Multi-Objective Optimization in Literature** is given, where various strategies and algorithms, with a focus on Multi-Objective Particle Swarm Optimization (MOPSO) are highlighted. Then, state-of-the-art **MOO Development Frameworks** are explored by reviewing their capabilities in implementation, testing and evaluation of algorithms. Finally, **Edge Simulation Frameworks** are explained, which allows the evaluation of algorithms in realistic mobile edge computing scenarios. The general core features are outlined and common frameworks are presented and compared.

3.1 Multi-Objective Optimization in Literature

As described in section 2.4, the area of multi-objective optimization comes with different challenges and considerations. Over the years, various algorithms have been proposed and introduced innovative mechanisms to address the challenges in MOO, such as managing non-dominated solutions, maintaining diversity and adapting to specific problem requirements [24].

When proposing new methods and optimization algorithms, the evaluation and comparison to well-established reference algorithms, such as NSGA-II and SPEA2, ensure better classifiability throughout the research area [9]. Hence, this section provides a comprehensive overview of innovative approaches with a focus on particle swarm optimization algorithms and a review of common reference algorithms used for comparison.

3.1.1 Particle Swarm Optimization Based Algorithms

Multi-Objective Particle Swarm Optimization (MOPSO) algorithms are widely used for solving multi-objective optimization problems due to their adaptability and relatively

simple nature [24]. Some of their key strategies and innovative approaches can be categorized as following:

Archive Management: As described in section 2.5.5, MOPSO algorithms rely on effective archive management strategies to maintain the set of non-dominated solutions. This archive on the one hand represents the Pareto front approximation found and on the other hand serve as swarm knowledge guiding particles. Depending on the swarm size and the iterations, this history of found solutions can increase drastically while processing requires a strategy for maintaining it.

For example, MOPSO-CD [76] uses a crowding distance metric to select representative solutions from the Pareto front which are evenly spread across the front in order to push diversity. Zhang et al. [93] introduced a MOPSO algorithm using a multi-archiving strategy (MSMOPSO), where separate archives for the different objectives are maintained. This enables the algorithm to focus its search on specific regions of the front.

Velocity and Constraint Strategies: There are different strategies on how the velocities for the particles are calculated and how the constraints are handled within the position update. SMPSO [20], known as Speed-constrained MOPSO, for example, introduces limits on particle velocity to prevent particles from diverging too far from the search space. By bounding velocities, SMPSO enhances convergence rates and avoids premature stagnation.

Speaking of feasible solution space, Nshimirimana et al. [71] proposed that NF-MOPSO uses a death penalty constraint handling mechanism, which calculates the velocity until all constraints on the new position are valid. Another strategy is the collection of feedback within the particle evolution to dynamically adjust the guidance of the particles proposed in the robust MOPSO with feedback compensation (RMOPSO-FC) [47] or used in Adaptive MOPSO (AMOPSO) [40].

Discrete Problem Handling: As the search for a valid deployment within MECO dealt with in this thesis, many optimization problems require solving combinatorial problems with discrete decision variables, where classic MOPSO algorithms, which are defined in continuous space, need to be adapted. For example, Qiao et al. [75] proposed a particle encoding strategy and dedicated operators for updating the velocity, which adapts classical PSO mechanisms to handle combination decision spaces. This approach is used as a base for the proposed algorithm in this thesis which is described in section 4.2.

Another example is DPSO-AIW [41], a PSO with adaptive inertia weight, which uses a two-layer coding structure to encode the discrete chromosomes and try to enhance the diversity by adjusting the variation of the inertia weight. Other algorithms, like set-based PSO (S-PSO) treats particles as sets and their movements involve operations on sets instead of arithmetic operations [17].

Decomposition-Based Approaches: Decomposition strategies divide multi-objective optimization problems into simpler sub-problems, where each of them is optimized independently, however, the information is shared [92]. This approach is especially effective for handling high-dimensional objective spaces. Decomposition is mainly used in Multi-Objective Evolutionary Algorithms (MOEAs), especially in the MOEA/D algorithm.

However, also some PSO algorithms were proposed, where each particle solves a sub-problem by making use of MOEA/D, for example, MOEA/D-PSO [92] and MOPSO/D [73]. MPSO/D [25] is built on top of MOPSO/D, adapting the decomposition strategy dynamically during optimization.

Hybrid Approaches: Approaches such as MOEA/D-PSO are called hybrid MOPSO algorithms, which combine basic PSO operations with other optimization mechanisms, regarding local search methods or genetic operations. For example, HPSO [89] uses evolutionary techniques to improve the diversity of solutions by taking advantage of genetic operators such as mutation and crossover. HD-MOPSO [51] on the other hand merges discrete MOPSO with heuristic methods for local refinement designed for combinatorial problems.

3.1.2 Reference Algorithms for MOO Performance Evaluation

When proposing a new optimization algorithm, it is essential to evaluate its performance against well-established reference algorithms [9]. This thesis uses NSGA-II, NSGA-III, MOCell and SPEA2 as benchmark for comparison, as they are commonly used in the research community regarding multi-objective optimization and their available implementation in the used development framework *jMetal* as described in 3.2.2.

- **NSGA-II [29]** (Non-dominated Sorting Genetic Algorithm II) is a genetic algorithm that classifies solutions into different Pareto fronts by their dominance levels by using a non-dominated sorting procedure. In order to enhance diversity, the algorithm uses the crowding distance measure to prefer non-crowded regions during selection. Over the years it has become a standard benchmark algorithm for two or three objectives.

- **NSGA-III [54]** (Non-dominated Sorting Genetic Algorithm III) is an extension of NSGA-II and designed to efficiently handle more than three objective problems. Additionally, it introduces a reference point-based diversity mechanism, where each solution will be associated to the closest reference point in the objective space. When it comes to selecting, solutions based on the least occupied reference point will be preferred. This mechanism replaces the crowding distance measure NSGA-II, and hence enables a better computational efficiency in many-objective problems.

- **MOCell [94]** (Multi-Objective Cellular Genetic Algorithm) is characterized by its cellular structure and localized genetic operations, where only nighborhoods are

considered for crossover and mutation operators. This is combined with a global optimization strategy, by using an external archive, which maintains non-dominated solutions found during the search process and serves as a global repository for selection.

- **SPEA2 [61]** (Strength Pareto Evolutionary Algorithm 2) uses a fitness assignment strategy, which assigns strength value to each solution in the population, measuring the number of solutions dominated by it. The algorithm also uses an external archive to maintain elite solutions, which influnces the search process. When this archive exceeds the size limit, it uses clustering mechanisms to select solutions which are removed from the archive by focusing on diversity.

3.2 MOO Development Frameworks

The implementation, testing and analysis of MOO algorithms have been significantly advanced by specialized frameworks, providing researchers with reusable and extensible tools for conducting structured and reproducible experiments.

This section reviews scientific work related to key MOO frameworks by highlighting the core features as well as the key considerations for choosing the right tool.

3.2.1 Core Features

In general, multi-objective optimization (MOO) development frameworks share a set of core features and competences to facilitate the implementation, evaluation and benchmarking of MOO algorithms.

A key feature of any MOO framework is a diverse library of multi-objective optimization algorithms, such as classic evolutionary (NSGA-II and SPEA2) or PSO-based algorithms. The same applies to the availability of problem definitions, where most frameworks include standardized benchmark suites (ZDT, DTLZ, WFG, etc.) which, additionally, allow cross framework comparisons [91].

When thinking about evaluation and comparing algorithms with each other, frameworks usually include several standardized quality indicators and performance metrics. Additionally to these statistical values, tools for visualization of the Pareto front (in 2D or 3D) improve interpreting MOO results and understanding the trade-offs. Some frameworks also provide advanced tools, such as parallel coordinates plots, in order to visualize the distribution of solutions in the decision space.

Another discipline covered by some MOO frameworks is the dynamic problem handling, where objectives, constraints or other conditions can vary during the optimization process, e.g. via event-driven updates [70]. Thinking of optimizing large populations across many objectives can be computationally intensive, which can make palatalization important, by offering not only multi-threading, but also distributed computation [67].

Last but not least, a major consideration for choosing a MOO framework is the extensibility of the previous mentioned components: algorithms, problems, custom metrics and solution representations such as flexible encoding schemes for handling complex real-world problems in diverse research areas [69]. Due to this flexibility, the frameworks themselves can benefit from active community, as capabilities and components are contributed and extended.

3.2.2 State-of-the-art Frameworks

As described in the previous section, state-of-the-art MOO frameworks typically share core capabilities. Each of them usually offer unique strengths and limitations, which are attractive for different user bases and research needs. For better understanding, the following provides brief descriptions of prominent frameworks:

jMetal: Started as scientific side project for evaluating MOO algorithms, jMetal was first published in 2011 by Durillo and Nebro [33] as an open source Java based MOO framework. Since, it was constantly extended as well as redesigned [69] and has evolved to a highly modular, easy-to-use and extensible framework within the scientific community.

jMetal includes various state-of-the-art multi-objective algorithms and supports a variety of benchmark problems (e.g., DTLZ, ZDT). Its object-oriented design and component based architecture allows researchers to flexibly integrate new problems and algorithms, accelerate to evaluate them and adapt it to their research goals [69].

Making use of this high degree of flexibility, jMetal was used in this thesis to implement and evaluate the proposed algorithm as well as modeling the concrete problem definition of MECO and their solution representation.

jMetalSP: As described in section 2.4.5, in many real-world applications data arrives continuously and dynamically, requiring solutions that can adapt to new information in real-time.

To address this gap, jMetalSP [67] was developed by the same authors as jMetal, combining the strenghts and capabilities of the existing framework with Apache Spark cluster computing system. Additionally, due to its streaming- and event-driven design, jMetalSP provides new streaming data sources (like Apache Kafka, MQTT, etc.) and provides interfaces for solving dynamic problems that can continuously change over time [70].

MOEA Framework: The MOEA Framework [44] is another Java based platform for MOO which is widely used in academic research and benchmarking. It is also shipped with a range of algorithms and problem definitions as well as various performance metrics and provides enhanced visualization.

Compared with jMetal, both frameworks are considered as powerful tools in the research of MOO [77]. On the one hand, jMetal offers a better flexibility, which is beneficial for custom applications, while MOEA Framework on the other hand stands out in standardized benchmarking and performance analysis [91].

PlatEMO: PlatEMO [84] is a MATLAB-based framework which especially focuses on evolutionary algorithms. Unlike others, MATLAB toolboxes often provide algorithm-specific features which include a suite of various implementations of single- and multi-objective evolutionary algorithms and problems. PlatEMO's goal is to provide a accessible framework for various optimization techniques for users across different research areas without requiring extensive programming expertise, by providing a friendly, graphical user interface and a modular design [84].

DEAP: Distributed Evolutionary Algorithms in Python (DEAP) [38] is a Python-based framework for genetic and evolutionary algorithms. Its key strengths are the simplicity for rapid prototyping and testing of ideas [27].

pymoo: With pymoo another Python framework was introduced by Blank and Deb in 2020 [11]. It offers a library of state-of-the-art MOO algorithms, including genetic algorithms, particle swarm optimization and others. Unlike DEAP, pymoo is more focused on multi-objective problems and provides detailed visualization capabilities, which makes it useful for academic research [11].

3.3 Edge Simulation Frameworks

This thesis deals with an optimization algorithm to solve a deployment problem in a mobile edge cloud offloading environment with focus on application runtime, battery life and user costs. In order to evaluate the algorithm in question successfully, it is necessary to simulate specific aspects of edge computing, which are resource allocation, latency, bandwidth constraints and energy consumption.

Simulation frameworks for edge and fog computing like SLEIPNIR[1] [26], FogTorchPI[2] [13], iFogSim[3] [43], EdgeCloudSim[4] [82], MobFogSim[5] [74] and YAFS (Yet Another Fog Simulator)[6] [59] have been proposed to help researchers achieving successful performance analysis under diverse conditions by providing controlled, customizable environments.

[1] https://github.com/vindem/sleipnir
[2] https://github.com/di-unipi-socc/FogTorchPI
[3] https://github.com/Cloudslab/iFogSim
[4] https://github.com/CagataySonmez/EdgeCloudSim
[5] https://github.com/diogomg/MobFogSim
[6] https://github.com/acsicuib/YAFS

3.3.1 Core Features

The capabilities of edge and fog simulation frameworks can be categorized across several areas described below. While certain features are found across these frameworks, others are unique and support different research needs and applications [35].

Resource Management and Application Modeling: One of the core capabilities of these frameworks is the modeling computational, storage and network resources as well as applications and their tasks. The frameworks can simulate the tasks running on the nodes, data transfer requirements and computational load, by allocating the resources across multiple nodes [56].

For example, the frameworks iFogSim and EdgeCloudSim provide more detailed resource modeling options, such as allocation policies in order to allow dynamic adjustments. This makes them useful for real-time data processing [35]. FogTorchPI and SLEIPNIR with its probabilistic approach for allocation handling, on the other hand, is better in simulating unreliable environments [15].

Quality of Service (QoS) Modeling: Modeling of QoS constraints, such as latency, bandwidth and reliability, allow researchers to define and simulate scenarios which enforce these constraints to a varying degree [35].

While most frameworks deal with a more deterministic approach by using fixed sets of QoS constraints, in order to evaluate concrete configurations, SLEIPNIR as well as FogTorchPI are designed to use probabilistic QoS evaluations, which provide insights into the likelihood of achieving target QoS levels under varying conditions [15].

Network and Mobility Modeling: Modeling of the network, using different topologies, latency and bandwidth for data transmission, is a feature provided by most frameworks. Compared with others, YAFS provides the most advanced capabilities for network configurations and routing. This framework is ideal for simulating network-intensive applications [35].

Another aspect in edge computing is that user devices often move across different network zones, which requires to model this mobility of devices and other nodes. Therefore, MobFogSim offers unique features for simulating real-world user mobility and its impact on system performance [74].

Energy Consumption and Cost Modeling: Especially for mobile devices, the battery lifetime can be an important factor when deciding whether to offload tasks or not. Not only considering the energy consumption for the local execution, some frameworks also use energy models of power consumption for sending and receiving data via the network in their resource allocation strategies [26].

Speaking about offloading, also user costs can be a relevant factor for decisions, which requires simulation of costs when computational nodes in the infrastructure

are allocated. Therefore, SLEIPNIR includes an enhanced cost model proposed in [14], considering higher costs for near edge resources which are available with lower latency.

Monte Carlo Simuations: Most frameworks' architecture is designed for being event-driven, which is valuable for dynamic simulation scenarios [56]. SLEIPNIR and FogTorchPI stands out by its support for stochastic analysis through Monte Carlo simulations. This probabilistic approach is especially useful for real-world scenarios where for example network conditions or the availability of resources can act unpredictably [15].

3.3.2 FogTorchPi Extended

As already mentioned before, FogTorchPI stands out as its probabilistic features allow Monte Carlo simulations. It is built on top of the original FogTorch[7] framework, which was proposed by Brogi and Forti in 2017 [12].

FogTorch was designed to provide a simple and effective way to evaluate the deployment of applications in edge and fog computing environments. It introduced the specification of infrastructure resources, application requirements as well as basic QoS metrics like latency and bandwith. The FogTorch framework operates with a deterministic approach, meaning it evaluates each deployment option against fixed QoS constraints without accounting for variability or uncertainty in resource availability.

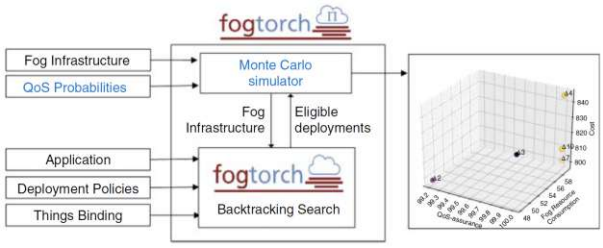


Figure 3.1: Bird's-eye view of FogTorchPI [15]

Here FogTorchPI comes into play. It is based on the work of [13] and represented in Figure 3.1. Its input consists of following [15]:

- **Infrastructure** $\mathcal{I}$ specifies the devices, edge and cloud nodes as well as their hardware resources (CPU, RAM, storage) and software (e.g., OS, libraries, frameworks. In addition, it contains the communication links between each node including quality of service constraints, such as latency, up- and download-bandwidth, as well as cost for purchasing the computation power of these nodes.
- **Application** $\mathcal{A}$ specifies the required hardware (CPU, RAM, storage), software (e.g., OS, libraries, frameworks) and QoS needs of each application component.

[7] https://github.com/di-unipi-socc/FogTorch

- **Things binding** ϕ represents the mapping of each application component to an actual node in $\mathcal{I}$.

- **Deployment policy** δ adds additional constraints for application components where it can be deployed, according to security or business reasons.

Based on this work De Maio et al. [26] developed and proposed an extended version of FogTorchPI, called SLEIPNIR[8], which is used for Monte Carlo simulations in this thesis. Relevant for the scenario of mobile offloading, it adds item support for mobile devices in the infrastructure $\mathcal{I}$, including the resource characteristic of battery life. Additionally, an energy consumption model for both, local execution on the mobile devices and for network transfer while offloading, were introduced.

De Maio et al. [26] also included an enhanced user cost calculation model, which represents the different scenarios of local execution (no cost) by offloading as well as execution on an edge node with additional costs. Additionally, user preference parameters regarding the decision on lower latency versus cheaper price are offered.

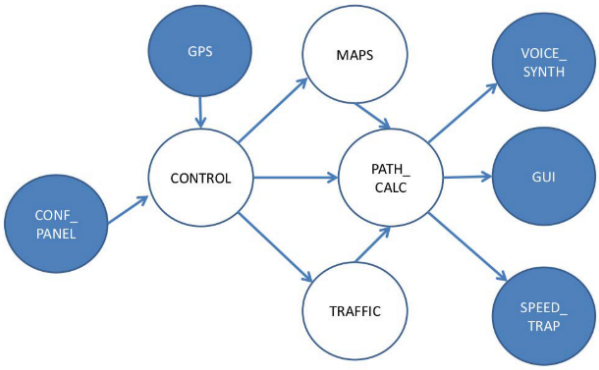


Figure 3.2: DAG of Navigator app [26]

Another feature which extends FogTorchPI by [26] is the use of Directed Acyclic Graphs (DAGs) for representing mobile applications to simulate the offloading. Using this approach it is not only possible to model the application's tasks with their requirements but includes a definition whether the task can be offloaded not. Additionally, dependencies between the tasks are represented as links within the DAG. The extended framework provides the following four mobile applications:

- **Navigator** represents the behavior of a GPS navigation software, based on the work in [3]. For better illustration, Figure 3.2 shows the structure of the DAG for the Navigator application, where filled nodes indicate the need of local execution.

[8]https://github.com/vindem/sleipnir

- **Facerecognizer** models an image processing application for recognizing a face in a picture, as described in [23]
- **Antivirus** charaterizes the behavior of an antivirus scanning files, as described in [19]
- **Chess** models a chess game between a user and an AI, based on [23]. The visualization of the application model mapped to an infrastructure can be seen in Figure 4.1.

CHAPTER 4

MDPSO Algorithm for MECO

This Chapter focuses on the main contribution of this thesis, proposing the Multi-Objective Discrete Particle Swarm Optimization (MDPSO) algorithm tailord to MECO scenarios. It starts with a detailed **Deployment Problem Definition**, explaining the application and infrastructure model, constraints as well as the objective functions based on [26]. Then the **Algorithm Design** is described in greater detail by dealing with each of its components. It is started with the leader archive strategy and followed by the global best selection mechanism. Then the particle encoding for the discrete search space of MECO is outlined. Finally, all components are united and illustrated according to the two staged algorithm procedure.

4.1 Deployment Problem Definition

As described in 3.3.2, this thesis uses the extended version of FogTorchPI proposed by [26]. Therefore, also the definitions and modeling of the mobile edge infrastructure, application as well as the deployment problem with its constraints are based on [26], and described in greater detail in the following section.

4.1.1 Application and Infrastructure Model

De Maio et al. [26] defines an Application $\mathcal{A}$ as Directed Acyclic Graphs (DAGs) where the graph nodes represent the computational tasks and the directed edges indicate the dependencies and order of execution. A visual example of DAG modeling a navigator application for a mobile device can be found in Figure 3.2.

It can be formally described as $\mathcal{A} = \{\mathcal{T}_a, \mathcal{L}_a\}$, where:

- $\mathcal{T}_a$ is a set off all tasks. Each task t_i is defined as

$$t_i = (MI, RAM, DATA_{in}, DATA_{out}, OFF) \tag{4.1}$$

MI represents the millions of instructions, RAM is the memory requirement, $DATA_{in}$ and $DATA_{out}$ represent input and output data sizes, and OFF is a binary indicator (1 if the task can be offloaded, 0 otherwise) [26].

- $\mathcal{L}_a \subseteq \mathcal{T}_a \times \mathcal{T}_a$ is a set of dependencies between tasks, represented by edges. Each edge l_{ij} between tasks t_i and t_j enforces an execution order, where t_i must be completed before t_j can start. Additionally, each edge l_{ij} also has a maximum allowed latency $latency(l_{ij})$ and bandwidth requirements $bw(l_{ij})$ which are associated in [26].

The Mobile Edge Computing (MEC) infrastructure $\mathcal{I}$ used in this work and proposed by [26], contains computational nodes $\mathcal{N}_\mathcal{I}$ and network connections between them $\mathcal{L}_\mathcal{I}$, defined as $I = \{\mathcal{N}_\mathcal{I}, \mathcal{L}_\mathcal{I}\}$, where:

- $\mathcal{N}_\mathcal{I} = \{\mathcal{C}_\mathcal{I}, \mathcal{E}_\mathcal{I}, md\}$ is the collection of cloud nodes $\mathcal{C}_\mathcal{I}$, edge nodes $\mathcal{E}_\mathcal{I}$ and mobile device md. As described in 2.2, cloud nodes in $\mathcal{C}_\mathcal{I}$ are typically characterized as high-resource nodes which are far from the user, edge nodes in $\mathcal{E}_\mathcal{I}$ are located closer to users, with lower latency but limited resources and the mobile device md has limited computational power and battery life. Each of those nodes $n_j \in \mathcal{N}_\mathcal{I}$ has defined parameters

$$n_j = (CORES, MIPS, RAM, STORAGE) \tag{4.2}$$

  where CORES is the number of CPU cores, MIPS is the processing speed in million instructions per second, RAM and STORAGE are the memory and storage capacities. Additionally to them, edge nodes $\mathcal{E}_\mathcal{I}$ and mobile device md defines the *coords* which represent the GPS coordinates. For md the amount of energy of the device's battery is also stored by the parameter $BATTERY$

- $\mathcal{L}_\mathcal{I} = \mathcal{N}_\mathcal{I} \times \mathcal{N}_\mathcal{I}$ is the set of links l_{ij} between nodes n_i and n_j. The latency and bandwidth for each connection is defined as $latency(n_i, n_j)$ and $bw(n_i, n_j)$, where the link to itself having $latency(n_i, n_j) = 0$ and $bw(n_i, n_j) = \infty$

4.1.2 Constraints for a valid deployment

De Maio et al. [26] described a deployment $\mathcal{D}(\mathcal{A}, \mathcal{I})$ of an application $\mathcal{A}$ on an infrastructure $\mathcal{I}$ as mapping all of the tasks in $\mathcal{T}_a$ to the nodes in $\mathcal{N}_\mathcal{I}$ as well as the dependencies in $\mathcal{L}_a$ to the physical links in $\mathcal{L}_\mathcal{I}$, as:

$$(t_i, n_j) \in \mathcal{D} \iff t_i \text{ is mapped to } n_j \tag{4.3}$$

Additionally, the function ϕ, which returns the node $n_j \in \mathcal{N}_\mathcal{I}$ where a task $t_i \in \mathcal{T}_a$ is executed, is defined as:

$$\phi(t_i) = n_j : \exists (t_i, n_j) \in \mathcal{D} \tag{4.4}$$

Therefore, each task t_i can be mapped to:

- the mobile device $\phi(t_i) = md$, if it's executed locally
- a Cloud node $\phi(t_i) \in \mathcal{C}_\mathcal{I}$, if it's offloaded to the *cloud*
- an Edge node $\phi(t_i) \in \mathcal{E}_\mathcal{I}$, if it's offloaded to the *edge*

A deployment $\mathcal{D}(\mathcal{A}, \mathcal{I})$ is considered as valid when it satisfies specific criteria related to task assignment, resource limitations, execution order of tasks and dependency constraints within the infrastructure model [26]. More specifically, it is valid if it meets the following criteria:

1. *All Tasks Deployed Exactly Once* [26]: Each task $t_i \in \mathcal{T}_a$ must be deployed on exactly one computational node $n_j \in \mathcal{N}_\mathcal{I}$, ensuring every task is processed only once during the application execution:

$$\bigcup_{n_j \in \mathcal{N}_\mathcal{I}} \mathcal{D}(n_j) = \mathcal{T}_a \tag{4.5}$$

2. *Offloading Constraints* [26]: Tasks t_i which are not offloadable ($OFF(t_i) = 0$) must be deployed on the mobile device md:

$$(t_i, n_j) \in \mathcal{D}(\mathcal{A}, \mathcal{I}) \text{ and } OFF(t_i) = 0 \Rightarrow n_j = md \tag{4.6}$$

3. *Node Capacity Constraints* [26]: Each node n_j has limited resources, which must not be exceeded by the sum of the requirements of tasks deployed on this node. First, the total requirements for CPU cores (Equation 4.7), RAM (Equation 4.8) and storage (Equation 4.9) by the tasks on node n_j must be within the node's available capacity.

$$\sum_{t_i \in \mathcal{D}(n_j)} CORES(t_i) \leq CORES(n_j) \tag{4.7}$$

$$\sum_{t_i \in \mathcal{D}(n_j)} RAM(t_i) \leq RAM(n_j) \tag{4.8}$$

$$\sum_{t_i \in \mathcal{D}(n_j)} STORAGE(t_i) \leq STORAGE(n_j) \tag{4.9}$$

4. *Task Dependency Constraints* [26]: The execution order of tasks within the application must be fulfilled for any dependency $l_{ij} = (t_i, t_j) \in \mathcal{L}_a$, where the execution t_j starts only after t_i is completed.

5. *Latency and Bandwidth Constraints for Communication Links* [26]: The latency constraint ensures that the time used for transferring the data between two tasks t_i and t_j given via the communication link $latency(\phi(t_i), \phi(t_j)) : l_{ij} \in \mathcal{L}_{\mathcal{I}}$ must not exceed the maximum latency specified in $latency(l_{ij}) : l_{ij} \in \mathcal{L}_a$. Additionally, the bandwidth of the used communication link $bw(\phi(t_i), \phi(t_j)) : l_{ij} \in \mathcal{L}_{\mathcal{I}}$ must meet or exceed the bandwidth required by the task dependency $bw(l_{ij}) : l_{ij} \in \mathcal{L}_a$.

4.1.3 Objective Functions

Having the model of application $\mathcal{A}$, infrastrutucr $\mathcal{I}$ and deployment $\mathcal{D}(\mathcal{A}, \mathcal{I})$ defined, the goal is to search for valid deployments with focus on optimizing them among following objectives, which are defined by De Maio et al. [26].

(1) Minimize Application Runtime $RT(\mathcal{D}(\mathcal{A}, \mathcal{I}))$

The target of this objective is to minimize the total execution time, starting when the first task begins until the last task finishes [26]. The objective function is expressed as:

$$RT(\mathcal{D}(\mathcal{A}, \mathcal{I})) = \tau_{end} \tag{4.10}$$

where τ_{end} represents the time when the last task in the application's DAG completes execution. For each task t_i deployed on a computational node n_j, the runtime includes several parts:

- *Local Execution Time* $RT_{local}(t_i, n_i)$ represents the time taken for a task t_i to execute on a specific node n_i depending on the task's millions of instructions (MI) and the processing speed (MIPS) of the node [26]:

$$RT_{local}(t_i, n_i) = \frac{MI(t_i)}{MIPS(n_i)} \tag{4.11}$$

- *Offloading Time* $OT(t_i, n_i)$ stands for the total offloading time, including both, the time required to upload the task data to the target node $OT_{up}(t_i)$ and downloading the results back to the mobile device $OT_{down}(t_i)$. Both mainly depends on the required data to be sent and the connection link's bandwidth between n_i and md, described as [26]:

$$OT_{up}(t_i) = \frac{MI(t_i) \cdot instr\_size + DATA_{in}(t_i)}{bw(md, n_i)} \tag{4.12}$$

$$OT_{down}(t_i) = \frac{DATA_{out}(t_i)}{bw(md, n_i)} \tag{4.13}$$

assuming $instr\_size = 5 \times 10^{-9}$ based on [1]. Therefore, the total runtime for a task t_i offloaded to node n_j is defined as [26]:

$$RT(t_i) = \tau(t_i) + OT_{up}(t_i) + RT_{local}(t_i, n_j) + OT_{down}(t_i) \tag{4.14}$$

where $\tau(t_i)$ is the time at which the task t_i begins with the execution. For tasks executed directly on the mobile device, $RT(t_i) = \tau(t_i) + RT_{local}(t_i, md)$ since no offloading occurs.

(2) Maximize Battery Lifetime $BL(\mathcal{D}(\mathcal{A}, \mathcal{I}))$

This objective aims to maximize the remaining battery life of the mobile device by minimizing energy consumption during task execution and offloading [26]. The objective function is defined as:

$$BL(\mathcal{D}(\mathcal{A}, \mathcal{I})) = \frac{BATTERY(md) - E_d(md, \mathcal{D}(\mathcal{A}, \mathcal{I}))}{BATTERY(md)} \tag{4.15}$$

where $BATTERY(md)$ is the initial battery capacity of the mobile device and $E_d(md, \mathcal{D}(\mathcal{A}, \mathcal{I}))$ is the total energy consumed by the mobile device throughout the deployment. The energy consumption E_d is computed by integrating the overall power consumption over the runtime of the application [26]. The overall power consumption consists of the following parts:

1. *Processing Power Consumption* $P_p(md, \tau)$ [26]: The power used by the mobile device for local computation based on the CPU model designed by [4], described as:

$$P_p(md, \tau) = \sum_{i=0}^{CORES(md)} \beta_{freq}(i, \tau) \cdot U_{cpu}(md, \tau) + \beta_{base} \tag{4.16}$$

$$U_{cpu}(md, \tau) = \frac{\sum_{t_i \in \mathcal{D}(md)} MI(t_i, \tau)}{MIPS(md)} \tag{4.17}$$

   where β_{freq} and β_{base} are hardware-specific constants and $U_{cpu}(md, \tau)$ is the CPU utilization.

2. *Offloading Power Consumption* $P_{off}(t_i, n_j, \tau)$ [26]: The power used when offloading a task t_i depending on the connection type and network utilization:

$$P_{off}(t_i, n_j, \tau) = \epsilon_{conn}(n_j) \cdot U_{net}(t_i, n_j, \tau) + K_{conn}(n_j) \tag{4.18}$$

   where ϵ_{conn} models the relationship between network utilization and power, $K_{conn}(n_j)$ stands for a hardware related constant of the node and $U_{net}(t_i, n_j, \tau)$ represents the network utilization, calculated as:

[1] https://www.strchr.com/x86_machine_code_statistics

$$U_{net}(t_i, n_j, \tau) = \frac{DATA_{net}(t_i, \tau)}{bw(md, n_j)} \tag{4.19}$$

where $DATA_{net}(t_i, \tau)$ is the data transferred at time τ between the mobile device md and node n_j.

(3) Minimize User Costs $UC(\mathcal{D}(\mathcal{A}, \mathcal{I}))$

This objective minimizes the cost the user has to bear for offloading tasks to remote nodes and renting their computing power. De Maio et al. [26] proposed a cost model which considers where each task t_i is executed (locally, on the Edge or in the Cloud), the runtime of the t_i as well as the resource usage. The objective function is defined as:

$$UC(\mathcal{D}(\mathcal{A}, \mathcal{I})) = \sum_{t_i \in A} uc(t_i) \tag{4.20}$$

where $uc(t_i)$ is the cost for executing task t_i is based on its deployment location and calculated according to the following cases:

- *Executed locally on the mobile device md* no costs are calculated [26]:

$$uc(t_i) = 0 \quad \text{if } \phi(t_i) = md \tag{4.21}$$

- *Offloaded on a Cloud node*, the user is charged for the cloud resources during the runtime of t_i [26]:

$$uc(t_i) = p(\phi(t_i)) \cdot RT(t_i) \quad \text{if } \phi(t_i) \in \mathcal{C}_{\mathcal{I}} \tag{4.22}$$

- *Offloaded on an Edge node*, the user pays the price as it would be a Cloud node, plus an additional premium charge p_e for executing on an Edge node, which is defined in accordance to the lower latency and better user experience provided by Edge node [26]:

$$uc(t_i) = p(\phi(t_i)) \cdot RT(t_i) + p_e(\phi(t_i), \eta) \quad \text{if } \phi(t_i) \in \mathcal{E}_{\mathcal{I}} \tag{4.23}$$

where $p_e(\phi(t_i), \eta)$, defined by [26], is depended on the user preference parameter η, a value between 0.01 and 1. Lower values indicate a preference for lower latency, resulting in higher Edge usage costs and higher values prioritize cost savings, probably preferring Cloud over Edge nodes.

4.2 Algorithm Design

Having the deployment problem defined in section 4.1, the aim of this thesis is to propose an algorithm to find valid deployments by focusing on the objective goals on minimizing runtime and user costs as well as maximizing battery life. Since these three objectives are conflicting with each other, a multi-objective optimization algorithm with three objective functions is needed. As described in chapter 2.4, there are various different types for MOO algorithms. This thesis proposes a Particle Swarm Optimization algorithm, addressing the deployment problem with its discrete objective space. This section describes the algorithm and its approach on how particles are encoded and the movement within a discrete objective space is handled, which is based on the algorithm proposed by [75].

4.2.1 Leader Archive and Constraint Handling

As highlighted in section 2.5, the core principle of PSO is that each particle within a swarm represents a potential solution which iterativly moves through the objective space. Therefore, it is necessary to evaluate the objective functions of all particles in each iteration and store the best solutions in a leader archive $\mathcal{X}$. Best solution in this multi objective scenario means non-dominated solutions. Formally described, a solution x dominates another solution y as:

$$x \prec y \iff \forall i \in \{1, 2, \ldots, m\}, f_i(x) \leq f_i(y) \text{ and } \exists j \in \{1, \ldots, m\}, f_j(x) < f_j(y) \quad (4.24)$$

where m is the number of objectives and $f_i(x)$ represents the value of the i_{th} objective function for solution x.

At each point of time, $\mathcal{X}$ represents the current Pareto front approximation and can also be seen as the output of the algorithm, when the end condition is met. As described in 2.4.1, this approximation should solely contain Pareto optimal solutions, having no other dominating solution across all three objectives, which is formally described as:

$$\mathcal{X} = \{x \mid \nexists y : y \prec x\} \quad (4.25)$$

This means when the archive is updated, not only currently created non-dominated solution will be added, but also existing solutions will be checked against dominance of added ones and be possibly removed.

The goal of the algorthim is to find valid deployments. Since not each possible deployment solution automatically meet the criteria defined in 4.1.2, the deployment constraints also need to be evaluated in each iteration, beside calculating the objective functions. Therefore, a hybrid approach is proposed, which combines the penalty method and the constraint-dominance mechanism, as described in 2.4.2.

The evaluation of the solution constraint will not only differentiate between valid and invalid, but a degree of constraint violation will also be calculated. This enables a more fine granular punishment and allows to better differentiate between a small violation, such as the latency's failure of meeting the requirements for transferring a task, and more serious violations, such as proposing the execution of a not offload-able task, e.g. reading GPS data from the mobile device shown in Figure 3.2. The values for violation calculation $V(x)$ is based on proposed values within extended FogTorchPI by [26].

This degree of constraint violation is directly included in the dominance mechanism (denoted as $\prec_c$) while updating the archive. This check is done before comparing the objectives, so that solutions with a lower degree of constraint violation are always preferred over infeasible ones, meaning, additionally, that a valid deployments x having $V(x) = 0$ is always preferred over others which are infeasible of any degree. Only if both compared solutions x and y have the same degree of constraint violation, the standard dominance $\prec$, defined in Equation 4.25, applies. Formally described as:

$$x \prec_c y \iff \begin{cases} \text{If } V(x) = V(y): & x \prec y, \\ \text{Else:} & V(x) < V(y) \end{cases} \tag{4.26}$$

If there were already feasible deployments found, no invalid deployment will be added anymore. During the run of the algorithm, solutions within the *Archive* are always considered in the global best selection. As described in the next section, this approach allows to push the particles towards the (at least) less violated solutions.

Additionally, the algorithm uses a bounded archive, which limits the number of stored solutions in order to avoid overloading the memory (depending on where the algorithm is running) when many non-dominated solutions are found. As strategy on which solutions to prune, the crowding distance CD is used. As described in section 2.5.5, relying on this indicator and removing solutions with the worst crowding distance, maintain a better spread across the set and promote diverse solution set with respect to the objectives, visualized in Figure 2.6.

The overall mechanism of updating the archive $\mathcal{X}$ with a new swarm $\mathcal{S}$ in iteration is described in Algorithm 4.1.

4.2.2 Global Best Selection

As highlighted in section 2.5, one main principle of PSO is the knowledge sharing of best solutions across the swarm, represented by the solution component within the velocity calculation. Therefore, the selection of the global best position is required. While optimizing the deployment problem according to the three objectives, there is no single best solution, but rather a set of Pareto optimal solutions found along the way, which are tracked and can be accessed during the process within the archive $\mathcal{X}$.

Since all of the solutions within the archive are non-dominated, each of them could be a global best in its way. It would be valid to randomly select one solution and promote

Algorithm 4.1: Update Leader Archive

Input: swarm $\mathcal{S}$, archive $\mathcal{X}$ with $|\mathcal{X}| \leq archiveSize$
Output: updated archive $\mathcal{X}$

```
foreach p ∈ S do
    if ∄x ∈ X such that x ≺_c p then
        X ← X ∪ {p} \ {x ∈ X | p ≺_c x}
        if |X| > archiveSize then
            x_worst ← min_{x∈X} CD(x)
            X ← X \ x_worst
        end
    end
end
```

it as the global best. However, this algorithm follows the approach proposed by Qiao et al. [75], where global guiders for each objective $gbest_{RT}$, $gbest_{BL}$ and $gbest_{UC}$ are introduced. Each one is randomly picked from the top-L solutions within the Archive corresponding to the single-objective function.

Therefore, in each iteration the solutions within $\mathcal{X}$ will be sorted by the objective function $RT(x)$ and the top-L_{RT} solutions will be put into $\mathcal{X}_{RT}$. Then the top-L_{BL} solutions according to $BL(x)$ into $\mathcal{X}_{BL}$ and, last but not least, sorted by $UC(x)$ the top-L_{UC} solutions will be stored in $\mathcal{X}_{UC}$. In iteration t, each particle x_i^t randomly selects its $gbest_{RT,i}^t$, $gbest_{BL,i}^t$ and $gbest_{UC,i}^t$ from $\mathcal{X}_{RT}$, $\mathcal{X}_{BL}$ and $\mathcal{X}_{UC}$.

The parameterization of the upper bounds L_{RT}, L_{BL} and L_{UC} allows to limit or expand the search with regard to the diversity in the global best selection. The configuration of each of the L values, also enables expressing potential preferences for objectives. For example, restricting one objective, like $L_{BL} = 3$ - only allowing $gbest_{BL,i}^t$ to be picked from the 3 most promising solutions regarding battery life, potentially with bad runtime or high user costs, while setting the others $L_{RT} = L_{UC} = 20$ could probably lead to better trade-off selections.

4.2.3 Particle Encoding

Each particle within a swarm represents a potential solution which iterativly moves through the objective space. The particle can also be seen as a set of variables, used as input for the objective functions. Within the scenario of Mobile Edge Cloud Offloading used in this thesis, a particle represents a proposed deployment $\mathcal{D}(\mathcal{A}, \mathcal{I})$ of application $\mathcal{A}$ on infrastructure I - also seen as mapping of all tasks in $\mathcal{T}_a$ to the nodes in $\mathcal{N}_\mathcal{I}$.

Since the main criteria for complete deployment $\mathcal{D}(\mathcal{A}, \mathcal{I})$ is that each task in $\mathcal{T}_a$ must be deployed on exactly one computational node in $\mathcal{N}_\mathcal{I}$, but more tasks can be assigned to one computational node, the overall application workload of application $\mathcal{A}$ with N tasks

$t_1, \ldots, t_N : \forall t \in \mathcal{T}_a$ is fixed for the given problem and across all possible solutions, while the assignment of a node permutes for each solution.

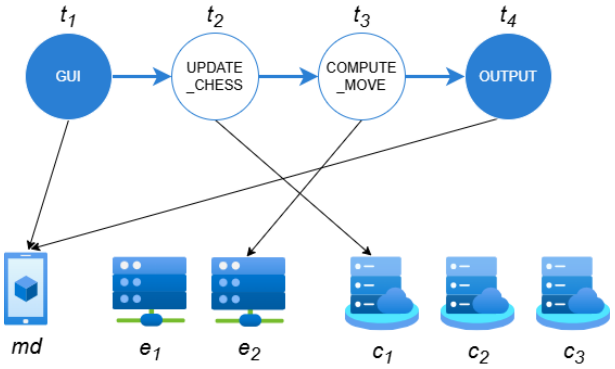


Figure 4.1: Potential solution for mapping chess application [26] to a given infrastructure

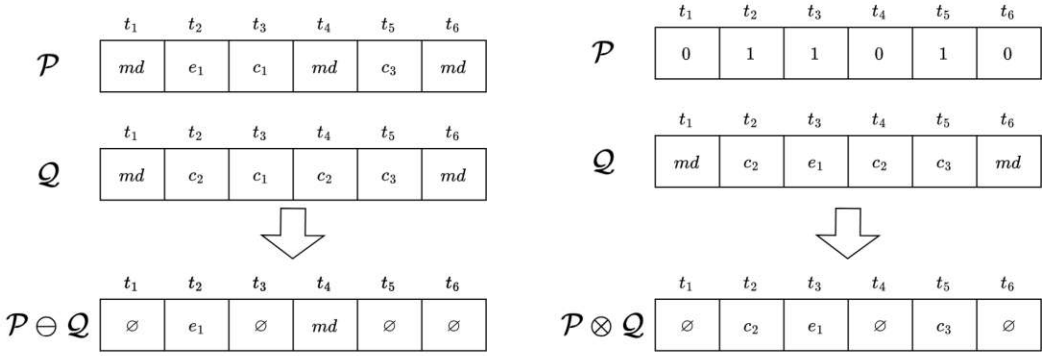
Therefore, a deployment can also be seen as N-dimensional vector $\overrightarrow{x}$, where each index i corresponds to a task $t_i \in \mathcal{A}$ and the value on index i stands for $\phi(t_i)$, the computational node $n_j \in \mathcal{N}_\mathcal{I}$ where t_i is deployed. For example, the vector $\overrightarrow{x} = [md, c1, e2, md]$ is a potential solution by given $N = 4$ (application workload of 4 tasks) and $\mathcal{I} = \{\{c1, c2, c3\}, \{e1, e2\}, md\}$ (infrastructure with 3 Cloud nodes, 2 Edge nodes and a mobile device) visualized in Figure 4.1.

4.2.4 Velocity Calculation and Position Update

Classic PSO algorithms are commonly designed to solve problems and their objective functions within the real number space, making it possible to seamlessly navigate with its trajectory through the search space. In this scenarios, the velocity can be easily calculated like described in 2.5.2 and Equation 2.3, by substracting the current position from the global and personal best, multiplying with random factors and finally creating the sum of all parts.

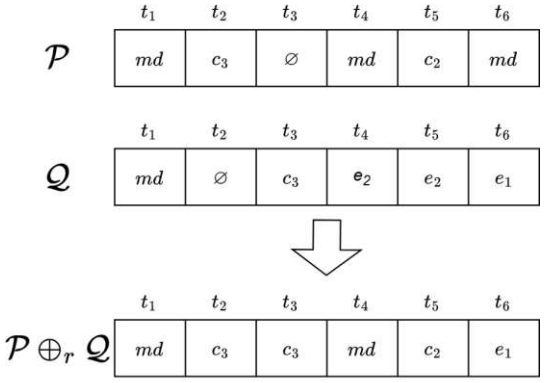
Since each particle in this scenario represents a discrete assignment of tasks to computational nodes, particles cannot smoothly navigate through the search space but rather permute their mappings. This makes it necessary to design new operators replacing addition, subtraction and multiplication, which are used in the velocity calculation of classic PSO algorithms, seen in Equation 2.3. Qiao et al. [75] proposed such operators for updating particle positions within a discrete space, subject to the same ranking and order of operations as the conventional counterpart, as as following:

Subtraction Operator $\ominus$: This operator is designed to extract differences of the left particle from the right one. Within the velocity calculation it is used to reserve unique information from *gbest* compared with the current position x. For given particles $\mathcal{P}[p_1, \ldots, p_N]$ and $\mathcal{Q}[q_1, \ldots, q_N]$, if $p_i \neq q_i$, the i_{th} index of $\mathcal{P} \ominus \mathcal{Q}$ is set to p_i, otherwise it is set to $\emptyset$. An example is visualized in Figure 4.2a.



(a) The $\ominus$ operator

(b) The $\otimes$ operator



(c) The $\oplus_r$ operator

Figure 4.2: Visualization of discrete particle operators based on [75]

Multiplication Operator $\otimes$: This operator takes a N-dimensional binary vector, with each bit contains 0 or 1, on the left side and masking the right particle with its pattern. It is used to randomly reserve parts of the *gbest* for the velocity. The behavior shown in Figure 4.2b, using a binary sequence $\mathcal{P}$ of N bits and a particle $\mathcal{Q}[q_1, \ldots, q_N]$, the i_{th} position of $\mathcal{P} \otimes \mathcal{Q}$ is set to q_i, if $p_i = 1$, otherwise it is set to $\emptyset$.

Addition Operator $\oplus_r$: This operator is used to integrate the information of two particles into one. It is applied to get one velocity vector out of its three parts as well as to operate the velocity on the current position. Given particles $\mathcal{P}[p_1, \ldots, p_N]$ and $\mathcal{Q}[q_1, \ldots, q_N]$, if $p_i = q_i$ then the i_{th} index of $\mathcal{P} \oplus_r \mathcal{Q}$ is set to the same value, else, if p_i or q_i is $\emptyset$ then the not empty one is selected. If both p_i and q_i contain values, but different ones, with a probability of r the value of p_i is picked otherwise q_i. Instead of complete random selection, this probability factor r allows to control the probability distribution when performing this operator multiple times in succession, e.g. $\mathcal{P} \oplus \mathcal{Q} \oplus \mathcal{X}$, or to eventually express preferences in which particle to prefer.

Having those operators defined, Qiao et al. [75] also suggest that such a discrete search space based on permutation has neither a current direction nor a previous path that needs to be kept, since each particle can *fly* towards any guider. Therefore, it is not necessary to keep the personal information within the calculation of the new velocity, represented by the inertia term and cognitive component as described in 2.5.2. This means that in each iteration, the new velocity v_i^{t+1} thus also the new position x_i^{t+1} is only influenced by the global bests $gbest_{RT,i}^t$, $gbest_{BL,i}^t$ and $gbest_{UC,i}^t$. The velocity calculation for the proposed algorithm is described as:

$$\overrightarrow{v_i^{t+1}} = \overrightarrow{c_{RT}} \otimes (\overrightarrow{gbest_{RT,i}^t} \ominus \overrightarrow{x_i^t}) \oplus \overrightarrow{c_{BL}} \otimes (\overrightarrow{gbest_{BL,i}^t} \ominus \overrightarrow{x_i^t}) \oplus \overrightarrow{c_{UC}} \otimes (\overrightarrow{gbest_{UC,i}^t} \ominus \overrightarrow{x_i^t}) \quad (4.27)$$

where $\overrightarrow{c_{RT}}$, $\overrightarrow{c_{BL}}$ and $\overrightarrow{c_{UC}}$ are three N-dimensional binary vectors, where each index containing 0 or 1. Using this velocity v_i^{t+1}, the new position x_i^{t+1} will be calculated as:

$$\overrightarrow{x_i^{t+1}} = \overrightarrow{x_i^t} \oplus \overrightarrow{v_i^{t+1}} \quad (4.28)$$

4.2.5 Use of Perturbation

In addition to the mechanism for calculating the new position of a particle described above, this algorithm also makes use of the principle of perturbation. As described in section 2.5.3, turbulences in PSO are used to promote diversity and can prevent premature convergence by allowing particles to randomly jump to new areas of the search space with a given probability. The goal is that in each iteration, some selected particles of the swarm with a given probability p_{mut} get caught up in turbulence after they already moved to the new position.

Since the algorithm is designed to solve the discrete deployment problem, where solutions are searched by permutation, the use of mutation-inspired turbulences based on the principles of genetic algorithms [76] fits perfectly. Therefore, a permutation based mutation is used, where the mapping of two tasks to their computational nodes are swapped. So, for each particle which randomly meet the probability p_{mut}, two tasks t_i and t_j in T_a where $t_i \neq t_j$ are randomly selected and their mappings to the nodes $n_i = \phi(t_i)$ and $n_j = \phi(t_j)$ in N_I are swapped, so that after the mutation operation $\phi(t_i) = n_j$ and $\phi(t_j) = n_i$.

Additionally, to prevent a violation regarding the offloading constraint of Equation 4.6, the mutation operator also makes sure that both tasks t_i and t_j are offload $OFF(t_i) = OFF(t_j) = 1$ before swapping. The algorithm for the deployment mutation is described in 4.2.

4.2.6 Algorithm

The proposed MDPSO algorithm for the deployment problem, which is shown in Algorithm 4.3, is made up of two stages which use the mechanisms described above.

Algorithm 4.2: Deployment Mutation

Input: swarm $\mathcal{S}$, p_{mut}
Output: swarm $\mathcal{S}$

```
1  foreach 𝒟 ∈ 𝒮 do
2      k ← random number from [0,1]
3      if k < p_mut then
4          do
5              t_i ← Random task in 𝒟
6              do
7                  t_j ← Random task in 𝒟
8              while t_i ≠ t_j;
9              isOffloadable ← OFF(t_i) = 1 ∧ OFF(t_j) = 1
10         while !isOffloadable;
11         n_i ← φ(t_i)
12         n_j ← φ(t_j)
13         𝒟 ← (𝒟 \ {(t_i, n_i), (t_j, n_j)}) ∪ {(t_i, n_j), (t_j, n_i)}
14     end
15 end
```

In the initialization phase, the swarm $\mathcal{S}$ with given *swarmSize* will be populated with random solutions. After each particle of $\mathcal{S}$ is evaluated against the objective functions RT, BL and UC as well as constraint violations V, they are added to the leaders archive $\mathcal{X}$ of size *archiveSize* by the mechanism described in Algorithm 4.1.

After initializing swarm $\mathcal{S}$ and archive $\mathcal{X}$, the evolution phase iteratively moves the particles through the search space until the stopping condition of *maxIterations* is reached. In each iteration, all particles select a global best for the three objectives as described in 4.2.2 and randomly generate the binary vectors c_{RT}, c_{BL} and c_{UC}. By using them, the velocity and new position according to the equations 4.27 and 4.28 are computed. The swarm $\mathcal{S}$ will be evaluated against the objective functions and violation of constraints after a mutation on particles in $\mathcal{S}$ with a chance of p_{mut}, as explained in Algorithm 4.2, has been carried out. Afterwards, the archive $\mathcal{X}$ is updated with the evaluated swarm.

Algorithm 4.3: Proposed MDPSO Algorithm for Deployment Problem

Data: $maxIterations$,$swarmSize$,$archiveSize$,L_{RT},L_{BL},L_{UC},p_{mut}
Result: archive $\mathcal{X}$ as Pareto front approximation

Initialize the swarm $\mathcal{S}$ of size $swarmSize$ with random solutions
Evaluate $\mathcal{S}$ by objectives (RT, BL, UC) and constraint violations V
Initialize $\mathcal{X}$ of size $archiveSize$ with $\mathcal{S}$ according to Alg. 4.1
$t \leftarrow 1$
while $t \leq maxIterations$ **do**
    foreach $obj \in \{RT, BL, UC\}$ **do**
        $\mathcal{X}_{obj} \leftarrow$ top-L_{obj} solutions from $\mathcal{X}$ sorted by objective obj
    end
    for $i \leftarrow 1$ **to** $|\mathcal{S}|$ **do**
        foreach $obj \in \{RT, BL, UC\}$ **do**
            $k_{obj} \leftarrow$ random integer from [1,L_{obj}]
            $gbest^{t}_{obj,i} \leftarrow$ the k_{obj}th member in $\mathcal{X}_{obj}$
            $c_{obj} \leftarrow$ N-dimensional vectors, each index randomly set to 0 or 1
        end
        Move to new position x_i^{t+1} according to Eqs. 4.27 and 4.28
    end
    Operate turbulence on $\mathcal{S}$ by probability p_{mut} according to Alg. 4.2
    Evaluate $\mathcal{S}$ by objectives (RT, BL, UC) and constraint violations V
    Update $\mathcal{X}$ with $\mathcal{S}$ according to Alg. 4.1
    $t \leftarrow t + 1$
end

CHAPTER 5

Evaluation

This chapter deals with the evaluation of the proposed Multi-Objective Discrete Particle Swarm Optimization (MDPSO) algorithm in the context of mobile cloud offloading compared to other state-of-the-art Multi-Objective Evolutionary Algorithms. Since the MDPSO and all compared evolutionary algorithms rely on randomness and are influenced by uncertainty of their operations, a deterministic evaluation using fixed inputs does not provide sufficient insights on their performance.

The experimental approach utilizes Monte Carlo simulations, which provide a methodology for analyzing those algorithms by performing a large number of simulation runs across diverse problem scenarios. Various quality indicators are then used in the evaluation of the results as well as statistical analysis of the resulting indicators.

The procedure of the simulating experiment is done as following:

1. *Preperation of Problems:* Different application workloads and infrastructure configurations with various sizes are randomly generated (detailed in Section 5.1.1).
2. *Parameterization of Algorithms:* In order to attempt to enable a fair comparison (detailed in Section 5.1.2), each algorithm is configured with fixed parameters, including population size, iteration count, crossover and mutation rates - trying to enable a fair comparison.
3. *Executions of Experiments:* Each algorithm was executed 1000 independent times per problem instance to ensure statistically reliable results and variability.
4. *Reference Pareto Front Generation:* Since for the randomly generated problem instances no true Pareto sets are known, reference Pareto fronts are constructed by aggregating all non-dominated solutions which are generated by all algorithms across all runs for a given problem instance.

5. *Quality Indicators Calculation:* Several quality indicators are computed for each run, by evaluating solution proximity, diversity and computational efficiency. Most of them are calculated against the reference Pareto front.

6. *Statistical Tests:* The results and quality indicators are evaluated to identify significant differences across the algorithms.

5.1 Experimental setup

To compare the performance of algorithms with each other, it is necessary to create a test setup, which provides similar conditions for the different algorithms [9]. This section outlines the characteristics of the selected problems, the configured parameter settings and the computational resources of the used test environment.

5.1.1 Used models of Problems

As described in section 3.3.2, the extended version of FogTorchPI by [26] was chosen as the edge simulation framework. It is used to simulate different real-world scenarios for MECO by providing an infrastructure $\mathcal{I}$ as well as the application workload $\mathcal{A}$, which are then used by the algorithms to search for valid deployments $\mathcal{D}(\mathcal{A}, \mathcal{I})$.

The base configuration for the simulated hardware specifications of computational nodes $\mathcal{N}_\mathcal{I}$ (like cores, RAM, storage, costs, etc.) as well as the parameters for energy coefficients (like β_{freq}, ϵ_{conn} and K_{conn}) used in the equations for *Processing Power Consumption* $P_p(md, \tau)$ 4.16 and *Offloading Power Consumption* $P_{off}(t_i, n_j, \tau)$ 4.18 are set to the same values as used in [26]. Also, the network availability distribution configuration is set to the same probabilities as in [26].

Even though the framework provides enhanced capabilities for different randomly created conditions for infrastructure and application workload for each evaluation run by using these pre-configured settings and probabilities, for this kind of performance evaluation of a new proposed algorithm against reference algorithms, it is necessary to provide the same conditions across the algorithm runs [9]. Meaning that the 4 different reference algorithms used in the evaluation should all be able to solve the same problem, with the same conditions in order to generate a comparable result. Otherwise, the resulting Pareto front approximations found in these runs could be significantly better or worse not due to the algorithm performance itself, but due to possible better network conditions. Thus, this could lead to a direct impact on the objectives.

In order to provide a broader coverage in the performance evaluation, 20 different deployment scenarios of various complexity were randomly created. Each consists of 5 application workloads and 4 different infrastructures. The setup of the 4 infrastructures with different numbers of nodes, are randomly generated by using the configuration and probabilities described above, which can be found in Table 5.1. For the application workload, random sequences of DAG executions, representing the applications NAVI,

Table 5.1: Randomly generated infrastructure setups used for evaluation

	$\lvert\mathcal{C}_\mathcal{I}\rvert$	$\lvert\mathcal{E}_\mathcal{I}\rvert$	md	$\lvert\mathcal{L}_\mathcal{I}\rvert$
$\mathcal{I}_1$	5	13	1	55
$\mathcal{I}_2$	10	13	1	70
$\mathcal{I}_3$	15	13	1	85
$\mathcal{I}_4$	20	13	1	100

FACEREC, CHESS and ANTIVIRUS described in 3.3.2, were assembled. This results in the workloads $\mathcal{A}_1, \ldots, \mathcal{A}_5$ of increasing size, which are shown in Table 5.2.

Table 5.2: Randomly generated application workloads used for evaluation

	Used applications	$\lvert\mathcal{T}_a\rvert$	$\lvert\mathcal{L}_a\rvert$
$\mathcal{A}_1$	1x NAVI	9	13
$\mathcal{A}_2$	1x NAVI, 1x FACEREC, 1x CHESS, 1x ANTIVIRUS	23	29
$\mathcal{A}_3$	2x NAVI, 2x FACEREC, 3x CHESS	40	51
$\mathcal{A}_4$	2x NAVI, 3x FACEREC, 2x CHESS, 1x ANTIVIRUS	46	59
$\mathcal{A}_5$	3x NAVI, 3x FACEREC, 4x CHESS, 2x ANTIVIRUS	68	87

5.1.2 Algorithm Configuration

To ensure a fair comparison across the algorithms and consistency with established benchmarks, the parameter settings were chosen based on default values which are widely accepted in literature comparing MOO algorithms, as in [20] and [7]. Table 5.3 provides an overview of the parameter configuration for all algorithms used in the evaluation, including common parameters, such as population size and maximum evaluations, as well as algorithm-specific parameters, such as crossover probability or archive size.

Table 5.3: Parameter of algorithms used in simulations

Parameter	MDPSO	NSGAII	NSGAIII	MOCell	SPEA2
$populationSize$	100	100	100	100	100
$archiveSize$	100	-	-	100	-
$maxIterations$	100	100	100	100	100
p_{mut}	$\frac{1}{\lvert\mathcal{T}_\mathcal{A}\rvert}$	$\frac{1}{\lvert\mathcal{T}_\mathcal{A}\rvert}$	$\frac{1}{\lvert\mathcal{T}_\mathcal{A}\rvert}$	$\frac{1}{\lvert\mathcal{T}_\mathcal{A}\rvert}$	$\frac{1}{\lvert\mathcal{T}_\mathcal{A}\rvert}$
p_{cross}	-	0.9	0.9	0.9	0.9
L_{RT}, L_{BL}, L_{UC}	25	-	-	-	-

The population size (corresponds to the swarm size in MDPSO) was set to 100, providing sufficient diversity while balancing computational cost [7]. Additionally, the size of the archives used in the proposed MDPSO and MOCell was set to 100, as used in [20], ensuring a robust representation of the Pareto front during optimization. A larger archive might retain redundant solutions while a smaller one could fail to capture the necessary diversity of the front. All algorithms were allowed to run 100 iterations before terminating and corresponded to 10.000 objective function evaluations in one execution. By giving a minimum complexity from application workload $\mathcal{A}_1$ with the infrastructure

$\mathcal{I}_1$ of $19^9 = 13,841,287,201$ possible permutations, this limit was selected to ensure a comprehensive exploration of the search space without approaching an exhaustive search scenario. The mutation probability was configured to depend on the problem size, set to $\frac{1}{|\mathcal{T}_\mathcal{A}|}$, where $|\mathcal{T}_\mathcal{A}|$ is the number of applications tasks $\mathcal{T}_\mathcal{A}$ within the workload $\mathcal{A}$, respectively, the number of decision variables. The crossover probability used by NSGAII, NSGAIII, MOCell and SPEA2 is set to 0.9. The MDPSO specific upper bound parameters L_{RT}, L_{BL} and L_{UC}, used in the global best selection process (see section 4.2.2 for more details), were uniformly set to 25. This value was chosen to provide the same level of restrictiveness across all parameters and to ensure that the algorithm's global best selection process is neither set too high (which could lead to poor convergence) nor too restrictive (which might hinder exploration). Since all algorithms are based on probabilities, leading to different results in every run, each algorithm was executed 1000 times for each problem statement to provide a better statistical significance and confidence of the results.

5.2 Result Comparison

In multi-objective optimization, there is no single optimal solution as result, which can be easily compared. Instead, algorithms produce sets of trade-off solutions aiming to approximate the true Pareto front of a given problem. This section describes how the results of the different algorithms are compared by the use of various quality indicators even when no true Pareto fronts for the problems are known and how they are statistically checked on significance.

5.2.1 Reference Pareto Front

To compare the performance of multi-objective optimization algorithm with each other, it is necessary not only to provide equal conditions for the execution, but also to have a shared reference to which the results are compared to [16]. Therefore, some standardized benchmark problems and test suites were proposed in literature where a true Pareto front is known, as the ZDT or DTLZ [30]. Meaning that the overall best solutions possible for the problem are known before and serve as the benchmark for evaluating the performance.

In real-world problems, such as the deployment problem in the scenario of mobile offloading used in this thesis, where randomly generated problem instances are used, there are no true Pareto sets known. Instead, a reference Pareto front R is constructed by aggregating all solutions S_i which are generated by all independet runs n across all algorithms for a given problem instance to a set S. Then, all non-dominated solutions are extracted [16], which are formally defined as:

$$S = \bigcup_{i=1}^{n} S_i \tag{5.1}$$

$$R = \{x \in S \mid \nexists y \in S,\ y \prec x\} \tag{5.2}$$

Technically, this reference front is constructed by using the same mechanism as Algorithm 4.1, which adds solutions one by one to the front. Each time the dominance is checked and it is made sure that only non-dominated solutions remain. Therefore, the reference front R can be seen as archive $\mathcal{X}_R$ with $archiveSize = \infty$, to not prune some non-dominated solutions regarding size constraints. This allows to conduct a comparative analysis of the Pareto approximations which are generated by the algorithms relative to a common baseline.

5.2.2 Quality Indicators

Quality indicators are mathematical tools that measure different aspects of the Pareto front approximations A in order to provide insights on how well the algorithm performs. They can be expressed as functions $I(A, R) \to \mathbb{R}$ which assigns a real number for each Pareto set approximation A for a given reference front R. These numerical values enable comparisons between algorithms, even in the absence of the true Pareto front. This subsections describe each used quality indicator in detail:

Unary Epsilon Additive Indicator ($I_{\epsilon+}$)

The ϵ-indicator was introduced by Eckart Zitzler et al. [99] for comparing the quality of multi-objective optimization result sets. It measures how much a Pareto front approximations A needs to be additively shifted to dominate the reference front R [99], which is defined as the following:

$$I_{\epsilon+}(A, R) = \inf_{\epsilon \in \mathbb{R}} \{\forall r \in R, \exists a \in A : a \preceq_{\epsilon+} r\} \tag{5.3}$$

where the ϵ-dominance relation, $\preceq_{\epsilon+}$, is defines as:

$$a \preceq_{\epsilon+} r \iff \forall i \in \{1, \ldots, n\} : f_i(a) \leq \epsilon + f_i(r) \tag{5.4}$$

A smaller $I_{\epsilon+}(A, R)$ indicates a better convergence which means that the solutions in A are closer to the solutions in the reference front R.

Objective: Needs to be **minimized**

General Distance (I_{GD})

This indicator measures the proximity of the solutions in the Pareto front approximation A to the reference Pareto front R [10]. It was originally introduced by Van Veldhuizen et al. [87] and can be defined as:

$$I_{GD}(A, R) = \sqrt{\frac{1}{|A|} \sum_{a \in A} d(a, R)^2} \tag{5.5}$$

where $d(a, R)$ is the minimum Euclidean distance between a solution $a \in A$ and any point in R [10].

The indicator focuses on convergence, where a smaller I_{GD} value indicates that on average the solutions in A are closer to the reference Pareto front R. However, it reveals nothing about the diversity or coverage of the entire reference front [10].

Objective: Needs to be **minimized**

Inverted General Distance (I_{IGD})

The Inverted Generational Distance (IGD) is conceptually similar to the General Distance (GD), but they measure different aspects of how an approximated Pareto front relates to the reference Pareto front. As the GD lacks in significance for diversity or coverage, the IGD was proposed by [21] and measures the coverage of the reference Pareto front R by the solutions in the Pareto front approximation A, by basically swapping the use of R and A within the equation. Then, it measures the average distance from each point on the reference Pareto front R to its closest point in the approximation set A [10], which is described by:

$$I_{IGD}(A, R) = \sqrt{\frac{1}{|R|} \sum_{r \in R} d(r, A)^2} \tag{5.6}$$

where $d(a, R)$ calculates the Euclidean distance from $r \in R$ to the nearest point in A [10].

A smaller I_{IGD} value indicates that the approximation A covers the solutions in the reference Pareto front R better by penalizing missing regions of the Pareto front. However, the I_{IGD} does not consider the dominance, which means that whether the solutions in A dominate or are dominated by the reference front R is not distinguished [10].

Objective: Needs to be **minimized**

Inverted Generational Distance Plus (I_{IGD+})

Proposed by [53], the IGD+ enhances the IGD by adding domination sensitivity, which penalizes solutions of the approximation A. These, in turn, are dominated by the reference front R and defined as:

$$I_{IGD+}(A, R) = \frac{1}{|R|} \sum_{r \in R} d^+(r, A) \tag{5.7}$$

where $d^+(r, A)$ is the perpendicular distance from $r \in R$ to the closest dominating or equal solution in A [10].

A smaller I_{IGD+} value indicates not only better coverage of R by A, but also that more solutions in A are either equal or better than R in terms of domination. This makes it more robust for evaluating algorithms in real-world optimization scenarios where solution dominance plays a critical role as quality measure [10]

Objective: Needs to be **minimized**

Generalized SPREAD (I_Δ)

The original SPREAD indicator was introduced as diversity metric by Deb et al. [29], in order to measure the extant distribution which is achieved by of approximated solutions A along the reference front R. Since this approach is limited for a 2-objective problem, Zhou et al. [96] proposes an extension to handle higher-dimensional optimizations, known as the Generalized SPREAD, defined as:

$$I_\Delta(A, R) = \frac{\sum_{i=1}^{m} d(e_i, R) + \sum_{a \in A} |d(a, R) - \bar{d}|}{\sum_{i=1}^{m} d(e_i, R) + |A|\bar{d}} \tag{5.8}$$

where $\{e_1, \ldots, e_m\}$ are m extreme solutions in A and $d(a, R)$ is the minimum Euclidean distance between a solution $a \in A$ and any point in R. $\bar{d}$ represents the average distance of all solutions in A to their nearest neighbors in R as:

$$\bar{d} = \frac{1}{|A|} \sum_{a \in A} d(a, R) \tag{5.9}$$

A smaller I_Δ indicates a more even distribution of solutions along the reference front, whereas a larger one reveals clustering of solutions [96].

Objective: Needs to be **minimized**

Hypervolume (I_{HV})

The hypervolume indicator, proposed by Zitzler et al. [98], has become one of the most common indicator to evaluate the performance of multi-objective optimization algorithms [97]. It is used for evaluating convergence and diversity by measuring the volume of the objective space which is dominated by the solutions of the approximated front A against a reference point r. This reference point is usually chosen as the nadir point of the referencing Pareto front R [97], which is defined as the vector with the worst values of each objective [16]. This vector is important as it is a reference point which is the same for all algorithm runs compared.

By calculating the volume for each solution a in A relative to the given reference point r, the HV is computed by building the union of all these volumes (ensuring no double-counting of overlapping regions), formally described as:

$$I_{HV}(A, r) = \text{Volume}\left(\bigcup_{a \in A} \mathcal{H}(a, r)\right) \tag{5.10}$$

where $\mathcal{H}(a, r)$ is the volume of the region dominated by solution a to r. This volume calculation depends on the number of objectives and makes it increasingly complex to compute for higher dimensions [16]. For example, in bi-objective optimization, this is

the volume of a rectangle and for an optimization with three objectives, the region is represented as cuboid, formally described as:

$$\mathcal{H}(a, r) = \prod_{k=1}^{m} (r_k - a_k) \tag{5.11}$$

where m is the number of objectives $r_k - a_k$ and represents the length of the hyper-rectangle along objective k, which are multiplied with every lengths across all objectives.

A larger I_{HV} indicates a better approximation of the reference Pareto front with regard to both convergence and diversity [97].

Objective: Needs to be **maximized**

Error Ratio (I_{ER})

Proposed by [87], this indicator measures the proportion of solutions in an approximation set A which are not included in the reference front R and defined as:

$$I_{ER}(A, R) = \frac{|A \setminus R|}{|A|} \tag{5.12}$$

A smaller I_{ER} indicates a better accuracy of an algorithm in identifying Pareto-optimal solutions [87].

Objective: Needs to be **minimized**

Computing Time (I_T)

The previously described quality indicators only consider the given result compared to the collected reference Pareto front, but in a real life scenario - especially including mobile devices with less computation power than used in this experimental setup - also the time of the algorithm execution is an important factor. This indicator will also be calculated for each run by measuring the timestamp before the algorithm execution will be subtracted from the timestamp afterwards, which can be formally defined as:

$$I_T(Alg, P_{\mathcal{A},\mathcal{I}}) = t_{end} - t_{start} \tag{5.13}$$

where t_{start} and t_{end} are the timestamps when the execution of algorithm Alg on problem $P_{\mathcal{A},\mathcal{I}}$ begins and ends.

The value will be stored in the a separate file "TIMEn.tsv" where n is the number for each independent run.

Objective: Needs to be **minimized**

5.2.3 Statistical Evaluation and Indicators Comparison

While the quality indicators, described in the previous section, provide comparable values, indicating which algorithm performs better or worse in various aspects, these insightscould only be observed by a random chance. In order to provide a confident basis for interpreting these results, following statistical tests are applied on the quality indicator values, which validate the statistical significance of the differences across algorithms.

Wilcoxon Test

The Wilcoxon signed rank test checks whether two samples differ significantly from each other. It tests the null hypothesis (H_0) whether the median difference is zero or not [65]. Additionally, the test is a non-parametric statistical test, which means that it does not rely on assumptions of the underlying data distribution [65].

In this context it is used to check the significant difference of the quality indicators of the proposed MDPSO algorithm against the other algorithms on every problem instance independently. By comparing the observations $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$ where $n = 1000$ executions of the algorithm for the problem instance, the difference $d_i = x_i - y_i$ of each pair (x_i, yi) is calculated. By defining $r(x)$ as the rank of an observation and $I(\rho)$ as indicator function returning 1 if ρ is true, otherwise false, the Wilcoxon statistic W is calculated as[83]:

$$W = \sum_{i=1}^{n} r(|d_i|) I(d_i > 0) \tag{5.14}$$

Since $n > 25$ the critical value Z needed to compared for H_0 is calculated as following [83]:

$$Z = \frac{W - \frac{n(n+1)}{4}}{\sqrt{\frac{n(n+1)(2n+1)}{24}}} \tag{5.15}$$

If this critical value Z is smaller than the significance level $\alpha = 0.05$, the null hypothesis is rejected and interpreted that the median is statistically significant different between the two samples [83].

The results of all pairwise tests comparing MDPSO for each indicator and problem instance against every other algorithm are presented in Table 5.5. The symbol "▲" is used when the median indicator value of this problem instance of MDPSO is confidentially better than the one from the algorithm in the column, "▽", which indicates that the median of the compared algorithm is statistically better. Additionally, "–" means that there is no statistical significance between the compared algorithms.

Friedman Test

While the Wilcoxon test just compares two samples with each other, the Friedman test proposed by [39] is designed to detect differences in performance across multiple samples (algorithms) when the same subjects (problem instances) are used. This test is also non-parametric [65]. In the context of quality indicators for multi-objective optimization, the objective of the Friedman test is to check whether all algorithms perform equally well for the given quality indicator (the null hypothesis H_0 [39]) or not. This is done by, firstly, calculating first a ranking of the algorithms R_j based on the mean performance of the indicator across all problems. Secondly, the Friedmann test statistic χ^2 is conducted and compared with the critical value $\chi^2_{critical}$. For a given number of k algorithms and n problems it can be done as following [39]:

$$R_j = \frac{1}{n} \sum_{i=1}^{n} R_{ij} \tag{5.16}$$

where R_{ij} is the rank of the j_{th} algorithm for the i_{th} problem instance based on the mean value of the compared quality indicator. Based on this ranking the $\chi^2_{observed}$ is calculated as following [39]:

$$\chi^2 = \frac{12n}{k(k+1)} \left(\sum_{j=1}^{k} R_j^2 - \frac{k(k+1)^2}{4} \right) \tag{5.17}$$

The critical value $\chi^2_{critical}$ can be obtained from pre-calculated χ^2 distribution tables (like in [72]), by a wanted significance level α and a degree of freedom $df = k - 1$. For this evaluation for a given $df = 4$ and a choosen $\alpha = 0.05$ the $\chi^2_{critical}$ is 9.488. Meaning that if the observed value $\chi^2_{observed}$ for a quality indicator is > 9.488, the null hypothesis (H_0) will be rejected. This can be interpreted as, yes there is a significant difference performance of the algorithms.

Nemenyi Test

As described above, the Friedman test calculates a ranking between the mean values for each algorithm across each problem instance, but does not check statistically significance of the ranking itself. That is where the post-hoc Nemenyi test is used on top, which helps to substantiate the significance of this ranking [86]. In other words, this test takes the Friedman ranking and checks, if an algorithm A_1 ranked above A_2 is confidently better. This is conducted by checking the difference of the ranking against a Critical Distance (CD) which is calculated by a given number of k algorithms and n problems as following [86]:

$$CD = q_\alpha \cdot \sqrt{\frac{k(k+1)}{6n}} \tag{5.18}$$

The CD in this evaluation by given $k = 5$, $n = 20$ and a critical value $q_\alpha = 2.569$ (derived from Table 1 of [86] for $\alpha = 0.05$ and $k = 4$) is approximately 1.285. This means that the ranking of two algorithms is statistically significant at the $\alpha = 0.05$ level, if the difference between the ranks is greater then 1.285.

5.3 Evaluation Results and Discussion

This chapter presents a detailed analysis of the evaluation results of the proposed MDPSO algorithm, compared to the reference algorithms NSGA-II, NSGA-III, MOCell and SPEA2, for the 20 problem instances for MECO deployments. The discussion focuses on the quality indicator described in the previous section. Since every indicator provides unique insights into algorithm performance, the discussion is clustered in the following aspects:

- **Convergence:** How close are the solutions to the reference Pareto front?
- **Diversity:** How well-distributed are the solutions across the front?
- **Trade-off Analysis:** How effectively does the algorithm balance convergence and diversity?
- **Accuracy:** How many solutions in the approximation set are contained in the reference front?
- **Efficiency:** How much computational time is required to achieve the results?

By calculating all described indicators for each independent execution (1000x) of each algorithm (5x) for each problem instance (20x), this study ensures a comprehensive evaluation of the algorithm performance. The tables x to y each show the mean and standard deviation for the representing quality indicator across all independent runs. Table 5.4 contains the Friedman test results, showing the overall ranking of the algorithms for each indicator with its calculation described in section 5.2.3. In those tables, the dark and light gray column highlights the best and second best algorithm.

5.3.1 Convergence

The convergence analysis evaluates the proximity of the solutions generated by optimization algorithms to the reference Pareto front, which represents the ideal set of trade-offs for the given problem. This aspect is measured by the following quality indicators, where lower values indicate better convergence, as described in detail in section 5.2.2:

- $I_{\epsilon+}$ **(Unary Epsilon Additive Indicator):** Measures the minimum shift which is necessary for letting each point in the obtained Pareto front dominate the reference front.

- I_{GD} **(Generational Distance):** Calculates the average Euclidean distance between the solutions in the Pareto front approximation and the nearest solutions in the reference Pareto front.
- I_{IGD} **(Inverted Generational Distance):** Measures the distance from each point of the reference Pareto front to the nearest point in the resulting Pareto front approximation.
- I_{IGD+} **(Inverted Generational Distance Plus):** An extension of I_{IGD} that penalizes solutions far from the reference front, focusing more on convergence than distribution.

The visualized comparison represented by collections of boxplots for each problem instance can be found in the figures 5.1 to 5.4. The raw value comparison for all problem instances, highlighting the winner and second best, of mean and median can be found in tables A.1 to A.4.

By comparing raw values of mean and median shown in the tables A.1, A.3 and A.4, MDPSO outperforms the other algorithms in terms of $I_{\epsilon+}$, I_{IGD} and I_{IGD+} across all problem instances starting with $\mathcal{A}_2$ where the best algorithms are consistently ranked. Even though the mean of I_{IGD} and I_{IGD+} are close in terms of the median, the distance is greater as shown in boxplot collections of figure 5.3 and 5.4. The $I_{\epsilon+}$ results for MDPSO on the other hand are clearlier separated compared to the others, as visualized in the boxplots of figure 5.1.

Additionally, the Wilcoxon test with its pairwise comparison, shown in table 5.5, significantly indicates the dominance of MDPSO according to the quality indicators. Its good performance regarding the convergence can be attributed to its velocity update mechanism and global best selection, which supports the guidance of particles moving closer to the true Pareto front. NSGA-II follows closely by performing second best across the majority of problem instances, also benefiting from its diversity-preserving mechanisms.

At the same time, for the problem instances having the least complex application workload $\mathcal{A}_1$ to be deployed, SPEA2 achieves the best results while MDPSO is dominated by all other algorithms except MOCell. All results are statistically confident as proved by the Wilcoxon test. As defined in section 5.1.1, $\mathcal{A}_1$ solely includes the workload of the NAVI application having 9 tasks to map and 13 links to verify. This indicates that MDPSO performs better in terms of convergence with particle vectors of higher dimension.

Furthermore, when checking the I_{GD} in table A.2, the picture drastically changes. SPEA2 leads across all problems while MDPSO is second best in 13 of 20 problem instances, especially with the more complex ones. Even though SPEA2 performs best, when analyzing the combinations with its higher I_{IGD} and I_{IGD+}, it can be concluded that its exploration capabilities are relatively limited. It is strong in maintaining close proximity to the reference front but struggles with diversity across the front.

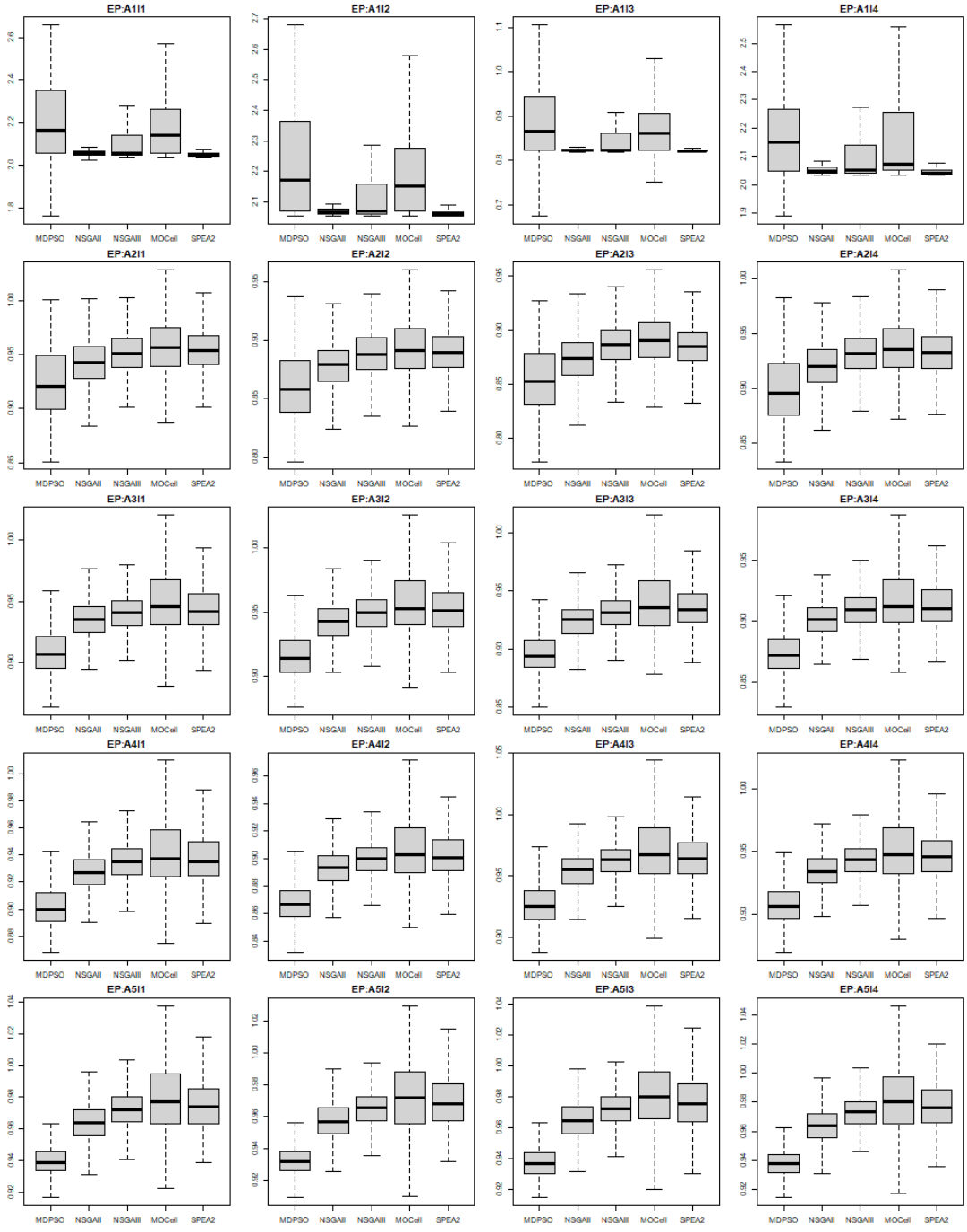


Figure 5.1: Boxplots for quality indicator results of $I_{\epsilon+}$

The Friedman test, shown in table 5.4, characterizes the same picture as analyzed above. First of all, the null hypothesis (H_0) can be rejected for all four quality indicators

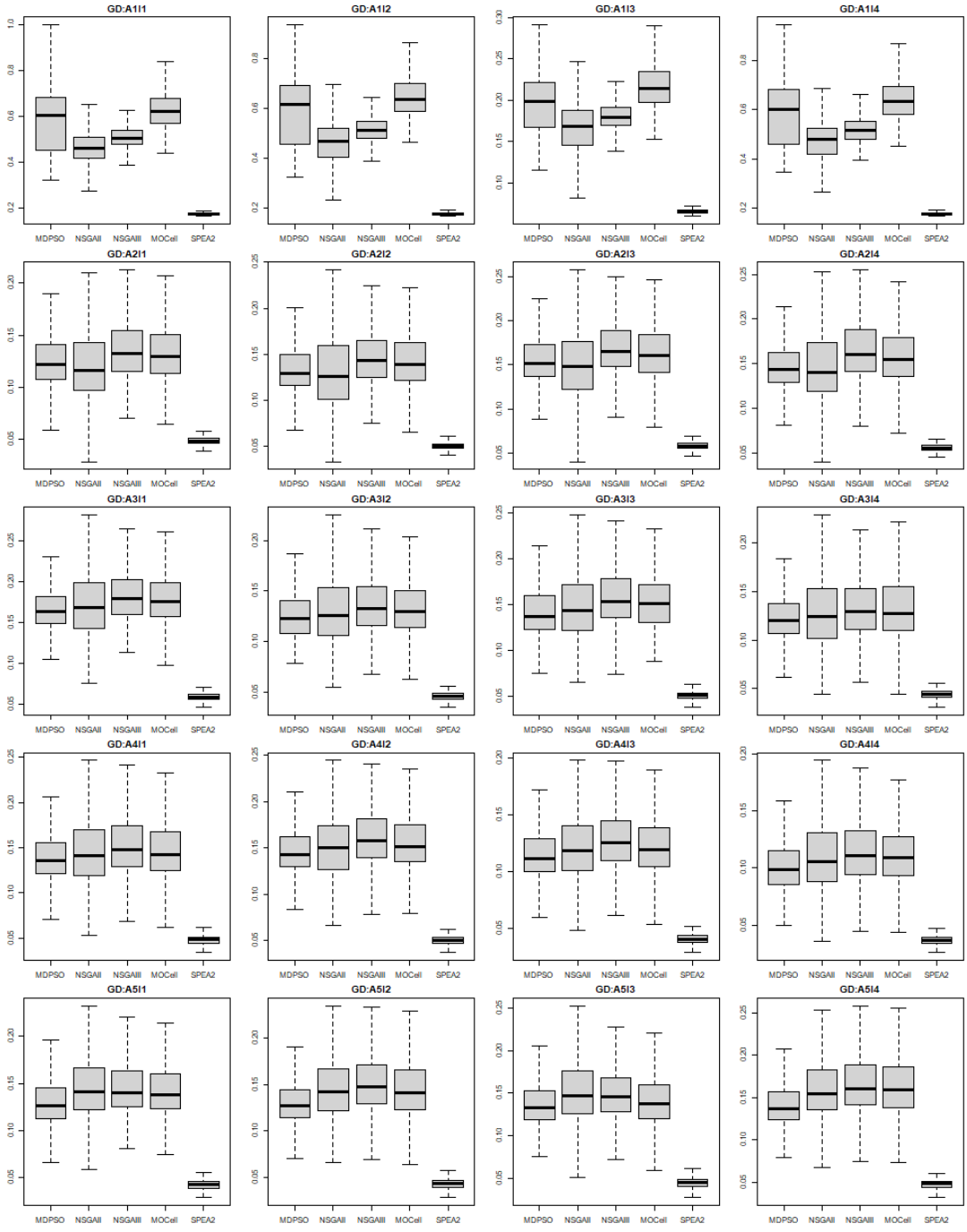


Figure 5.2: Boxplots for quality indicator results of I_{GD}

as $\chi^2_{observed} > \chi^2_{critical}$, which means that the performances of the algorithms differ significantly. The test outlines the exceptional performance of MDPSO for $I_{\epsilon+}$, I_{IGD} and

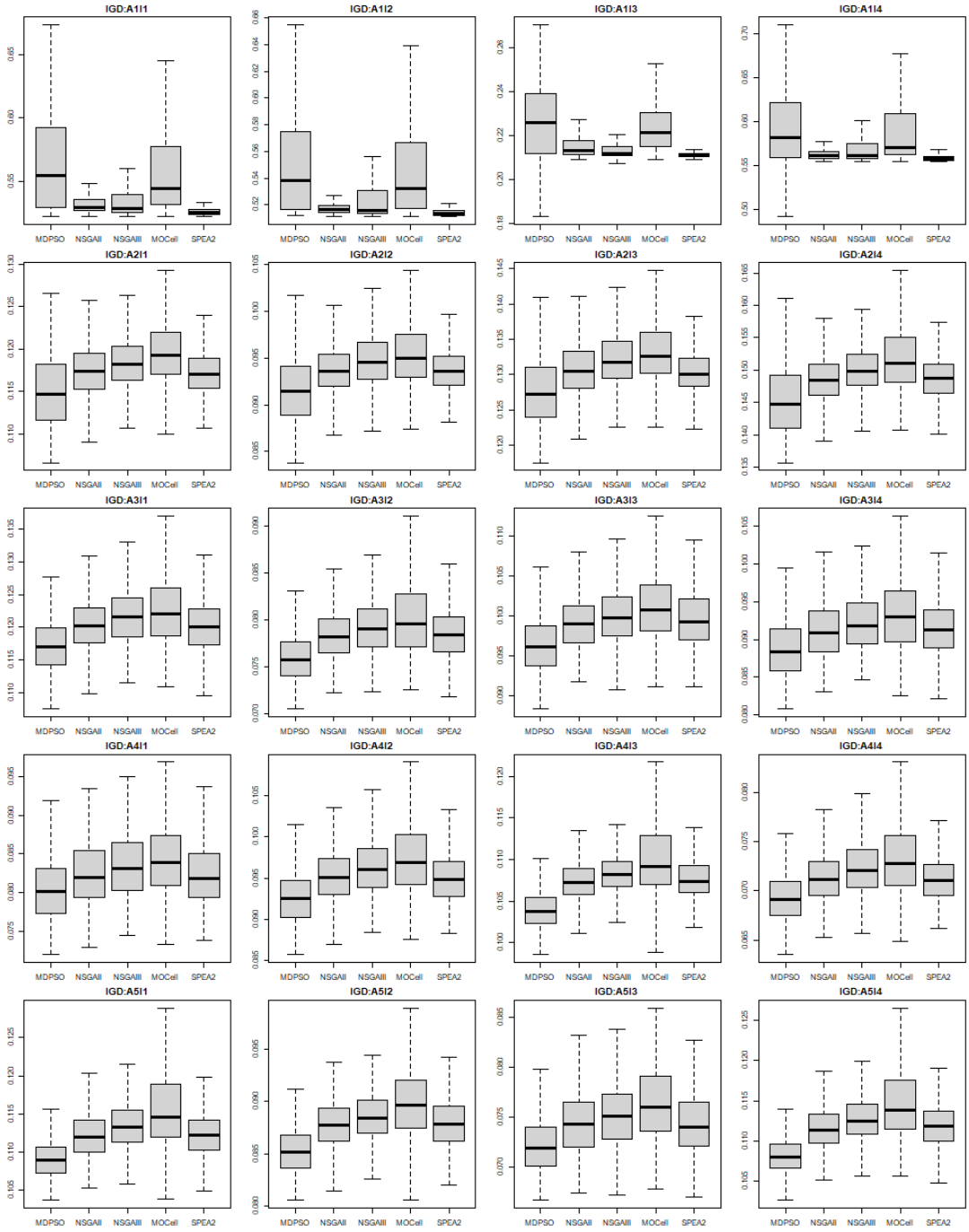


Figure 5.3: Boxplots for quality indicator results of I_{IGD}

I_{IGD+} with a ranking of 1.8 for each of them. Another aspect which identifies SPEA2 as the definite best is the reference to I_{IG} with a ranking of 1.0. MDPSO is the second best

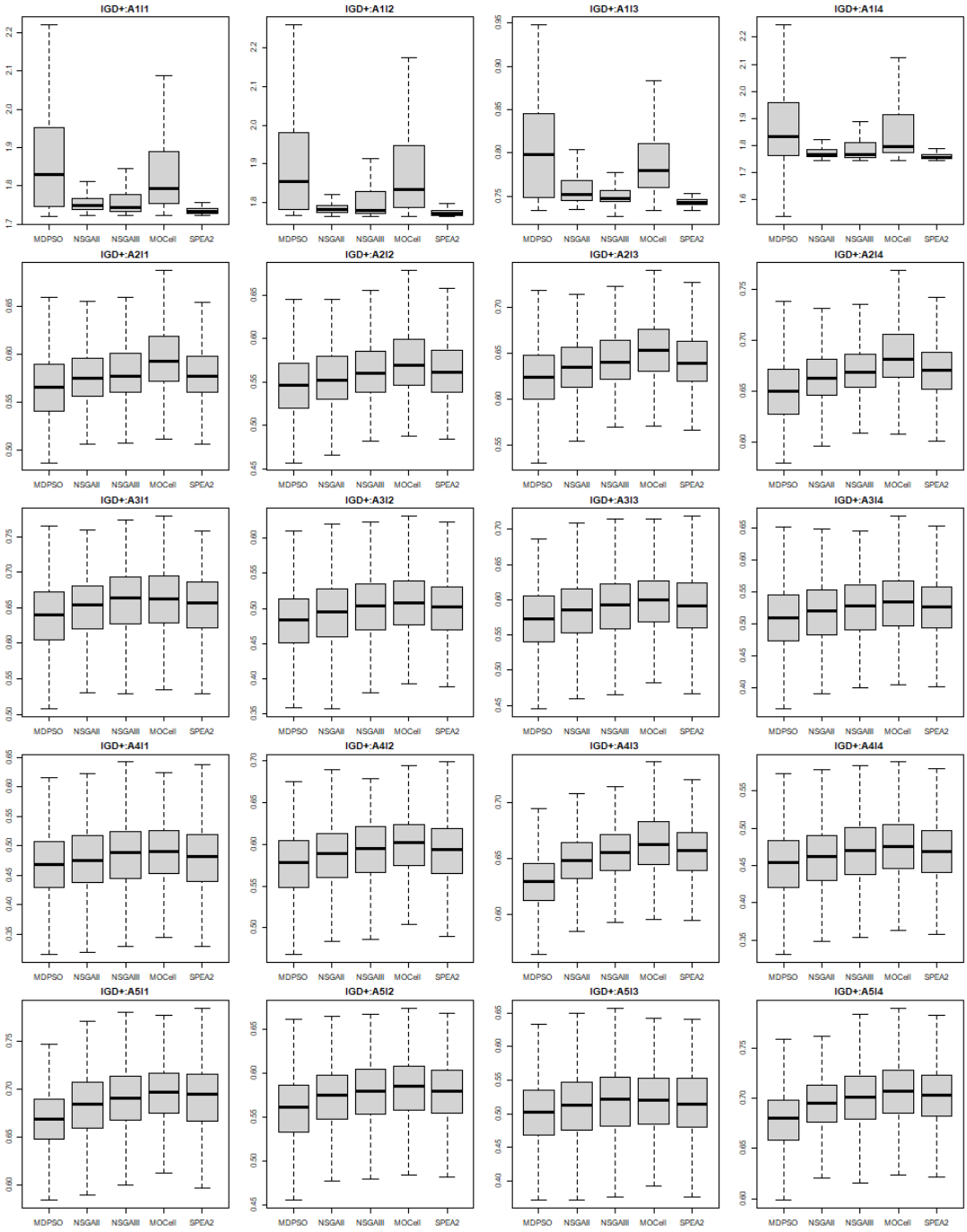


Figure 5.4: Boxplots for quality indicator results of I_{IGD+}

with 2.6.

Table 5.4: Friedman test - Average ranking of the algorithms by quality indicator

Indicator	MDPSO	NSGA-II	NSGA-III	MOCell	SPEA2	$\chi^2_{observed}$
$I_{\epsilon+}$	1.8	2.0	3.1	4.8	3.3	46.24
I_{GD}	2.6	2.9	4.5	4.0	1.0	59.28
I_{IGD}	1.8	2.5	3.7	4.8	2.3	47.16
I_{IGD+}	1.8	2.15	3.4	4.8	2.85	44.68
I_{Δ}	2.0	1.4	3.6	3.1	5.0	62.92
I_{HV}	1.4	2.2	3.0	4.5	4.0	50.96
I_{ER}	1.5	2.4	3.0	3.2	5.0	54.44
I_{T}	3.3	2.5	1.1	3.2	5.0	65.88

Table 5.5: Wilcoxon Test - pairwise comparison of the MDPSO result against each problem instance of the reference algorithms (see section 5.2.3 for explanation)

		NSGA-II				NSGA-III				MOCell				SPEA2			
		$\mathcal{I}_1$	$\mathcal{I}_2$	$\mathcal{I}_3$	$\mathcal{I}_4$	$\mathcal{I}_1$	$\mathcal{I}_2$	$\mathcal{I}_3$	$\mathcal{I}_4$	$\mathcal{I}_1$	$\mathcal{I}_2$	$\mathcal{I}_3$	$\mathcal{I}_4$	$\mathcal{I}_1$	$\mathcal{I}_2$	$\mathcal{I}_3$	$\mathcal{I}_4$
I_{EP}	$\mathcal{A}_1$	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	–	▽	▽	▽	▽
	$\mathcal{A}_2$	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲
	$\mathcal{A}_3$	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲
	$\mathcal{A}_4$	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲
	$\mathcal{A}_5$	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲
I_{GD}	$\mathcal{A}_1$	▽	▽	▽	▽	▽	▽	▽	▽	▲	▲	▲	▲	▽	▽	▽	▽
	$\mathcal{A}_2$	▽	▽	▽	▽	▲	▲	▲	▲	▲	▲	▲	▲	▽	▽	▽	▽
	$\mathcal{A}_3$	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▽	▽	▽	▽
	$\mathcal{A}_4$	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▽	▽	▽	▽
	$\mathcal{A}_5$	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▽	▽	▽	▽
I_{IGD}	$\mathcal{A}_1$	▽	▽	▽	▽	▽	▽	▽	▽	–	–	–	–	▽	▽	▽	▽
	$\mathcal{A}_2$	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲
	$\mathcal{A}_3$	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲
	$\mathcal{A}_4$	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲
	$\mathcal{A}_5$	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲
I_{IGD+}	$\mathcal{A}_1$	▽	▽	▽	▽	▽	▽	▽	▽	–	–	▽	–	▽	▽	▽	▽
	$\mathcal{A}_2$	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲
	$\mathcal{A}_3$	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲
	$\mathcal{A}_4$	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲
	$\mathcal{A}_5$	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲
I_{Δ}	$\mathcal{A}_1$	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▲	▲	▲	▲
	$\mathcal{A}_2$	▽	▽	▽	▽	▲	▲	▲	▲	–	▲	▲	▲	▲	▲	▲	▲
	$\mathcal{A}_3$	–	–	–	–	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲
	$\mathcal{A}_4$	–	–	–	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲
	$\mathcal{A}_5$	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲
I_{HV}	$\mathcal{A}_1$	–	–	▽	–	–	–	▽	–	–	–	▲	–	–	–	▽	–
	$\mathcal{A}_2$	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲
	$\mathcal{A}_3$	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲
	$\mathcal{A}_4$	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲
	$\mathcal{A}_5$	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲
I_{ER}	$\mathcal{A}_1$	–	–	–	–	–	–	–	–	–	–	–	–	▲	▲	–	▲
	$\mathcal{A}_2$	–	–	–	–	–	▲	–	–	▲	–	–	–	–	▲	–	–
	$\mathcal{A}_3$	–	–	–	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲
	$\mathcal{A}_4$	▲	–	–	▲	▲	▲	–	▲	▲	▲	▲	▲	▲	▲	▲	▲
	$\mathcal{A}_5$	–	▲	▲	–	▲	▲	▲	–	▲	–	▲	–	▲	▲	▲	▲
I_{T}	$\mathcal{A}_1$	▽	▽	▽	▽	▲	–	▽	▽	▽	▽	▽	▽	▲	▲	▲	▲
	$\mathcal{A}_2$	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▲	▲	▲	▲
	$\mathcal{A}_3$	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽
	$\mathcal{A}_4$	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽
	$\mathcal{A}_5$	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽

The statistical confidence in this ranking can be checked by using the critical distance (CD) from the post-hoc Nemenyi test. As calculated in section 5.2.3, the CD in this experiment is 1.285. By adding this CD to the MDPSO ranking, $1.8 + 1.285 = 3.085$, it can be concluded that MDPSO performs significantly better than NSGA-III and MOCell in terms of all four quality indicators $I_{\epsilon+}$, I_{GD}, I_{IGD}, and I_{IGD+} as well as SPEA2 in terms of $I_{\epsilon+}$. On the other hand, SPEA2 outperforms all algorithms significantly, in terms of I_{GD}. While MDPSO is better ranked than NSGA-II across all 4 indicators as well as SPEA2 regarding I_{IGD} and I_{IGD+}, it cannot be justified with statistical confidence. The ranking also shows that NSGA-II closely follows MDPSO and presents itself as a strong general-purpose algorithm, whereas NSGA-III shows its drawbacks in optimizing problems with three objectives due to its design for efficiently searching solutions in higher-dimensional search spaces. The concerning indicators show that MOCell struggles with convergence due to its localized search approach.

5.3.2 Diversity

Diversity analysis assesses how well the solutions, which are generated by optimization algorithms, are distributed across the reference Pareto front. A diverse set of solutions ensures that decision-makers have a wide range of trade-offs to choose from. This aspect is evaluated by comparing the I_Δ **(Generalized SPREAD)** quality indicator, which measures the extent to which solutions are evenly and well distributed. As described in combination with the calculation in section 5.2.2, lower I_Δ values indicate better diversity.

As shown in table A.5, MDPSO and NSGA-II alternate in being the best and second-best performers. Even though they are close together, MDPSO consistently achieves the best mean by starting with problems of $\mathcal{A}_4$ and demonstrating its ability to produce well-distributed solutions with growing complexity within the problems. This phenomenon also applies to the median values, which also be encountered in the boxplot of figure 5.5. However, as the Wilcoxon test in table 5.5 indicates that both algorithms are neighboring for $\mathcal{A}_3$ and $\mathcal{A}_4$ problems to such an extent that no statistical significance can be encountered in 7 out of 20 problem instances.

The only exception, similar as seen in the previous comparison for convergence, are the results of the first 4 problem instances, which deploy the smallest application workload $\mathcal{A}_1$. Here, the MDPSO algorithm is significantly dominated by NSGA-III and MOCell, as shown in the Wilcoxon test table 5.5.

What stands out very clearly is the poor performance of SPEA2 in terms of diversity, compared across all algorithms and problem instances, as can be seen in the boxplots of figure 5.5. This can be traced back, as already demonstrated in the previous section, to its relatively limited exploration capabilities.

When checking the Friedman test table 5.4, it can be concluded that average ranking for I_Δ differs significantly, since the H_0 can again be rejected. This test clearly ranks NSGA-II as best with a rank of 1.4 compared to MDPSO as second best, having 2.0.

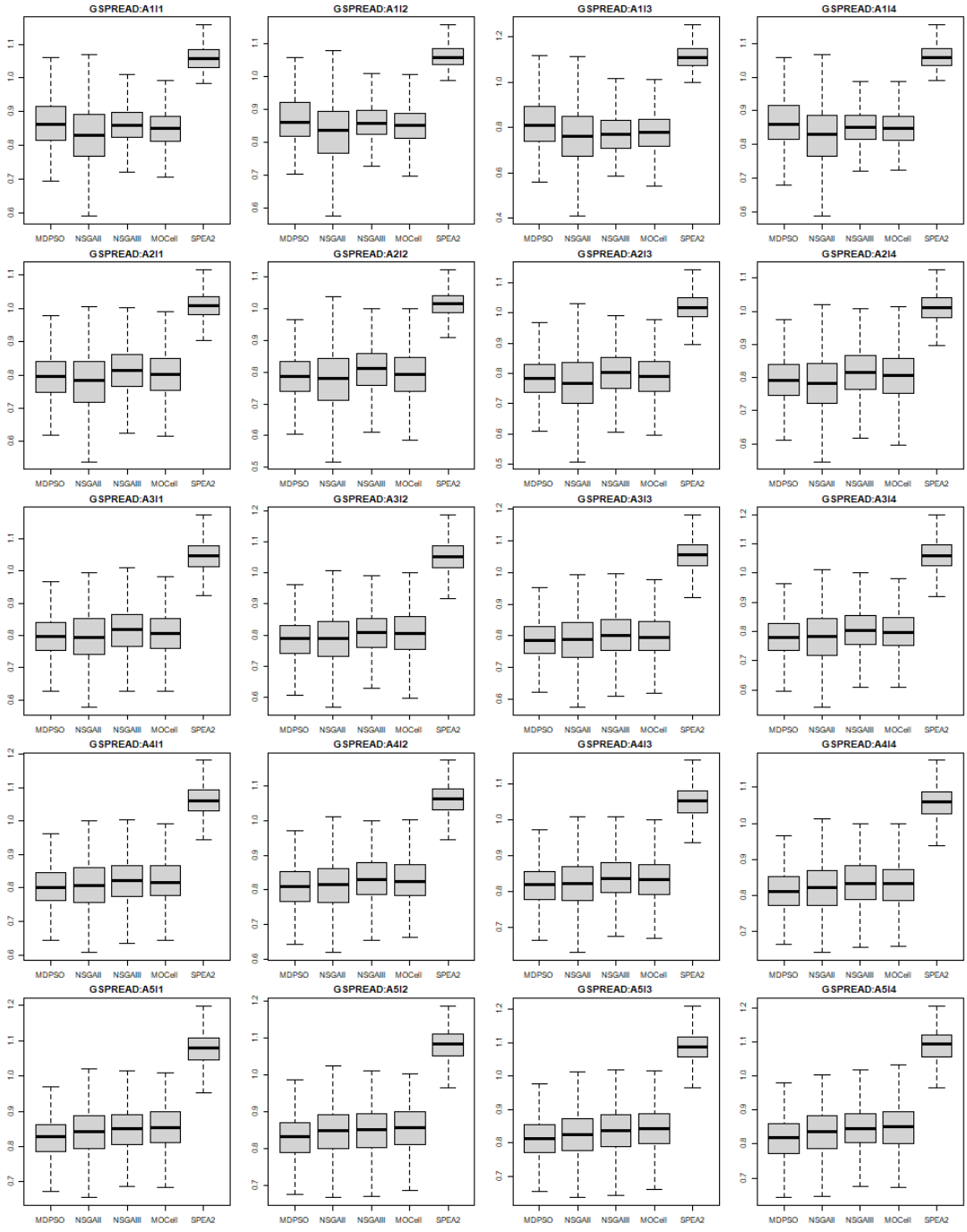


Figure 5.5: Boxplots for quality indicator results of I_{Δ}

However, considering the critical distance from the Nemenyi test, this ranking cannot be made with statistical certainty. Compared with the other algorithms, it is significantly

better than NSGA-III and SPEA2, but not as good as MOCell.

5.3.3 Trade-off Analysis

The trade-off analysis evaluates how effectively the MDPSO algorithm balances convergence and diversity, which are essential for providing high-quality solutions in multi-objective optimization. Therefore, the I_{HV} (Hypervolume) quality indicator is used to measure the volume of the objective space dominated by the approximated Pareto front relative to a reference point, is defined by the vector of the worst values. As described in section 5.2.2, the I_{HV} is the only indicator where a larger value implies a better approximation.

As underpinned by the results in table A.6, MDPSO is outlined as the best performer for the problem instances starting with $\mathcal{A}_2$ and again followed by NSGA-II. Unlike the diversity analysis, the distance between MDPSO and NSGA-II is not that close, also well observable in the boxplot collection in figure 5.6. This is also substantiated with statistical confidence by the Wilcoxon test in table 5.5, where it is shown that MDPSO dominates all algorithms in terms of I_{HV} for those problems.

Also, the Friedman ranking test identifies the MDPSO algorithm as the best, with a rank of 1.4. Validating against the critical distance of the post-hoc Nemenyi test, this makes the trade-off performance of the MDPSO statistically significantly better than NSGA-III, MOCell and SPEA2.

5.3.4 Accuracy

Accuracy analysis examines how many solutions in the approximation set produced by an algorithm are contained in the reference Pareto front. By using reference Pareto fronts, this provides a measure of how effectively the algorithm identifies Pareto-optimal solutions across all algorithms. As described in section 5.2.2, the I_{ER} (Error Ratio) indicator is used by calculating the proportion of solutions in the approximation set that are not part of the reference Pareto front. Therefore, a lower value is better.

Due to the nature of this indicator and the large sample set of 1000 executions per problem per algorithm, the median and interquartile range and therefore the boxplot visualization, can be ignored. However, when checking the mean value comparison in tables A.7, MDPSO achieves the lowest mean error ratio in 13 out of 20 problem instances, demonstrating its ability to accurately approximate the reference Pareto front.

The statistical significance of the pairwise comparison of this quality indicator done by the Wilcoxon test in table 5.5 is not as clear as the other indicators. However, the test highlights the MDPSO's dominance over NSGA-III, MOCell and SPEA2 in more complex problem instances.

The Friedman test (see table 5.4) on the other side outlines a statistically significant difference in the performance and again ranks MDPSO in the best place with a ranking of 1.5. By applying the Nemenyi test on this ranking, the MDPSO algorithm is even

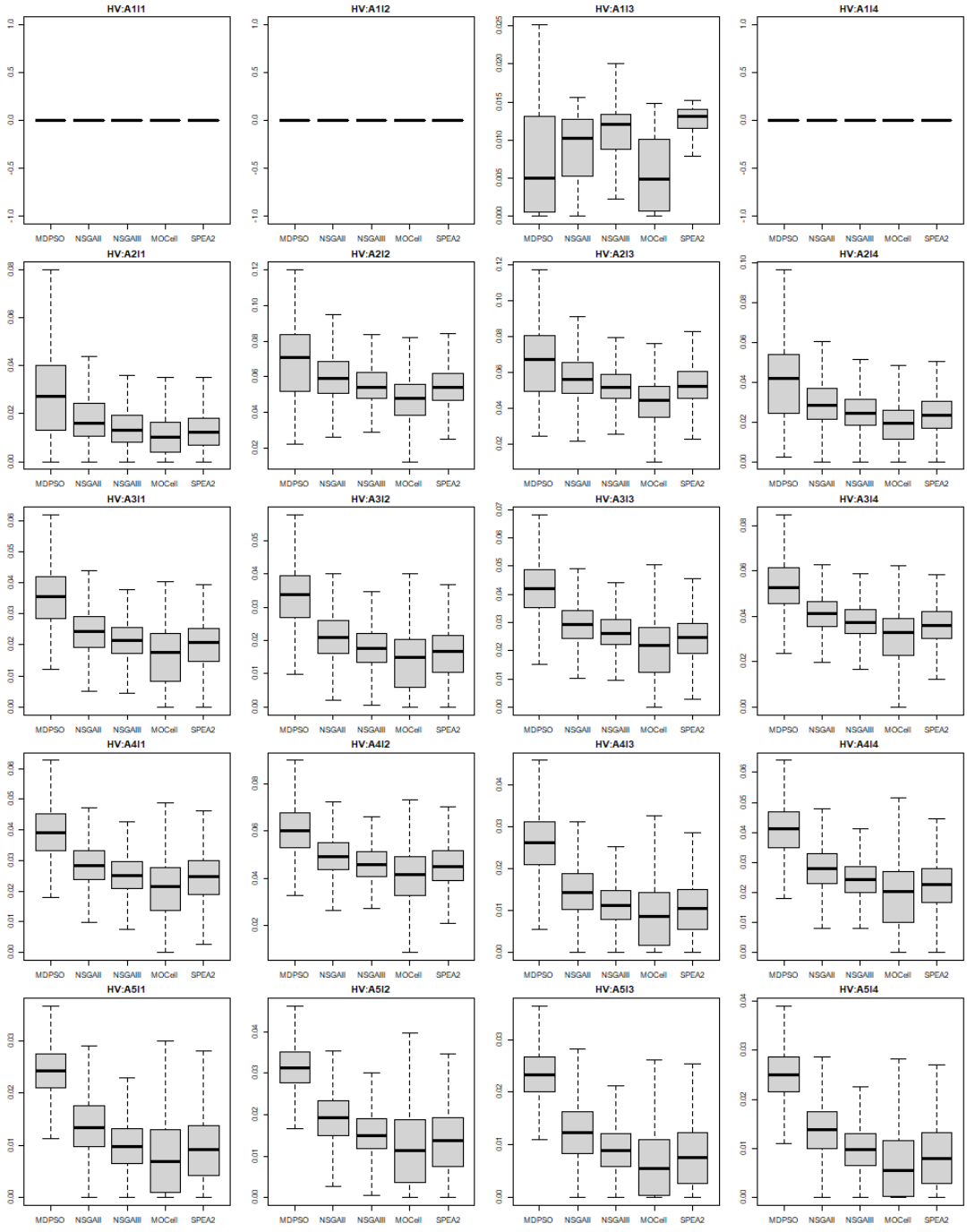


Figure 5.6: Boxplots for quality indicator results of I_{HV}

more confidently better than NSGA-III, MOCell and SPEA2, but the critical distance given by this evaluation is too large to enclose NSGA-II.

f

5.3.5 Efficiency

Last but not least, the efficiency analysis evaluates the computational resources required by an algorithm to achieve its results, with a focus on the time taken to produce a high-quality approximation of the Pareto front. This evaluation is done by comparing the I_T (Computational Time) indicator, which measures the total computational time required for an algorithm to reach the specified stopping criteria. Therefore, its objective is to be minimized.

The values in table A.8 highlight NSGA-III as the most efficient and computationally fastest algorithm, even with the growing complexity of the problem instances.

In terms of the mean, MDPSO appears to be a solid performer by being located closely to MOCell while SPEA2 is by far the adverst performer. This result is underpinned by the Friedman test in table 5.4, where MDPSO is clearly ranked lower than NSGA-II and NSGA-II, however, significant better than SPEA2.

However, when comparing the median visualized in the boxplot collection in figure 5.7, MDPSO performs poorly. According to the Wilcoxon test in table 5.5 MDPSO, is indeed significantly better in solving simpler problems, nevertheless, compared to the other algorithms, its significant confidence is worse. It is even outperformed by SPEA2 when it comes to more complex problem instances.

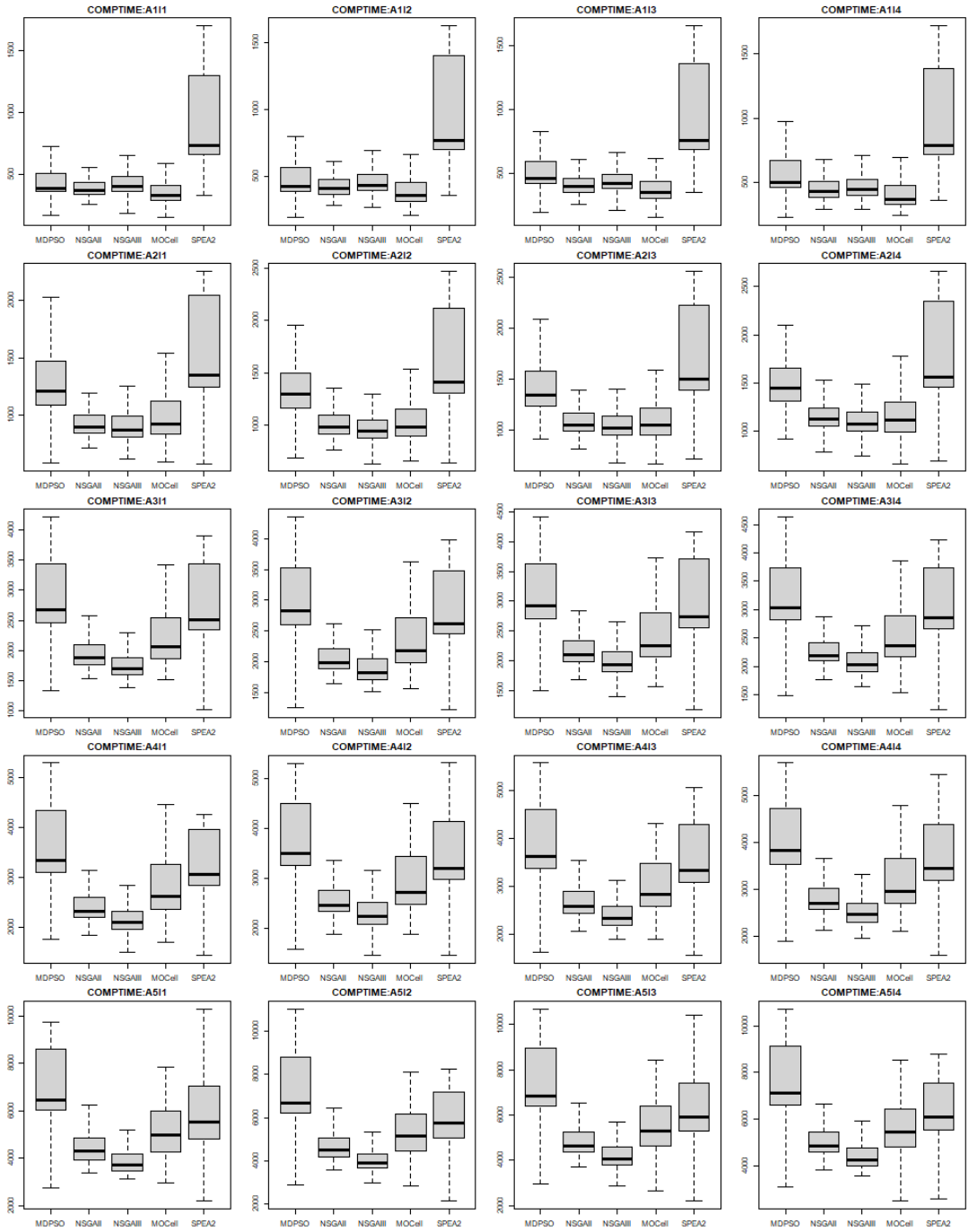


Figure 5.7: Boxplots for quality indicator results of I_T

CHAPTER 6

Conclusio

This thesis investigated the optimization of Mobile Edge Cloud Offloading (MECO) and its conflicting objectives of minimizing the application runtime, maximizing the mobile device battery lifetime as well as minimizing the user costs. This issue was dealt with by proposing a Multi-Objective Discrete Particle Swarm Optimization (MDPSO) algorithm tailored to this deployment problem and capable of finding high-quality Pareto-optimal solutions. To handle the discrete nature of the task-to-node mappings in combination with the principles of particle swarm optimizations, the algorithm was designed with a dedicated leader archive strategy, an objective-based global best selection rule as well as a particle encoding and velocity updated mechanism. Eventually, the results were evaluated by comparing the MDPSO algorithm against established reference algorithms, including NSGA-II, NSGA-III, MOCell and SPEA2. 20 problem instances with varying complexity across 1000 independent runs per instance were used to increase the statistical significance.

MDPSO was the most convincing in terms of *convergence*, as can be seen in its dominance of the quality indicators such as $I_{\epsilon+}$, I_{IGD} and I_{IGD+}, which is statistically significant across the majority of problem instances, particularly in complex scenarios. However, for simpler instances with lower-dimensional particle vectors, MDPSO was outperformed by SPEA2, indicating potential areas for refinement in handling less complex tasks.

In terms of *diversity*, by maintaining a well-distributed set of solutions across the Pareto front, MDPSO can compete with NSGA-II. By balancing *convergence* and *diversity*, MDPSO highlighted its strength shown by outperforming the other algorithms with statistical significance in the hypervolume indicator (I_{HV}). This underlines its capability to provide decision-makers with high-quality trade-offs.

Additionally, MDPSO achieved a high level of *accuracy* in identifying solutions close to the reference Pareto front, achieving the lowest error ratio in most cases and outperforming

NSGA-III, MOCell, and SPEA2. However, its performance was comparable to NSGA-II, particularly in simpler problem scenarios.

While constantly showing good results in previously mentioned performance dimensions, the *efficiency* analysis revealed a trade-off. MDPSO achieved competitive results in computational time relative to MOCell and SPEA2, however, it was clearly outperformed by NSGA-II and the most efficient algorithm, NSGA-III. This limitation highlights the need for further computational optimizations to enhance its scalability.

In summary, MDPSO proved to be a robust algorithm for solving the MECO deployment problem, particularly excelling in convergence, diversity and trade-off quality. Its evaluated performance highlights its applicability for complex multi-objective optimization tasks. For simpler problems and time-sensitive applications other algorithms display higher efficiency.

The findings of this thesis underline the potential of the MDPSO algorithm for solving multi-objective optimization problems in Mobile Edge Cloud Offloading. Nevertheless, various possible improvements according to the identified limitations and additional features which can further enhance the algorithm's application possibilities need to be outlined:

Improving Computational Efficiency: According to the evaluation, it was revealed that MDPSO shows higher computational costs compared to well-known MOO algorithms like NSGA-II and NSGA-III. In this respect, further research could be conducted in order to explore parallel and distributed implementations of MDPSO to leverage multi-core and cloud-based computational resources. Additionally, an adaptive stopping criteria could be embedded, which can help to balance solution quality and computational time for time-sensitive applications. Addressing the discovered difficulties in simpler problem instances, future work could investigate hybrid approaches that combine MDPSO with simpler optimization techniques, by dynamically selecting the most suitable strategy based on problem characteristics.

Dynamic Adaptation for Real-World Environments: Real-world MECO scenarios involve dynamic changes in network conditions and resource availability. Adding a dynamic adaptation mechanism to the infrastructure as well as applications workloads, as via event-driven updates, could enable the algorithm to respond to real-time without the need to stop and rerun the algorithm.

Add Preference-Based Optimization: Involved users or stakeholders of MECO scenarios often have preferences regarding certain objectives, such as prioritizing cost reduction over energy efficiency. A future research could be to integrate a preference-based optimization method into MDPSO, such as reference points or weighted objective adjustments. This could allow for tailored solutions that align with specific user goals.

Generalization Across Problem Domains: While this thesis focused on MECO deployments, the generalization of MDPSO according to other discrete optimization problems could be part of further research. Additionally, future work could evaluate the possibility of expanding the number of objective functions in the algorithm by modifying the global best selection mechanism and the velocity calculation rules.

APPENDIX A

Quality Indicator Results

This appendix contains the detailed results for the following quality indicators calculated for the proposed MDPSO algorithm and the reference algorithms NSGA-II, NSGA-III, MOCell and SPEA2. The tables contain a comparison for the mean and standard deviation as well as for the median and interquartile range. The best performing algorithm for each problem instance is highlighted dark gray and the second best light gray. In addition to these result tables, a collection of boxplots for each quality indicator for each problem instance is illustrated in section 5.3, in order to provide a visually better comparison.

- $I_{\epsilon+}$ results can be find in Table A.1 and boxplots in Figure 5.1
- I_{GD} results can be find in Table A.2 and boxplots in Figure 5.2
- I_{IGD} results can be find in Table A.3 and boxplots in Figure 5.3
- I_{IGD+} results can be find in Table A.4 and boxplots in Figure 5.4
- I_{Δ} results can be find in Table A.5 and boxplots in Figure 5.5
- I_{HV} results can be find in Table A.6 and boxplots in Figure 5.6
- I_{ER} results can be find in Table A.7, boxplots are skipped since the median has no significance for this indicator
- I_{T} results can be find in Table A.8 and boxplots in Figure 5.7

Table A.1: Mean and Median Results for $I_{\epsilon+}$

		MDPSO	NSGA-II	NSGA-III	MOCell	SPEA2
Mean and Standard Deviation	$\mathcal{A}_1\mathcal{I}_1$	$2.19e+00_{2.1e-01}$	$2.08e+00_{1.1e-01}$	$2.08e+00_{1.3e-01}$	$2.17e+00_{1.8e-01}$	$2.05e+00_{8.4e-02}$
	$\mathcal{A}_1\mathcal{I}_2$	$2.20e+00_{2.0e-01}$	$2.09e+00_{1.0e-01}$	$2.10e+00_{1.5e-01}$	$2.18e+00_{2.0e-01}$	$2.07e+00_{6.9e-02}$
	$\mathcal{A}_1\mathcal{I}_3$	$8.82e-01_{7.2e-02}$	$8.33e-01_{4.0e-02}$	$8.38e-01_{3.9e-02}$	$8.74e-01_{6.7e-02}$	$8.24e-01_{2.7e-02}$
	$\mathcal{A}_1\mathcal{I}_4$	$2.17e+00_{1.9e-01}$	$2.07e+00_{1.1e-01}$	$2.09e+00_{1.1e-01}$	$2.16e+00_{2.0e-01}$	$2.05e+00_{7.6e-02}$
	$\mathcal{A}_2\mathcal{I}_1$	$9.15e-01_{7.7e-02}$	$9.40e-01_{4.1e-02}$	$9.48e-01_{5.0e-02}$	$9.60e-01_{6.6e-02}$	$9.48e-01_{6.9e-02}$
	$\mathcal{A}_2\mathcal{I}_2$	$8.57e-01_{5.0e-02}$	$8.75e-01_{4.8e-02}$	$8.87e-01_{3.8e-02}$	$8.96e-01_{6.1e-02}$	$8.89e-01_{4.3e-02}$
	$\mathcal{A}_2\mathcal{I}_3$	$8.54e-01_{4.4e-02}$	$8.69e-01_{5.1e-02}$	$8.83e-01_{4.8e-02}$	$8.96e-01_{5.4e-02}$	$8.83e-01_{4.6e-02}$
	$\mathcal{A}_2\mathcal{I}_4$	$8.98e-01_{4.6e-02}$	$9.14e-01_{6.3e-02}$	$9.29e-01_{4.8e-02}$	$9.42e-01_{5.8e-02}$	$9.31e-01_{4.8e-02}$
	$\mathcal{A}_3\mathcal{I}_1$	$9.07e-01_{3.6e-02}$	$9.30e-01_{5.4e-02}$	$9.39e-01_{3.6e-02}$	$9.50e-01_{4.8e-02}$	$9.43e-01_{3.7e-02}$
	$\mathcal{A}_3\mathcal{I}_2$	$9.14e-01_{4.6e-02}$	$9.39e-01_{4.9e-02}$	$9.48e-01_{4.1e-02}$	$9.53e-01_{6.8e-02}$	$9.52e-01_{4.3e-02}$
	$\mathcal{A}_3\mathcal{I}_3$	$8.94e-01_{4.5e-02}$	$9.21e-01_{5.1e-02}$	$9.27e-01_{5.2e-02}$	$9.39e-01_{5.1e-02}$	$9.35e-01_{3.5e-02}$
	$\mathcal{A}_3\mathcal{I}_4$	$8.72e-01_{4.2e-02}$	$8.99e-01_{4.6e-02}$	$9.06e-01_{4.7e-02}$	$9.16e-01_{5.0e-02}$	$9.11e-01_{4.4e-02}$
	$\mathcal{A}_4\mathcal{I}_1$	$8.99e-01_{4.7e-02}$	$9.24e-01_{4.3e-02}$	$9.31e-01_{5.1e-02}$	$9.40e-01_{4.0e-02}$	$9.35e-01_{4.5e-02}$
	$\mathcal{A}_4\mathcal{I}_2$	$8.64e-01_{4.8e-02}$	$8.89e-01_{4.8e-02}$	$8.97e-01_{3.9e-02}$	$9.05e-01_{4.1e-02}$	$8.99e-01_{4.9e-02}$
	$\mathcal{A}_4\mathcal{I}_3$	$9.23e-01_{5.1e-02}$	$9.50e-01_{5.6e-02}$	$9.59e-01_{5.2e-02}$	$9.71e-01_{3.8e-02}$	$9.62e-01_{5.0e-02}$
	$\mathcal{A}_4\mathcal{I}_4$	$9.02e-01_{6.0e-02}$	$9.31e-01_{4.8e-02}$	$9.40e-01_{4.8e-02}$	$9.49e-01_{4.7e-02}$	$9.46e-01_{3.2e-02}$
	$\mathcal{A}_5\mathcal{I}_1$	$9.35e-01_{5.6e-02}$	$9.57e-01_{6.3e-02}$	$9.67e-01_{5.8e-02}$	$9.77e-01_{4.0e-02}$	$9.73e-01_{4.0e-02}$
	$\mathcal{A}_5\mathcal{I}_2$	$9.27e-01_{5.8e-02}$	$9.50e-01_{6.9e-02}$	$9.63e-01_{4.5e-02}$	$9.67e-01_{6.6e-02}$	$9.66e-01_{5.0e-02}$
	$\mathcal{A}_5\mathcal{I}_3$	$9.36e-01_{3.5e-02}$	$9.59e-01_{5.9e-02}$	$9.69e-01_{4.8e-02}$	$9.78e-01_{4.9e-02}$	$9.74e-01_{4.4e-02}$
	$\mathcal{A}_5\mathcal{I}_4$	$9.33e-01_{5.6e-02}$	$9.58e-01_{6.2e-02}$	$9.69e-01_{4.7e-02}$	$9.78e-01_{4.7e-02}$	$9.75e-01_{3.7e-02}$
Median and Interquartile Range	$\mathcal{A}_1\mathcal{I}_1$	$2.16e+00_{3.0e-01}$	$2.05e+00_{2.0e-02}$	$2.06e+00_{9.3e-02}$	$2.14e+00_{2.1e-01}$	$2.05e+00_{1.4e-02}$
	$\mathcal{A}_1\mathcal{I}_2$	$2.17e+00_{3.0e-01}$	$2.07e+00_{1.6e-02}$	$2.07e+00_{9.9e-02}$	$2.15e+00_{2.1e-01}$	$2.06e+00_{1.4e-02}$
	$\mathcal{A}_1\mathcal{I}_3$	$8.66e-01_{1.2e-01}$	$8.23e-01_{4.3e-03}$	$8.24e-01_{4.0e-02}$	$8.62e-01_{8.2e-02}$	$8.21e-01_{3.3e-03}$
	$\mathcal{A}_1\mathcal{I}_4$	$2.15e+00_{2.2e-01}$	$2.05e+00_{1.9e-02}$	$2.05e+00_{9.8e-02}$	$2.07e+00_{2.0e-01}$	$2.04e+00_{1.6e-02}$
	$\mathcal{A}_2\mathcal{I}_1$	$9.20e-01_{4.9e-02}$	$9.42e-01_{3.0e-02}$	$9.51e-01_{2.6e-02}$	$9.56e-01_{3.6e-02}$	$9.54e-01_{2.7e-02}$
	$\mathcal{A}_2\mathcal{I}_2$	$8.58e-01_{4.4e-02}$	$8.79e-01_{2.7e-02}$	$8.88e-01_{2.7e-02}$	$8.91e-01_{3.4e-02}$	$8.90e-01_{2.7e-02}$
	$\mathcal{A}_2\mathcal{I}_3$	$8.53e-01_{4.7e-02}$	$8.74e-01_{3.1e-02}$	$8.86e-01_{2.7e-02}$	$8.90e-01_{3.3e-02}$	$8.85e-01_{2.6e-02}$
	$\mathcal{A}_2\mathcal{I}_4$	$8.95e-01_{4.7e-02}$	$9.20e-01_{3.0e-02}$	$9.32e-01_{2.7e-02}$	$9.36e-01_{3.6e-02}$	$9.33e-01_{2.9e-02}$
	$\mathcal{A}_3\mathcal{I}_1$	$9.07e-01_{2.6e-02}$	$9.35e-01_{2.1e-02}$	$9.41e-01_{2.0e-02}$	$9.46e-01_{3.6e-02}$	$9.42e-01_{2.5e-02}$
	$\mathcal{A}_3\mathcal{I}_2$	$9.14e-01_{2.5e-02}$	$9.43e-01_{2.1e-02}$	$9.50e-01_{2.1e-02}$	$9.53e-01_{3.4e-02}$	$9.51e-01_{2.6e-02}$
	$\mathcal{A}_3\mathcal{I}_3$	$8.94e-01_{2.4e-02}$	$9.25e-01_{2.1e-02}$	$9.31e-01_{2.1e-02}$	$9.36e-01_{3.8e-02}$	$9.34e-01_{2.5e-02}$
	$\mathcal{A}_3\mathcal{I}_4$	$8.73e-01_{2.4e-02}$	$9.02e-01_{1.9e-02}$	$9.10e-01_{2.1e-02}$	$9.13e-01_{3.6e-02}$	$9.10e-01_{2.6e-02}$
	$\mathcal{A}_4\mathcal{I}_1$	$9.00e-01_{2.1e-02}$	$9.27e-01_{1.9e-02}$	$9.35e-01_{1.9e-02}$	$9.37e-01_{3.5e-02}$	$9.35e-01_{2.5e-02}$
	$\mathcal{A}_4\mathcal{I}_2$	$8.66e-01_{1.9e-02}$	$8.93e-01_{1.8e-02}$	$9.00e-01_{1.7e-02}$	$9.03e-01_{3.3e-02}$	$9.01e-01_{2.2e-02}$
	$\mathcal{A}_4\mathcal{I}_3$	$9.25e-01_{2.4e-02}$	$9.55e-01_{2.0e-02}$	$9.63e-01_{1.9e-02}$	$9.68e-01_{3.7e-02}$	$9.64e-01_{2.5e-02}$
	$\mathcal{A}_4\mathcal{I}_4$	$9.06e-01_{2.2e-02}$	$9.34e-01_{1.9e-02}$	$9.43e-01_{1.8e-02}$	$9.48e-01_{3.6e-02}$	$9.46e-01_{2.5e-02}$
	$\mathcal{A}_5\mathcal{I}_1$	$9.39e-01_{1.2e-02}$	$9.64e-01_{1.7e-02}$	$9.72e-01_{1.6e-02}$	$9.77e-01_{3.1e-02}$	$9.74e-01_{2.2e-02}$
	$\mathcal{A}_5\mathcal{I}_2$	$9.32e-01_{1.2e-02}$	$9.57e-01_{1.7e-02}$	$9.65e-01_{1.5e-02}$	$9.72e-01_{3.2e-02}$	$9.68e-01_{2.3e-02}$
	$\mathcal{A}_5\mathcal{I}_3$	$9.37e-01_{1.3e-02}$	$9.64e-01_{1.8e-02}$	$9.72e-01_{1.6e-02}$	$9.80e-01_{3.0e-02}$	$9.76e-01_{2.4e-02}$
	$\mathcal{A}_5\mathcal{I}_4$	$9.38e-01_{1.3e-02}$	$9.64e-01_{1.6e-02}$	$9.73e-01_{1.5e-02}$	$9.81e-01_{3.3e-02}$	$9.76e-01_{2.3e-02}$

Table A.2: Mean and Median Results for I_{GD}

		MDPSO	NSGA-II	NSGA-III	MOCell	SPEA2
Mean and Standard Deviation	$\mathcal{A}_1\mathcal{I}_1$	$5.82e-01_{1.4e-01}$	$4.47e-01_{1.1e-01}$	$5.08e-01_{5.0e-02}$	$6.41e-01_{1.2e-01}$	$1.73e-01_{6.4e-03}$
	$\mathcal{A}_1\mathcal{I}_2$	$5.89e-01_{1.4e-01}$	$4.49e-01_{1.1e-01}$	$5.15e-01_{5.3e-02}$	$6.56e-01_{1.2e-01}$	$1.77e-01_{7.0e-03}$
	$\mathcal{A}_1\mathcal{I}_3$	$1.97e-01_{3.5e-02}$	$1.61e-01_{4.1e-02}$	$1.81e-01_{1.8e-02}$	$2.21e-01_{3.6e-02}$	$6.61e-02_{2.7e-03}$
	$\mathcal{A}_1\mathcal{I}_4$	$5.84e-01_{1.3e-01}$	$4.59e-01_{1.1e-01}$	$5.20e-01_{5.4e-02}$	$6.58e-01_{1.3e-01}$	$1.77e-01_{6.4e-03}$
	$\mathcal{A}_2\mathcal{I}_1$	$1.27e-01_{3.1e-02}$	$1.24e-01_{4.1e-02}$	$1.38e-01_{3.5e-02}$	$1.38e-01_{4.2e-02}$	$4.83e-02_{3.7e-03}$
	$\mathcal{A}_2\mathcal{I}_2$	$1.36e-01_{3.2e-02}$	$1.35e-01_{4.7e-02}$	$1.48e-01_{3.6e-02}$	$1.46e-01_{3.6e-02}$	$5.03e-02_{4.1e-03}$
	$\mathcal{A}_2\mathcal{I}_3$	$1.58e-01_{3.2e-02}$	$1.54e-01_{4.7e-02}$	$1.71e-01_{3.7e-02}$	$1.68e-01_{4.0e-02}$	$5.82e-02_{4.6e-03}$
	$\mathcal{A}_2\mathcal{I}_4$	$1.49e-01_{3.1e-02}$	$1.49e-01_{4.8e-02}$	$1.68e-01_{3.9e-02}$	$1.63e-01_{4.4e-02}$	$5.57e-02_{4.2e-03}$
	$\mathcal{A}_3\mathcal{I}_1$	$1.68e-01_{3.0e-02}$	$1.74e-01_{4.5e-02}$	$1.86e-01_{3.9e-02}$	$1.82e-01_{3.7e-02}$	$5.85e-02_{5.1e-03}$
	$\mathcal{A}_3\mathcal{I}_2$	$1.26e-01_{2.7e-02}$	$1.35e-01_{4.4e-02}$	$1.39e-01_{3.6e-02}$	$1.37e-01_{3.8e-02}$	$4.51e-02_{4.4e-03}$
	$\mathcal{A}_3\mathcal{I}_3$	$1.44e-01_{3.0e-02}$	$1.51e-01_{4.4e-02}$	$1.62e-01_{3.8e-02}$	$1.57e-01_{3.7e-02}$	$5.09e-02_{5.3e-03}$
	$\mathcal{A}_3\mathcal{I}_4$	$1.26e-01_{3.2e-02}$	$1.32e-01_{4.4e-02}$	$1.35e-01_{3.7e-02}$	$1.36e-01_{4.0e-02}$	$4.40e-02_{5.4e-03}$
	$\mathcal{A}_4\mathcal{I}_1$	$1.42e-01_{3.1e-02}$	$1.50e-01_{4.7e-02}$	$1.55e-01_{4.0e-02}$	$1.50e-01_{3.9e-02}$	$4.77e-02_{5.7e-03}$
	$\mathcal{A}_4\mathcal{I}_2$	$1.49e-01_{3.1e-02}$	$1.57e-01_{4.5e-02}$	$1.64e-01_{3.5e-02}$	$1.59e-01_{4.0e-02}$	$4.97e-02_{5.5e-03}$
	$\mathcal{A}_4\mathcal{I}_3$	$1.17e-01_{2.5e-02}$	$1.24e-01_{3.7e-02}$	$1.30e-01_{3.1e-02}$	$1.25e-01_{3.3e-02}$	$4.00e-02_{4.8e-03}$
	$\mathcal{A}_4\mathcal{I}_4$	$1.03e-01_{2.5e-02}$	$1.13e-01_{3.7e-02}$	$1.17e-01_{3.3e-02}$	$1.15e-01_{3.3e-02}$	$3.66e-02_{4.4e-03}$
	$\mathcal{A}_5\mathcal{I}_1$	$1.32e-01_{2.8e-02}$	$1.48e-01_{4.0e-02}$	$1.48e-01_{3.5e-02}$	$1.45e-01_{3.9e-02}$	$4.19e-02_{5.5e-03}$
	$\mathcal{A}_5\mathcal{I}_2$	$1.33e-01_{3.1e-02}$	$1.49e-01_{4.3e-02}$	$1.54e-01_{3.7e-02}$	$1.47e-01_{3.7e-02}$	$4.27e-02_{5.9e-03}$
	$\mathcal{A}_5\mathcal{I}_3$	$1.40e-01_{3.2e-02}$	$1.56e-01_{4.6e-02}$	$1.53e-01_{3.9e-02}$	$1.44e-01_{4.0e-02}$	$4.39e-02_{6.8e-03}$
	$\mathcal{A}_5\mathcal{I}_4$	$1.43e-01_{3.1e-02}$	$1.63e-01_{4.4e-02}$	$1.70e-01_{4.1e-02}$	$1.66e-01_{4.1e-02}$	$4.70e-02_{6.0e-03}$
Median and Interquartile Range	$\mathcal{A}_1\mathcal{I}_1$	$6.03e-01_{2.3e-01}$	$4.61e-01_{9.6e-02}$	$5.02e-01_{6.1e-02}$	$6.20e-01_{1.1e-01}$	$1.72e-01_{7.2e-03}$
	$\mathcal{A}_1\mathcal{I}_2$	$6.13e-01_{2.4e-01}$	$4.68e-01_{1.2e-01}$	$5.11e-01_{6.5e-02}$	$6.35e-01_{1.1e-01}$	$1.75e-01_{7.9e-03}$
	$\mathcal{A}_1\mathcal{I}_3$	$1.98e-01_{5.5e-02}$	$1.68e-01_{4.3e-02}$	$1.79e-01_{2.1e-02}$	$2.14e-01_{3.7e-02}$	$6.60e-02_{3.4e-03}$
	$\mathcal{A}_1\mathcal{I}_4$	$6.00e-01_{2.2e-01}$	$4.78e-01_{1.1e-01}$	$5.15e-01_{7.2e-02}$	$6.32e-01_{1.2e-01}$	$1.75e-01_{8.3e-03}$
	$\mathcal{A}_2\mathcal{I}_1$	$1.22e-01_{3.3e-02}$	$1.16e-01_{4.7e-02}$	$1.32e-01_{3.9e-02}$	$1.29e-01_{3.8e-02}$	$4.83e-02_{4.7e-03}$
	$\mathcal{A}_2\mathcal{I}_2$	$1.30e-01_{3.4e-02}$	$1.26e-01_{5.8e-02}$	$1.43e-01_{4.0e-02}$	$1.39e-01_{4.1e-02}$	$5.04e-02_{5.2e-03}$
	$\mathcal{A}_2\mathcal{I}_3$	$1.52e-01_{3.5e-02}$	$1.48e-01_{5.5e-02}$	$1.65e-01_{4.1e-02}$	$1.61e-01_{4.3e-02}$	$5.83e-02_{5.7e-03}$
	$\mathcal{A}_2\mathcal{I}_4$	$1.43e-01_{3.4e-02}$	$1.40e-01_{5.4e-02}$	$1.60e-01_{4.6e-02}$	$1.55e-01_{4.3e-02}$	$5.56e-02_{5.4e-03}$
	$\mathcal{A}_3\mathcal{I}_1$	$1.63e-01_{3.3e-02}$	$1.68e-01_{5.6e-02}$	$1.79e-01_{4.2e-02}$	$1.75e-01_{4.2e-02}$	$5.91e-02_{6.1e-03}$
	$\mathcal{A}_3\mathcal{I}_2$	$1.23e-01_{3.2e-02}$	$1.26e-01_{4.8e-02}$	$1.33e-01_{3.9e-02}$	$1.30e-01_{3.7e-02}$	$4.53e-02_{5.4e-03}$
	$\mathcal{A}_3\mathcal{I}_3$	$1.38e-01_{3.7e-02}$	$1.43e-01_{5.0e-02}$	$1.53e-01_{4.2e-02}$	$1.51e-01_{4.1e-02}$	$5.15e-02_{6.2e-03}$
	$\mathcal{A}_3\mathcal{I}_4$	$1.21e-01_{3.1e-02}$	$1.24e-01_{5.1e-02}$	$1.30e-01_{4.1e-02}$	$1.27e-01_{4.5e-02}$	$4.45e-02_{6.6e-03}$
	$\mathcal{A}_4\mathcal{I}_1$	$1.36e-01_{3.5e-02}$	$1.42e-01_{5.1e-02}$	$1.48e-01_{4.6e-02}$	$1.43e-01_{4.3e-02}$	$4.85e-02_{6.9e-03}$
	$\mathcal{A}_4\mathcal{I}_2$	$1.43e-01_{3.2e-02}$	$1.51e-01_{4.8e-02}$	$1.58e-01_{4.1e-02}$	$1.51e-01_{4.0e-02}$	$5.05e-02_{6.4e-03}$
	$\mathcal{A}_4\mathcal{I}_3$	$1.12e-01_{2.9e-02}$	$1.18e-01_{4.0e-02}$	$1.25e-01_{3.5e-02}$	$1.20e-01_{3.5e-02}$	$4.04e-02_{6.0e-03}$
	$\mathcal{A}_4\mathcal{I}_4$	$9.83e-02_{3.0e-02}$	$1.06e-01_{4.2e-02}$	$1.11e-01_{3.8e-02}$	$1.09e-01_{3.4e-02}$	$3.69e-02_{5.3e-03}$
	$\mathcal{A}_5\mathcal{I}_1$	$1.27e-01_{3.4e-02}$	$1.41e-01_{4.5e-02}$	$1.40e-01_{3.9e-02}$	$1.38e-01_{3.7e-02}$	$4.28e-02_{7.0e-03}$
	$\mathcal{A}_5\mathcal{I}_2$	$1.27e-01_{3.1e-02}$	$1.42e-01_{4.5e-02}$	$1.47e-01_{4.1e-02}$	$1.41e-01_{4.3e-02}$	$4.35e-02_{7.5e-03}$
	$\mathcal{A}_5\mathcal{I}_3$	$1.33e-01_{3.5e-02}$	$1.47e-01_{5.1e-02}$	$1.46e-01_{4.0e-02}$	$1.38e-01_{4.0e-02}$	$4.47e-02_{8.7e-03}$
	$\mathcal{A}_5\mathcal{I}_4$	$1.37e-01_{3.4e-02}$	$1.55e-01_{4.7e-02}$	$1.61e-01_{4.7e-02}$	$1.59e-01_{4.7e-02}$	$4.81e-02_{7.8e-03}$

Table A.3: Mean and Median Results for I_{IGD}

		MDPSO	NSGA-II	NSGA-III	MOCell	SPEA2
Mean and Standard Deviation	$\mathcal{A}_1\mathcal{I}_1$	$5.62e-01_{5.0e-02}$	$5.34e-01_{2.6e-02}$	$5.33e-01_{3.0e-02}$	$5.57e-01_{4.4e-02}$	$5.25e-01_{1.9e-02}$
	$\mathcal{A}_1\mathcal{I}_2$	$5.47e-01_{4.9e-02}$	$5.21e-01_{2.4e-02}$	$5.21e-01_{3.5e-02}$	$5.42e-01_{4.7e-02}$	$5.15e-01_{1.6e-02}$
	$\mathcal{A}_1\mathcal{I}_3$	$2.26e-01_{1.6e-02}$	$2.15e-01_{9.7e-03}$	$2.14e-01_{7.6e-03}$	$2.25e-01_{1.5e-02}$	$2.11e-01_{5.7e-03}$
	$\mathcal{A}_1\mathcal{I}_4$	$5.91e-01_{5.0e-02}$	$5.66e-01_{2.9e-02}$	$5.67e-01_{2.6e-02}$	$5.88e-01_{5.2e-02}$	$5.58e-01_{2.0e-02}$
	$\mathcal{A}_2\mathcal{I}_1$	$1.14e-01_{9.4e-03}$	$1.17e-01_{5.7e-03}$	$1.18e-01_{6.8e-03}$	$1.20e-01_{8.7e-03}$	$1.16e-01_{8.2e-03}$
	$\mathcal{A}_2\mathcal{I}_2$	$9.13e-02_{5.3e-03}$	$9.35e-02_{5.0e-03}$	$9.48e-02_{4.7e-03}$	$9.61e-02_{7.4e-03}$	$9.36e-02_{4.7e-03}$
	$\mathcal{A}_2\mathcal{I}_3$	$1.27e-01_{6.9e-03}$	$1.30e-01_{8.0e-03}$	$1.32e-01_{7.6e-03}$	$1.35e-01_{9.3e-03}$	$1.30e-01_{6.8e-03}$
	$\mathcal{A}_2\mathcal{I}_4$	$1.45e-01_{7.4e-03}$	$1.48e-01_{1.0e-02}$	$1.50e-01_{8.2e-03}$	$1.53e-01_{1.1e-02}$	$1.49e-01_{7.5e-03}$
	$\mathcal{A}_3\mathcal{I}_1$	$1.17e-01_{5.9e-03}$	$1.20e-01_{8.3e-03}$	$1.22e-01_{6.6e-03}$	$1.23e-01_{8.5e-03}$	$1.20e-01_{6.1e-03}$
	$\mathcal{A}_3\mathcal{I}_2$	$7.58e-02_{4.1e-03}$	$7.82e-02_{4.5e-03}$	$7.92e-02_{3.7e-03}$	$8.00e-02_{6.3e-03}$	$7.85e-02_{4.1e-03}$
	$\mathcal{A}_3\mathcal{I}_3$	$9.61e-02_{5.1e-03}$	$9.88e-02_{6.1e-03}$	$9.97e-02_{5.9e-03}$	$1.01e-01_{6.3e-03}$	$9.95e-02_{4.6e-03}$
	$\mathcal{A}_3\mathcal{I}_4$	$8.86e-02_{5.2e-03}$	$9.09e-02_{5.4e-03}$	$9.19e-02_{5.5e-03}$	$9.31e-02_{6.1e-03}$	$9.13e-02_{5.2e-03}$
	$\mathcal{A}_4\mathcal{I}_1$	$8.02e-02_{5.1e-03}$	$8.22e-02_{5.1e-03}$	$8.32e-02_{5.7e-03}$	$8.42e-02_{5.2e-03}$	$8.21e-02_{5.4e-03}$
	$\mathcal{A}_4\mathcal{I}_2$	$9.23e-02_{5.3e-03}$	$9.50e-02_{5.9e-03}$	$9.62e-02_{5.4e-03}$	$9.77e-02_{6.3e-03}$	$9.46e-02_{5.7e-03}$
	$\mathcal{A}_4\mathcal{I}_3$	$1.04e-01_{6.6e-03}$	$1.07e-01_{6.9e-03}$	$1.08e-01_{6.5e-03}$	$1.10e-01_{6.0e-03}$	$1.07e-01_{6.0e-03}$
	$\mathcal{A}_4\mathcal{I}_4$	$6.90e-02_{4.5e-03}$	$7.14e-02_{4.0e-03}$	$7.23e-02_{4.2e-03}$	$7.33e-02_{4.5e-03}$	$7.12e-02_{2.9e-03}$
	$\mathcal{A}_5\mathcal{I}_1$	$1.08e-01_{6.7e-03}$	$1.12e-01_{6.9e-03}$	$1.13e-01_{7.0e-03}$	$1.15e-01_{6.3e-03}$	$1.12e-01_{5.2e-03}$
	$\mathcal{A}_5\mathcal{I}_2$	$8.48e-02_{5.2e-03}$	$8.73e-02_{6.2e-03}$	$8.84e-02_{4.2e-03}$	$8.94e-02_{6.1e-03}$	$8.77e-02_{4.7e-03}$
	$\mathcal{A}_5\mathcal{I}_3$	$7.21e-02_{3.5e-03}$	$7.41e-02_{4.9e-03}$	$7.51e-02_{4.4e-03}$	$7.62e-02_{4.6e-03}$	$7.42e-02_{4.2e-03}$
	$\mathcal{A}_5\mathcal{I}_4$	$1.08e-01_{6.6e-03}$	$1.11e-01_{7.6e-03}$	$1.12e-01_{5.7e-03}$	$1.14e-01_{6.6e-03}$	$1.12e-01_{4.8e-03}$
Median and Interquartile Range	$\mathcal{A}_1\mathcal{I}_1$	$5.54e-01_{6.4e-02}$	$5.29e-01_{8.7e-03}$	$5.28e-01_{1.4e-02}$	$5.44e-01_{4.6e-02}$	$5.25e-01_{3.8e-03}$
	$\mathcal{A}_1\mathcal{I}_2$	$5.38e-01_{5.8e-02}$	$5.16e-01_{5.2e-03}$	$5.16e-01_{1.7e-02}$	$5.32e-01_{4.9e-02}$	$5.14e-01_{3.7e-03}$
	$\mathcal{A}_1\mathcal{I}_3$	$2.26e-01_{2.7e-02}$	$2.13e-01_{6.4e-03}$	$2.12e-01_{3.8e-03}$	$2.21e-01_{1.5e-02}$	$2.11e-01_{1.3e-03}$
	$\mathcal{A}_1\mathcal{I}_4$	$5.82e-01_{6.2e-02}$	$5.61e-01_{7.6e-03}$	$5.61e-01_{1.8e-02}$	$5.70e-01_{4.6e-02}$	$5.58e-01_{4.7e-03}$
	$\mathcal{A}_2\mathcal{I}_1$	$1.15e-01_{6.6e-03}$	$1.17e-01_{4.2e-03}$	$1.18e-01_{4.0e-03}$	$1.19e-01_{4.9e-03}$	$1.17e-01_{3.5e-03}$
	$\mathcal{A}_2\mathcal{I}_2$	$9.14e-02_{5.2e-03}$	$9.36e-02_{3.5e-03}$	$9.45e-02_{3.9e-03}$	$9.50e-02_{4.6e-03}$	$9.36e-02_{3.1e-03}$
	$\mathcal{A}_2\mathcal{I}_3$	$1.27e-01_{7.0e-03}$	$1.30e-01_{5.2e-03}$	$1.32e-01_{5.2e-03}$	$1.33e-01_{5.9e-03}$	$1.30e-01_{4.0e-03}$
	$\mathcal{A}_2\mathcal{I}_4$	$1.45e-01_{8.2e-03}$	$1.49e-01_{4.8e-03}$	$1.50e-01_{4.7e-03}$	$1.51e-01_{6.9e-03}$	$1.49e-01_{4.4e-03}$
	$\mathcal{A}_3\mathcal{I}_1$	$1.17e-01_{5.6e-03}$	$1.20e-01_{5.3e-03}$	$1.22e-01_{6.0e-03}$	$1.22e-01_{7.3e-03}$	$1.20e-01_{5.5e-03}$
	$\mathcal{A}_3\mathcal{I}_2$	$7.58e-02_{3.7e-03}$	$7.82e-02_{3.6e-03}$	$7.90e-02_{4.0e-03}$	$7.96e-02_{5.6e-03}$	$7.84e-02_{3.8e-03}$
	$\mathcal{A}_3\mathcal{I}_3$	$9.61e-02_{5.0e-03}$	$9.90e-02_{4.6e-03}$	$9.98e-02_{4.8e-03}$	$1.01e-01_{5.8e-03}$	$9.93e-02_{5.1e-03}$
	$\mathcal{A}_3\mathcal{I}_4$	$8.84e-02_{5.5e-03}$	$9.08e-02_{5.4e-03}$	$9.17e-02_{5.4e-03}$	$9.30e-02_{6.7e-03}$	$9.12e-02_{5.1e-03}$
	$\mathcal{A}_4\mathcal{I}_1$	$8.01e-02_{5.9e-03}$	$8.19e-02_{6.1e-03}$	$8.32e-02_{6.3e-03}$	$8.39e-02_{6.5e-03}$	$8.19e-02_{5.8e-03}$
	$\mathcal{A}_4\mathcal{I}_2$	$9.25e-02_{4.5e-03}$	$9.51e-02_{4.3e-03}$	$9.60e-02_{4.7e-03}$	$9.69e-02_{5.9e-03}$	$9.48e-02_{4.2e-03}$
	$\mathcal{A}_4\mathcal{I}_3$	$1.04e-01_{3.1e-03}$	$1.07e-01_{3.1e-03}$	$1.08e-01_{3.0e-03}$	$1.09e-01_{5.9e-03}$	$1.07e-01_{3.2e-03}$
	$\mathcal{A}_4\mathcal{I}_4$	$6.92e-02_{3.4e-03}$	$7.12e-02_{3.5e-03}$	$7.21e-02_{3.8e-03}$	$7.28e-02_{5.0e-03}$	$7.10e-02_{3.1e-03}$
	$\mathcal{A}_5\mathcal{I}_1$	$1.09e-01_{3.4e-03}$	$1.12e-01_{4.2e-03}$	$1.13e-01_{4.2e-03}$	$1.15e-01_{7.0e-03}$	$1.12e-01_{4.0e-03}$
	$\mathcal{A}_5\mathcal{I}_2$	$8.51e-02_{3.1e-03}$	$8.77e-02_{3.2e-03}$	$8.84e-02_{3.2e-03}$	$8.96e-02_{4.6e-03}$	$8.78e-02_{3.4e-03}$
	$\mathcal{A}_5\mathcal{I}_3$	$7.19e-02_{4.0e-03}$	$7.43e-02_{4.5e-03}$	$7.51e-02_{4.6e-03}$	$7.61e-02_{5.6e-03}$	$7.41e-02_{4.4e-03}$
	$\mathcal{A}_5\mathcal{I}_4$	$1.08e-01_{3.0e-03}$	$1.11e-01_{3.6e-03}$	$1.12e-01_{3.7e-03}$	$1.14e-01_{6.2e-03}$	$1.12e-01_{3.7e-03}$

Table A.4: Mean and Median Results for I_{IGD+}

		MDPSO	NSGA-II	NSGA-III	MOCell	SPEA2
Mean and Standard Deviation	$\mathcal{A}_1\mathcal{I}_1$	$1.85e+00_{1.8e-01}$	$1.76e+00_{9.6e-02}$	$1.76e+00_{1.1e-01}$	$1.84e+00_{1.5e-01}$	$1.73e+00_{7.5e-02}$
	$\mathcal{A}_1\mathcal{I}_2$	$1.88e+00_{1.8e-01}$	$1.80e+00_{9.2e-02}$	$1.79e+00_{1.4e-01}$	$1.87e+00_{1.8e-01}$	$1.77e+00_{6.4e-02}$
	$\mathcal{A}_1\mathcal{I}_3$	$7.99e-01_{6.5e-02}$	$7.59e-01_{4.4e-02}$	$7.52e-01_{3.7e-02}$	$7.90e-01_{5.4e-02}$	$7.43e-01_{3.1e-02}$
	$\mathcal{A}_1\mathcal{I}_4$	$1.86e+00_{1.7e-01}$	$1.78e+00_{1.0e-01}$	$1.78e+00_{9.0e-02}$	$1.85e+00_{1.8e-01}$	$1.76e+00_{7.5e-02}$
	$\mathcal{A}_2\mathcal{I}_1$	$5.60e-01_{5.9e-02}$	$5.75e-01_{3.8e-02}$	$5.78e-01_{4.2e-02}$	$5.96e-01_{5.0e-02}$	$5.76e-01_{5.3e-02}$
	$\mathcal{A}_2\mathcal{I}_2$	$5.44e-01_{4.5e-02}$	$5.53e-01_{4.5e-02}$	$5.62e-01_{3.8e-02}$	$5.73e-01_{5.2e-02}$	$5.62e-01_{4.3e-02}$
	$\mathcal{A}_2\mathcal{I}_3$	$6.22e-01_{4.4e-02}$	$6.32e-01_{4.8e-02}$	$6.40e-01_{4.5e-02}$	$6.55e-01_{4.9e-02}$	$6.39e-01_{4.4e-02}$
	$\mathcal{A}_2\mathcal{I}_4$	$6.49e-01_{4.4e-02}$	$6.59e-01_{5.6e-02}$	$6.68e-01_{4.4e-02}$	$6.86e-01_{5.1e-02}$	$6.70e-01_{4.3e-02}$
	$\mathcal{A}_3\mathcal{I}_1$	$6.35e-01_{5.3e-02}$	$6.45e-01_{6.5e-02}$	$6.58e-01_{5.4e-02}$	$6.59e-01_{5.7e-02}$	$6.52e-01_{5.4e-02}$
	$\mathcal{A}_3\mathcal{I}_2$	$4.80e-01_{5.1e-02}$	$4.93e-01_{5.2e-02}$	$5.02e-01_{4.9e-02}$	$5.06e-01_{5.5e-02}$	$5.01e-01_{4.9e-02}$
	$\mathcal{A}_3\mathcal{I}_3$	$5.69e-01_{5.2e-02}$	$5.81e-01_{5.6e-02}$	$5.88e-01_{5.4e-02}$	$5.97e-01_{5.2e-02}$	$5.90e-01_{4.9e-02}$
	$\mathcal{A}_3\mathcal{I}_4$	$5.08e-01_{5.6e-02}$	$5.17e-01_{5.4e-02}$	$5.25e-01_{5.5e-02}$	$5.32e-01_{5.5e-02}$	$5.24e-01_{5.3e-02}$
	$\mathcal{A}_4\mathcal{I}_1$	$4.66e-01_{6.2e-02}$	$4.74e-01_{5.9e-02}$	$4.82e-01_{6.1e-02}$	$4.88e-01_{5.5e-02}$	$4.78e-01_{6.0e-02}$
	$\mathcal{A}_4\mathcal{I}_2$	$5.72e-01_{5.3e-02}$	$5.84e-01_{5.2e-02}$	$5.92e-01_{4.8e-02}$	$5.97e-01_{4.5e-02}$	$5.90e-01_{5.2e-02}$
	$\mathcal{A}_4\mathcal{I}_3$	$6.26e-01_{4.8e-02}$	$6.44e-01_{4.8e-02}$	$6.52e-01_{4.7e-02}$	$6.63e-01_{3.5e-02}$	$6.54e-01_{4.5e-02}$
	$\mathcal{A}_4\mathcal{I}_4$	$4.49e-01_{5.2e-02}$	$4.60e-01_{4.8e-02}$	$4.68e-01_{4.8e-02}$	$4.74e-01_{4.6e-02}$	$4.67e-01_{4.3e-02}$
	$\mathcal{A}_5\mathcal{I}_1$	$6.64e-01_{5.6e-02}$	$6.77e-01_{6.1e-02}$	$6.85e-01_{5.8e-02}$	$6.94e-01_{4.2e-02}$	$6.89e-01_{4.5e-02}$
	$\mathcal{A}_5\mathcal{I}_2$	$5.57e-01_{5.0e-02}$	$5.69e-01_{5.3e-02}$	$5.76e-01_{4.4e-02}$	$5.79e-01_{5.2e-02}$	$5.76e-01_{4.5e-02}$
	$\mathcal{A}_5\mathcal{I}_3$	$4.99e-01_{5.2e-02}$	$5.07e-01_{6.0e-02}$	$5.17e-01_{5.6e-02}$	$5.18e-01_{5.4e-02}$	$5.13e-01_{5.5e-02}$
	$\mathcal{A}_5\mathcal{I}_4$	$6.73e-01_{5.4e-02}$	$6.90e-01_{5.8e-02}$	$6.97e-01_{4.9e-02}$	$7.04e-01_{4.6e-02}$	$7.01e-01_{4.0e-02}$
Median and Interquartile Range	$\mathcal{A}_1\mathcal{I}_1$	$1.83e+00_{2.1e-01}$	$1.75e+00_{3.0e-02}$	$1.74e+00_{4.5e-02}$	$1.79e+00_{1.4e-01}$	$1.73e+00_{1.3e-02}$
	$\mathcal{A}_1\mathcal{I}_2$	$1.85e+00_{2.0e-01}$	$1.78e+00_{1.8e-02}$	$1.78e+00_{5.7e-02}$	$1.83e+00_{1.6e-01}$	$1.77e+00_{1.3e-02}$
	$\mathcal{A}_1\mathcal{I}_3$	$7.98e-01_{9.6e-02}$	$7.52e-01_{2.3e-02}$	$7.47e-01_{1.4e-02}$	$7.79e-01_{5.1e-02}$	$7.43e-01_{5.0e-03}$
	$\mathcal{A}_1\mathcal{I}_4$	$1.83e+00_{1.9e-01}$	$1.77e+00_{2.5e-02}$	$1.77e+00_{5.4e-02}$	$1.80e+00_{1.4e-01}$	$1.76e+00_{1.5e-02}$
	$\mathcal{A}_2\mathcal{I}_1$	$5.65e-01_{4.9e-02}$	$5.75e-01_{4.0e-02}$	$5.78e-01_{4.1e-02}$	$5.93e-01_{4.7e-02}$	$5.77e-01_{3.8e-02}$
	$\mathcal{A}_2\mathcal{I}_2$	$5.46e-01_{5.2e-02}$	$5.53e-01_{4.9e-02}$	$5.60e-01_{4.7e-02}$	$5.70e-01_{5.3e-02}$	$5.61e-01_{4.8e-02}$
	$\mathcal{A}_2\mathcal{I}_3$	$6.23e-01_{4.7e-02}$	$6.35e-01_{4.3e-02}$	$6.41e-01_{4.2e-02}$	$6.53e-01_{4.5e-02}$	$6.39e-01_{4.3e-02}$
	$\mathcal{A}_2\mathcal{I}_4$	$6.51e-01_{4.5e-02}$	$6.63e-01_{3.6e-02}$	$6.69e-01_{3.2e-02}$	$6.82e-01_{4.2e-02}$	$6.71e-01_{3.6e-02}$
	$\mathcal{A}_3\mathcal{I}_1$	$6.40e-01_{6.7e-02}$	$6.53e-01_{6.1e-02}$	$6.63e-01_{6.6e-02}$	$6.63e-01_{6.5e-02}$	$6.56e-01_{6.4e-02}$
	$\mathcal{A}_3\mathcal{I}_2$	$4.84e-01_{6.4e-02}$	$4.95e-01_{6.9e-02}$	$5.04e-01_{6.6e-02}$	$5.07e-01_{6.3e-02}$	$5.02e-01_{6.2e-02}$
	$\mathcal{A}_3\mathcal{I}_3$	$5.73e-01_{6.5e-02}$	$5.86e-01_{6.3e-02}$	$5.92e-01_{6.4e-02}$	$5.99e-01_{5.8e-02}$	$5.92e-01_{6.4e-02}$
	$\mathcal{A}_3\mathcal{I}_4$	$5.10e-01_{7.2e-02}$	$5.21e-01_{7.0e-02}$	$5.28e-01_{7.0e-02}$	$5.34e-01_{7.0e-02}$	$5.27e-01_{6.4e-02}$
	$\mathcal{A}_4\mathcal{I}_1$	$4.69e-01_{7.8e-02}$	$4.76e-01_{8.0e-02}$	$4.88e-01_{8.1e-02}$	$4.90e-01_{7.3e-02}$	$4.82e-01_{8.0e-02}$
	$\mathcal{A}_4\mathcal{I}_2$	$5.78e-01_{5.6e-02}$	$5.89e-01_{5.2e-02}$	$5.95e-01_{5.4e-02}$	$6.02e-01_{4.9e-02}$	$5.94e-01_{5.3e-02}$
	$\mathcal{A}_4\mathcal{I}_3$	$6.30e-01_{3.3e-02}$	$6.48e-01_{3.3e-02}$	$6.56e-01_{3.1e-02}$	$6.62e-01_{3.8e-02}$	$6.57e-01_{3.3e-02}$
	$\mathcal{A}_4\mathcal{I}_4$	$4.54e-01_{6.3e-02}$	$4.63e-01_{6.0e-02}$	$4.71e-01_{6.3e-02}$	$4.75e-01_{5.9e-02}$	$4.69e-01_{5.7e-02}$
	$\mathcal{A}_5\mathcal{I}_1$	$6.69e-01_{4.3e-02}$	$6.84e-01_{4.8e-02}$	$6.91e-01_{4.6e-02}$	$6.97e-01_{4.2e-02}$	$6.95e-01_{4.9e-02}$
	$\mathcal{A}_5\mathcal{I}_2$	$5.62e-01_{5.3e-02}$	$5.75e-01_{5.0e-02}$	$5.80e-01_{5.1e-02}$	$5.85e-01_{5.0e-02}$	$5.80e-01_{4.9e-02}$
	$\mathcal{A}_5\mathcal{I}_3$	$5.03e-01_{6.7e-02}$	$5.13e-01_{7.0e-02}$	$5.21e-01_{7.3e-02}$	$5.20e-01_{6.9e-02}$	$5.14e-01_{7.2e-02}$
	$\mathcal{A}_5\mathcal{I}_4$	$6.80e-01_{4.1e-02}$	$6.95e-01_{3.7e-02}$	$7.01e-01_{4.2e-02}$	$7.07e-01_{4.2e-02}$	$7.03e-01_{4.1e-02}$

Table A.5: Mean and Median Results for I_Δ

		MDPSO	NSGA-II	NSGA-III	MOCell	SPEA2
Mean and Standard Deviation	$\mathcal{A}_1\mathcal{I}_1$	$8.71e-01_{7.0e-02}$	$8.09e-01_{1.4e-01}$	$8.62e-01_{5.4e-02}$	$8.49e-01_{5.5e-02}$	$1.06e+00_{4.4e-02}$
	$\mathcal{A}_1\mathcal{I}_2$	$8.73e-01_{7.6e-02}$	$8.04e-01_{1.5e-01}$	$8.62e-01_{5.6e-02}$	$8.53e-01_{5.7e-02}$	$1.06e+00_{4.5e-02}$
	$\mathcal{A}_1\mathcal{I}_3$	$8.22e-01_{1.1e-01}$	$7.43e-01_{1.7e-01}$	$7.76e-01_{9.0e-02}$	$7.78e-01_{9.1e-02}$	$1.11e+00_{6.3e-02}$
	$\mathcal{A}_1\mathcal{I}_4$	$8.68e-01_{7.3e-02}$	$7.98e-01_{1.5e-01}$	$8.54e-01_{5.4e-02}$	$8.50e-01_{5.6e-02}$	$1.06e+00_{4.3e-02}$
	$\mathcal{A}_2\mathcal{I}_1$	$7.96e-01_{6.8e-02}$	$7.77e-01_{9.5e-02}$	$8.12e-01_{7.2e-02}$	$7.99e-01_{7.5e-02}$	$1.01e+00_{4.5e-02}$
	$\mathcal{A}_2\mathcal{I}_2$	$7.86e-01_{7.1e-02}$	$7.74e-01_{1.0e-01}$	$8.07e-01_{7.6e-02}$	$7.94e-01_{7.7e-02}$	$1.01e+00_{4.3e-02}$
	$\mathcal{A}_2\mathcal{I}_3$	$7.82e-01_{7.1e-02}$	$7.64e-01_{1.0e-01}$	$8.00e-01_{8.1e-02}$	$7.90e-01_{7.6e-02}$	$1.02e+00_{4.6e-02}$
	$\mathcal{A}_2\mathcal{I}_4$	$7.92e-01_{6.8e-02}$	$7.80e-01_{9.9e-02}$	$8.14e-01_{7.5e-02}$	$8.06e-01_{7.7e-02}$	$1.01e+00_{4.3e-02}$
	$\mathcal{A}_3\mathcal{I}_1$	$7.96e-01_{6.4e-02}$	$7.91e-01_{8.5e-02}$	$8.14e-01_{7.2e-02}$	$8.06e-01_{7.0e-02}$	$1.05e+00_{4.8e-02}$
	$\mathcal{A}_3\mathcal{I}_2$	$7.86e-01_{6.7e-02}$	$7.85e-01_{8.4e-02}$	$8.07e-01_{7.1e-02}$	$8.04e-01_{7.6e-02}$	$1.05e+00_{5.1e-02}$
	$\mathcal{A}_3\mathcal{I}_3$	$7.86e-01_{6.5e-02}$	$7.83e-01_{8.5e-02}$	$8.00e-01_{7.6e-02}$	$7.98e-01_{7.0e-02}$	$1.05e+00_{4.9e-02}$
	$\mathcal{A}_3\mathcal{I}_4$	$7.80e-01_{7.2e-02}$	$7.79e-01_{9.1e-02}$	$8.04e-01_{7.6e-02}$	$7.97e-01_{7.5e-02}$	$1.06e+00_{5.3e-02}$
	$\mathcal{A}_4\mathcal{I}_1$	$8.02e-01_{6.7e-02}$	$8.05e-01_{7.8e-02}$	$8.20e-01_{7.2e-02}$	$8.20e-01_{6.6e-02}$	$1.06e+00_{4.4e-02}$
	$\mathcal{A}_4\mathcal{I}_2$	$8.07e-01_{6.5e-02}$	$8.12e-01_{7.9e-02}$	$8.29e-01_{6.9e-02}$	$8.26e-01_{6.8e-02}$	$1.06e+00_{4.6e-02}$
	$\mathcal{A}_4\mathcal{I}_3$	$8.18e-01_{6.3e-02}$	$8.18e-01_{7.8e-02}$	$8.36e-01_{6.8e-02}$	$8.32e-01_{6.4e-02}$	$1.05e+00_{4.4e-02}$
	$\mathcal{A}_4\mathcal{I}_4$	$8.12e-01_{6.2e-02}$	$8.19e-01_{7.3e-02}$	$8.33e-01_{6.8e-02}$	$8.28e-01_{6.7e-02}$	$1.06e+00_{4.4e-02}$
	$\mathcal{A}_5\mathcal{I}_1$	$8.25e-01_{5.9e-02}$	$8.39e-01_{7.3e-02}$	$8.48e-01_{6.5e-02}$	$8.53e-01_{6.7e-02}$	$1.08e+00_{4.5e-02}$
	$\mathcal{A}_5\mathcal{I}_2$	$8.29e-01_{5.8e-02}$	$8.43e-01_{7.1e-02}$	$8.46e-01_{6.8e-02}$	$8.53e-01_{6.8e-02}$	$1.08e+00_{4.5e-02}$
	$\mathcal{A}_5\mathcal{I}_3$	$8.14e-01_{6.4e-02}$	$8.23e-01_{7.9e-02}$	$8.35e-01_{7.1e-02}$	$8.40e-01_{7.0e-02}$	$1.09e+00_{4.7e-02}$
	$\mathcal{A}_5\mathcal{I}_4$	$8.16e-01_{6.6e-02}$	$8.33e-01_{7.6e-02}$	$8.44e-01_{6.9e-02}$	$8.48e-01_{7.3e-02}$	$1.09e+00_{4.8e-02}$
Median and Interquartile Range	$\mathcal{A}_1\mathcal{I}_1$	$8.63e-01_{1.0e-01}$	$8.31e-01_{1.2e-01}$	$8.59e-01_{7.5e-02}$	$8.50e-01_{7.4e-02}$	$1.06e+00_{5.3e-02}$
	$\mathcal{A}_1\mathcal{I}_2$	$8.62e-01_{1.0e-01}$	$8.35e-01_{1.3e-01}$	$8.57e-01_{7.5e-02}$	$8.52e-01_{7.8e-02}$	$1.06e+00_{4.9e-02}$
	$\mathcal{A}_1\mathcal{I}_3$	$8.11e-01_{1.5e-01}$	$7.59e-01_{1.8e-01}$	$7.68e-01_{1.2e-01}$	$7.80e-01_{1.2e-01}$	$1.11e+00_{7.4e-02}$
	$\mathcal{A}_1\mathcal{I}_4$	$8.60e-01_{9.9e-02}$	$8.30e-01_{1.2e-01}$	$8.50e-01_{7.0e-02}$	$8.47e-01_{7.0e-02}$	$1.06e+00_{5.0e-02}$
	$\mathcal{A}_2\mathcal{I}_1$	$7.96e-01_{9.4e-02}$	$7.83e-01_{1.2e-01}$	$8.14e-01_{9.5e-02}$	$8.01e-01_{9.6e-02}$	$1.01e+00_{5.4e-02}$
	$\mathcal{A}_2\mathcal{I}_2$	$7.87e-01_{9.5e-02}$	$7.79e-01_{1.3e-01}$	$8.11e-01_{1.0e-01}$	$7.92e-01_{1.0e-01}$	$1.01e+00_{5.4e-02}$
	$\mathcal{A}_2\mathcal{I}_3$	$7.83e-01_{9.2e-02}$	$7.66e-01_{1.3e-01}$	$8.03e-01_{1.0e-01}$	$7.88e-01_{9.9e-02}$	$1.02e+00_{6.2e-02}$
	$\mathcal{A}_2\mathcal{I}_4$	$7.92e-01_{9.2e-02}$	$7.83e-01_{1.2e-01}$	$8.14e-01_{1.0e-01}$	$8.08e-01_{1.1e-01}$	$1.01e+00_{5.9e-02}$
	$\mathcal{A}_3\mathcal{I}_1$	$7.96e-01_{8.7e-02}$	$7.93e-01_{1.1e-01}$	$8.20e-01_{9.9e-02}$	$8.05e-01_{9.2e-02}$	$1.05e+00_{6.6e-02}$
	$\mathcal{A}_3\mathcal{I}_2$	$7.87e-01_{9.0e-02}$	$7.90e-01_{1.1e-01}$	$8.09e-01_{9.4e-02}$	$8.05e-01_{1.1e-01}$	$1.05e+00_{7.0e-02}$
	$\mathcal{A}_3\mathcal{I}_3$	$7.84e-01_{8.6e-02}$	$7.88e-01_{1.1e-01}$	$8.00e-01_{9.9e-02}$	$7.95e-01_{9.1e-02}$	$1.05e+00_{6.8e-02}$
	$\mathcal{A}_3\mathcal{I}_4$	$7.79e-01_{9.4e-02}$	$7.83e-01_{1.2e-01}$	$8.04e-01_{9.9e-02}$	$7.96e-01_{9.6e-02}$	$1.06e+00_{7.3e-02}$
	$\mathcal{A}_4\mathcal{I}_1$	$8.01e-01_{8.2e-02}$	$8.09e-01_{1.0e-01}$	$8.22e-01_{9.4e-02}$	$8.15e-01_{8.9e-02}$	$1.06e+00_{6.1e-02}$
	$\mathcal{A}_4\mathcal{I}_2$	$8.09e-01_{8.4e-02}$	$8.17e-01_{9.9e-02}$	$8.31e-01_{9.4e-02}$	$8.25e-01_{8.8e-02}$	$1.06e+00_{5.9e-02}$
	$\mathcal{A}_4\mathcal{I}_3$	$8.19e-01_{8.0e-02}$	$8.23e-01_{9.7e-02}$	$8.36e-01_{8.5e-02}$	$8.33e-01_{8.4e-02}$	$1.05e+00_{6.0e-02}$
	$\mathcal{A}_4\mathcal{I}_4$	$8.10e-01_{8.2e-02}$	$8.22e-01_{9.6e-02}$	$8.33e-01_{9.4e-02}$	$8.31e-01_{8.7e-02}$	$1.06e+00_{6.1e-02}$
	$\mathcal{A}_5\mathcal{I}_1$	$8.29e-01_{7.7e-02}$	$8.43e-01_{9.3e-02}$	$8.52e-01_{8.5e-02}$	$8.55e-01_{8.7e-02}$	$1.08e+00_{6.3e-02}$
	$\mathcal{A}_5\mathcal{I}_2$	$8.31e-01_{7.9e-02}$	$8.48e-01_{9.1e-02}$	$8.51e-01_{9.2e-02}$	$8.57e-01_{8.8e-02}$	$1.08e+00_{5.9e-02}$
	$\mathcal{A}_5\mathcal{I}_3$	$8.13e-01_{8.6e-02}$	$8.25e-01_{9.8e-02}$	$8.36e-01_{9.6e-02}$	$8.43e-01_{9.1e-02}$	$1.09e+00_{6.2e-02}$
	$\mathcal{A}_5\mathcal{I}_4$	$8.20e-01_{8.8e-02}$	$8.37e-01_{9.7e-02}$	$8.45e-01_{8.7e-02}$	$8.51e-01_{9.5e-02}$	$1.09e+00_{6.3e-02}$

Table A.6: Mean and Median Results for I_{HV}

		MDPSO	NSGA-II	NSGA-III	MOCell	SPEA2
Mean and Standard Deviation	$\mathcal{A}_1\mathcal{I}_1$	$5.64e-04_{8.0e-03}$	$2.56e-04_{5.1e-03}$	$2.42e-04_{3.9e-03}$	$2.83e-04_{5.7e-03}$	$1.72e-04_{4.3e-03}$
	$\mathcal{A}_1\mathcal{I}_2$	$4.13e-04_{6.9e-03}$	$1.37e-04_{3.2e-03}$	$4.35e-04_{6.8e-03}$	$8.96e-04_{1.2e-02}$	$7.15e-05_{2.3e-03}$
	$\mathcal{A}_1\mathcal{I}_3$	$7.09e-03_{1.6e-02}$	$9.40e-03_{1.1e-02}$	$1.14e-02_{1.4e-02}$	$5.50e-03_{7.1e-03}$	$1.28e-02_{1.2e-02}$
	$\mathcal{A}_1\mathcal{I}_4$	$2.63e-04_{4.9e-03}$	$3.79e-04_{8.1e-03}$	$2.29e-04_{5.2e-03}$	$8.77e-04_{1.3e-02}$	$1.78e-04_{4.1e-03}$
	$\mathcal{A}_2\mathcal{I}_1$	$3.07e-02_{3.8e-02}$	$1.86e-02_{1.5e-02}$	$1.56e-02_{1.9e-02}$	$1.23e-02_{2.0e-02}$	$1.60e-02_{3.1e-02}$
	$\mathcal{A}_2\mathcal{I}_2$	$7.05e-02_{2.8e-02}$	$6.17e-02_{2.7e-02}$	$5.57e-02_{1.8e-02}$	$4.64e-02_{2.9e-02}$	$5.49e-02_{2.3e-02}$
	$\mathcal{A}_2\mathcal{I}_3$	$6.67e-02_{2.5e-02}$	$5.90e-02_{2.5e-02}$	$5.36e-02_{2.5e-02}$	$4.23e-02_{2.6e-02}$	$5.33e-02_{2.0e-02}$
	$\mathcal{A}_2\mathcal{I}_4$	$4.06e-02_{2.3e-02}$	$3.25e-02_{2.8e-02}$	$2.65e-02_{2.5e-02}$	$1.98e-02_{2.4e-02}$	$2.53e-02_{2.4e-02}$
	$\mathcal{A}_3\mathcal{I}_1$	$3.64e-02_{1.9e-02}$	$2.71e-02_{3.3e-02}$	$2.22e-02_{1.4e-02}$	$1.75e-02_{2.0e-02}$	$2.07e-02_{1.7e-02}$
	$\mathcal{A}_3\mathcal{I}_2$	$3.41e-02_{2.0e-02}$	$2.28e-02_{2.2e-02}$	$1.90e-02_{2.1e-02}$	$1.62e-02_{2.7e-02}$	$1.71e-02_{1.8e-02}$
	$\mathcal{A}_3\mathcal{I}_3$	$4.32e-02_{2.0e-02}$	$3.16e-02_{2.3e-02}$	$2.84e-02_{2.3e-02}$	$2.16e-02_{2.3e-02}$	$2.47e-02_{1.6e-02}$
	$\mathcal{A}_3\mathcal{I}_4$	$5.44e-02_{1.9e-02}$	$4.31e-02_{2.1e-02}$	$3.94e-02_{2.2e-02}$	$3.15e-02_{2.0e-02}$	$3.70e-02_{2.1e-02}$
	$\mathcal{A}_4\mathcal{I}_1$	$4.15e-02_{3.0e-02}$	$2.99e-02_{1.8e-02}$	$2.73e-02_{2.3e-02}$	$2.12e-02_{1.9e-02}$	$2.56e-02_{2.3e-02}$
	$\mathcal{A}_4\mathcal{I}_2$	$6.29e-02_{2.8e-02}$	$5.18e-02_{2.4e-02}$	$4.76e-02_{2.2e-02}$	$4.13e-02_{1.9e-02}$	$4.70e-02_{2.6e-02}$
	$\mathcal{A}_4\mathcal{I}_3$	$2.73e-02_{2.1e-02}$	$1.63e-02_{2.1e-02}$	$1.29e-02_{2.0e-02}$	$9.38e-03_{1.3e-02}$	$1.19e-02_{2.1e-02}$
	$\mathcal{A}_4\mathcal{I}_4$	$4.32e-02_{2.6e-02}$	$3.01e-02_{2.6e-02}$	$2.58e-02_{2.3e-02}$	$2.02e-02_{2.3e-02}$	$2.26e-02_{1.4e-02}$
	$\mathcal{A}_5\mathcal{I}_1$	$2.67e-02_{2.9e-02}$	$1.72e-02_{3.6e-02}$	$1.29e-02_{3.3e-02}$	$8.83e-03_{1.7e-02}$	$1.03e-02_{1.9e-02}$
	$\mathcal{A}_5\mathcal{I}_2$	$3.42e-02_{3.1e-02}$	$2.26e-02_{3.3e-02}$	$1.69e-02_{2.5e-02}$	$1.47e-02_{3.2e-02}$	$1.53e-02_{2.6e-02}$
	$\mathcal{A}_5\mathcal{I}_3$	$2.44e-02_{1.8e-02}$	$1.54e-02_{3.0e-02}$	$1.10e-02_{2.2e-02}$	$8.75e-03_{2.6e-02}$	$9.24e-03_{2.0e-02}$
	$\mathcal{A}_5\mathcal{I}_4$	$2.76e-02_{2.7e-02}$	$1.66e-02_{2.9e-02}$	$1.16e-02_{2.3e-02}$	$8.70e-03_{2.2e-02}$	$9.32e-03_{1.6e-02}$
Median and Interquartile Range	$\mathcal{A}_1\mathcal{I}_1$	$0.00e+00_{0.0e+00}$	$0.00e+00_{0.0e+00}$	$0.00e+00_{0.0e+00}$	$0.00e+00_{0.0e+00}$	$0.00e+00_{0.0e+00}$
	$\mathcal{A}_1\mathcal{I}_2$	$0.00e+00_{0.0e+00}$	$0.00e+00_{0.0e+00}$	$0.00e+00_{0.0e+00}$	$0.00e+00_{0.0e+00}$	$0.00e+00_{0.0e+00}$
	$\mathcal{A}_1\mathcal{I}_3$	$5.06e-03_{1.3e-02}$	$1.03e-02_{7.4e-03}$	$1.20e-02_{4.6e-03}$	$4.85e-03_{9.4e-03}$	$1.32e-02_{2.5e-03}$
	$\mathcal{A}_1\mathcal{I}_4$	$0.00e+00_{0.0e+00}$	$0.00e+00_{0.0e+00}$	$0.00e+00_{0.0e+00}$	$0.00e+00_{0.0e+00}$	$0.00e+00_{0.0e+00}$
	$\mathcal{A}_2\mathcal{I}_1$	$2.71e-02_{2.7e-02}$	$1.61e-02_{1.4e-02}$	$1.32e-02_{1.1e-02}$	$1.01e-02_{1.3e-02}$	$1.24e-02_{1.1e-02}$
	$\mathcal{A}_2\mathcal{I}_2$	$7.08e-02_{3.2e-02}$	$5.90e-02_{1.8e-02}$	$5.43e-02_{1.5e-02}$	$4.81e-02_{1.8e-02}$	$5.39e-02_{1.5e-02}$
	$\mathcal{A}_2\mathcal{I}_3$	$6.73e-02_{3.1e-02}$	$5.59e-02_{1.8e-02}$	$5.18e-02_{1.4e-02}$	$4.42e-02_{1.7e-02}$	$5.22e-02_{1.5e-02}$
	$\mathcal{A}_2\mathcal{I}_4$	$4.18e-02_{3.0e-02}$	$2.85e-02_{1.6e-02}$	$2.45e-02_{1.3e-02}$	$1.93e-02_{1.5e-02}$	$2.32e-02_{1.4e-02}$
	$\mathcal{A}_3\mathcal{I}_1$	$3.55e-02_{1.3e-02}$	$2.42e-02_{1.0e-02}$	$2.14e-02_{8.5e-03}$	$1.75e-02_{1.5e-02}$	$2.07e-02_{1.0e-02}$
	$\mathcal{A}_3\mathcal{I}_2$	$3.38e-02_{1.2e-02}$	$2.09e-02_{9.7e-03}$	$1.76e-02_{8.6e-03}$	$1.48e-02_{1.4e-02}$	$1.66e-02_{1.1e-02}$
	$\mathcal{A}_3\mathcal{I}_3$	$4.21e-02_{1.4e-02}$	$2.92e-02_{1.0e-02}$	$2.62e-02_{8.7e-03}$	$2.20e-02_{1.6e-02}$	$2.47e-02_{1.1e-02}$
	$\mathcal{A}_3\mathcal{I}_4$	$5.28e-02_{1.6e-02}$	$4.10e-02_{1.1e-02}$	$3.73e-02_{1.1e-02}$	$3.30e-02_{1.6e-02}$	$3.61e-02_{1.2e-02}$
	$\mathcal{A}_4\mathcal{I}_1$	$3.92e-02_{1.2e-02}$	$2.84e-02_{9.5e-03}$	$2.52e-02_{8.8e-03}$	$2.13e-02_{1.4e-02}$	$2.46e-02_{1.1e-02}$
	$\mathcal{A}_4\mathcal{I}_2$	$6.03e-02_{1.5e-02}$	$4.92e-02_{1.2e-02}$	$4.59e-02_{1.1e-02}$	$4.17e-02_{1.6e-02}$	$4.51e-02_{1.3e-02}$
	$\mathcal{A}_4\mathcal{I}_3$	$2.61e-02_{1.0e-02}$	$1.43e-02_{8.5e-03}$	$1.12e-02_{7.0e-03}$	$8.63e-03_{1.3e-02}$	$1.06e-02_{9.5e-03}$
	$\mathcal{A}_4\mathcal{I}_4$	$4.11e-02_{1.2e-02}$	$2.80e-02_{1.0e-02}$	$2.43e-02_{8.6e-03}$	$2.04e-02_{1.7e-02}$	$2.25e-02_{1.1e-02}$
	$\mathcal{A}_5\mathcal{I}_1$	$2.42e-02_{6.6e-03}$	$1.34e-02_{7.7e-03}$	$9.67e-03_{6.7e-03}$	$6.95e-03_{1.2e-02}$	$9.19e-03_{9.6e-03}$
	$\mathcal{A}_5\mathcal{I}_2$	$3.13e-02_{7.6e-03}$	$1.93e-02_{8.4e-03}$	$1.50e-02_{7.4e-03}$	$1.14e-02_{1.5e-02}$	$1.37e-02_{1.2e-02}$
	$\mathcal{A}_5\mathcal{I}_3$	$2.33e-02_{6.6e-03}$	$1.24e-02_{7.9e-03}$	$8.94e-03_{6.2e-03}$	$5.58e-03_{1.1e-02}$	$7.66e-03_{9.6e-03}$
	$\mathcal{A}_5\mathcal{I}_4$	$2.50e-02_{7.1e-03}$	$1.38e-02_{7.5e-03}$	$9.83e-03_{6.5e-03}$	$5.60e-03_{1.1e-02}$	$7.91e-03_{1.0e-02}$

Table A.7: Mean and Median Results for I_{ER}

		MDPSO	NSGA-II	NSGA-III	MOCell	SPEA2
Mean and Standard Deviation	$\mathcal{A}_1\mathcal{I}_1$	$9.97e-01_{4.6e-02}$	$1.00e+00_{8.2e-03}$	$1.00e+00_{4.6e-03}$	$1.00e+00_{3.5e-03}$	$1.00e+00_{0.0e+00}$
	$\mathcal{A}_1\mathcal{I}_2$	$1.00e+00_{6.8e-03}$	$1.00e+00_{3.2e-03}$	$9.99e-01_{3.2e-02}$	$9.99e-01_{1.5e-02}$	$1.00e+00_{0.0e+00}$
	$\mathcal{A}_1\mathcal{I}_3$	$1.00e+00_{8.7e-03}$	$1.00e+00_{6.6e-03}$	$1.00e+00_{8.3e-03}$	$1.00e+00_{1.0e-02}$	$1.00e+00_{4.5e-04}$
	$\mathcal{A}_1\mathcal{I}_4$	$9.98e-01_{3.3e-02}$	$1.00e+00_{3.5e-03}$	$1.00e+00_{6.6e-03}$	$9.99e-01_{1.0e-02}$	$1.00e+00_{0.0e+00}$
	$\mathcal{A}_2\mathcal{I}_1$	$9.98e-01_{2.1e-02}$	$1.00e+00_{7.7e-03}$	$9.99e-01_{1.1e-02}$	$9.99e-01_{1.6e-02}$	$1.00e+00_{1.0e-03}$
	$\mathcal{A}_2\mathcal{I}_2$	$9.98e-01_{1.5e-02}$	$9.98e-01_{2.5e-02}$	$1.00e+00_{6.6e-03}$	$9.99e-01_{2.3e-02}$	$1.00e+00_{1.1e-03}$
	$\mathcal{A}_2\mathcal{I}_3$	$9.99e-01_{1.9e-02}$	$9.99e-01_{1.5e-02}$	$9.99e-01_{1.6e-02}$	$9.99e-01_{1.7e-02}$	$1.00e+00_{3.2e-04}$
	$\mathcal{A}_2\mathcal{I}_4$	$9.99e-01_{1.1e-02}$	$9.99e-01_{1.1e-02}$	$9.99e-01_{1.4e-02}$	$9.99e-01_{1.4e-02}$	$1.00e+00_{5.5e-04}$
	$\mathcal{A}_3\mathcal{I}_1$	$9.98e-01_{2.1e-02}$	$9.98e-01_{3.3e-02}$	$9.99e-01_{1.1e-02}$	$1.00e+00_{7.3e-03}$	$1.00e+00_{6.3e-04}$
	$\mathcal{A}_3\mathcal{I}_2$	$9.97e-01_{2.3e-02}$	$9.98e-01_{2.2e-02}$	$9.99e-01_{1.9e-02}$	$9.99e-01_{1.3e-02}$	$1.00e+00_{1.0e-03}$
	$\mathcal{A}_3\mathcal{I}_3$	$9.98e-01_{1.5e-02}$	$9.98e-01_{2.2e-02}$	$9.99e-01_{1.7e-02}$	$1.00e+00_{6.7e-03}$	$1.00e+00_{1.1e-03}$
	$\mathcal{A}_3\mathcal{I}_4$	$9.98e-01_{1.9e-02}$	$9.99e-01_{2.4e-02}$	$9.99e-01_{1.7e-02}$	$9.99e-01_{2.1e-02}$	$1.00e+00_{9.5e-04}$
	$\mathcal{A}_4\mathcal{I}_1$	$9.96e-01_{3.1e-02}$	$9.99e-01_{1.2e-02}$	$9.99e-01_{7.4e-03}$	$1.00e+00_{6.6e-03}$	$1.00e+00_{8.3e-04}$
	$\mathcal{A}_4\mathcal{I}_2$	$9.97e-01_{2.7e-02}$	$9.98e-01_{1.8e-02}$	$9.99e-01_{1.8e-02}$	$9.99e-01_{7.5e-03}$	$1.00e+00_{1.0e-03}$
	$\mathcal{A}_4\mathcal{I}_3$	$9.98e-01_{1.8e-02}$	$9.99e-01_{1.5e-02}$	$9.98e-01_{3.3e-02}$	$9.99e-01_{1.2e-02}$	$1.00e+00_{1.0e-03}$
	$\mathcal{A}_4\mathcal{I}_4$	$9.95e-01_{3.6e-02}$	$9.99e-01_{1.5e-02}$	$9.99e-01_{1.9e-02}$	$9.99e-01_{1.3e-02}$	$1.00e+00_{8.4e-04}$
	$\mathcal{A}_5\mathcal{I}_1$	$9.98e-01_{2.5e-02}$	$9.97e-01_{3.8e-02}$	$9.99e-01_{2.2e-02}$	$1.00e+00_{5.2e-03}$	$1.00e+00_{5.5e-04}$
	$\mathcal{A}_5\mathcal{I}_2$	$9.98e-01_{2.2e-02}$	$9.99e-01_{1.1e-02}$	$9.99e-01_{1.9e-02}$	$9.98e-01_{2.6e-02}$	$1.00e+00_{1.3e-03}$
	$\mathcal{A}_5\mathcal{I}_3$	$9.96e-01_{3.1e-02}$	$9.98e-01_{2.2e-02}$	$9.99e-01_{1.3e-02}$	$9.98e-01_{2.5e-02}$	$1.00e+00_{8.9e-04}$
	$\mathcal{A}_5\mathcal{I}_4$	$9.97e-01_{3.1e-02}$	$9.99e-01_{1.5e-02}$	$9.99e-01_{9.3e-03}$	$9.99e-01_{2.2e-02}$	$1.00e+00_{1.1e-03}$
Median and Interquartile Range	$\mathcal{A}_1\mathcal{I}_1$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$
	$\mathcal{A}_1\mathcal{I}_2$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$
	$\mathcal{A}_1\mathcal{I}_3$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$
	$\mathcal{A}_1\mathcal{I}_4$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$
	$\mathcal{A}_2\mathcal{I}_1$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$
	$\mathcal{A}_2\mathcal{I}_2$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$
	$\mathcal{A}_2\mathcal{I}_3$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$
	$\mathcal{A}_2\mathcal{I}_4$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$
	$\mathcal{A}_3\mathcal{I}_1$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$
	$\mathcal{A}_3\mathcal{I}_2$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$
	$\mathcal{A}_3\mathcal{I}_3$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$
	$\mathcal{A}_3\mathcal{I}_4$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$
	$\mathcal{A}_4\mathcal{I}_1$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$
	$\mathcal{A}_4\mathcal{I}_2$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$
	$\mathcal{A}_4\mathcal{I}_3$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$
	$\mathcal{A}_4\mathcal{I}_4$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$
	$\mathcal{A}_5\mathcal{I}_1$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$
	$\mathcal{A}_5\mathcal{I}_2$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$
	$\mathcal{A}_5\mathcal{I}_3$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$
	$\mathcal{A}_5\mathcal{I}_4$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$	$1.00e+00_{0.0e+00}$

Table A.8: Mean and Median Results for I_T

		MDPSO	NSGA-II	NSGA-III	MOCell	SPEA2
Mean and Standard Deviation	$\mathcal{A}_1\mathcal{I}_1$	$2.31e+03_{8.0e+03}$	$2.43e+03_{1.0e+04}$	$1.94e+03_{7.7e+03}$	$1.88e+03_{5.6e+03}$	$7.31e+03_{2.1e+04}$
	$\mathcal{A}_1\mathcal{I}_2$	$2.53e+03_{8.9e+03}$	$2.65e+03_{1.2e+04}$	$2.11e+03_{8.8e+03}$	$2.11e+03_{5.8e+03}$	$8.22e+03_{2.5e+04}$
	$\mathcal{A}_1\mathcal{I}_3$	$2.80e+03_{1.0e+04}$	$2.70e+03_{1.2e+04}$	$1.91e+03_{7.6e+03}$	$2.13e+03_{6.2e+03}$	$7.79e+03_{2.3e+04}$
	$\mathcal{A}_1\mathcal{I}_4$	$2.80e+03_{9.1e+03}$	$2.99e+03_{1.4e+04}$	$2.04e+03_{8.2e+03}$	$2.22e+03_{6.0e+03}$	$7.57e+03_{2.1e+04}$
	$\mathcal{A}_2\mathcal{I}_1$	$6.53e+03_{2.3e+04}$	$6.52e+03_{3.0e+04}$	$4.02e+03_{1.6e+04}$	$5.50e+03_{1.7e+04}$	$1.22e+04_{2.9e+04}$
	$\mathcal{A}_2\mathcal{I}_2$	$6.45e+03_{2.1e+04}$	$5.59e+03_{2.2e+04}$	$3.98e+03_{1.4e+04}$	$6.00e+03_{1.8e+04}$	$1.31e+04_{3.3e+04}$
	$\mathcal{A}_2\mathcal{I}_3$	$7.15e+03_{2.5e+04}$	$5.56e+03_{1.7e+04}$	$4.25e+03_{1.5e+04}$	$6.26e+03_{1.7e+04}$	$1.36e+04_{3.4e+04}$
	$\mathcal{A}_2\mathcal{I}_4$	$7.10e+03_{2.4e+04}$	$5.85e+03_{1.9e+04}$	$4.57e+03_{1.6e+04}$	$6.44e+03_{1.8e+04}$	$1.51e+04_{3.9e+04}$
	$\mathcal{A}_3\mathcal{I}_1$	$1.14e+04_{3.1e+04}$	$9.76e+03_{3.4e+04}$	$7.44e+03_{2.9e+04}$	$1.22e+04_{3.5e+04}$	$2.35e+04_{6.0e+04}$
	$\mathcal{A}_3\mathcal{I}_2$	$1.20e+04_{3.7e+04}$	$9.88e+03_{3.1e+04}$	$8.33e+03_{3.7e+04}$	$1.35e+04_{4.0e+04}$	$2.48e+04_{6.0e+04}$
	$\mathcal{A}_3\mathcal{I}_3$	$1.15e+04_{2.8e+04}$	$1.08e+04_{3.6e+04}$	$7.28e+03_{2.1e+04}$	$1.47e+04_{4.6e+04}$	$2.62e+04_{6.7e+04}$
	$\mathcal{A}_3\mathcal{I}_4$	$1.17e+04_{2.7e+04}$	$1.19e+04_{4.5e+04}$	$7.79e+03_{2.3e+04}$	$1.62e+04_{5.8e+04}$	$2.73e+04_{7.0e+04}$
	$\mathcal{A}_4\mathcal{I}_1$	$1.37e+04_{3.2e+04}$	$1.20e+04_{4.3e+04}$	$8.36e+03_{2.7e+04}$	$1.69e+04_{5.6e+04}$	$2.98e+04_{8.4e+04}$
	$\mathcal{A}_4\mathcal{I}_2$	$1.44e+04_{3.6e+04}$	$1.38e+04_{5.6e+04}$	$9.36e+03_{3.1e+04}$	$1.81e+04_{6.3e+04}$	$2.92e+04_{7.4e+04}$
	$\mathcal{A}_4\mathcal{I}_3$	$1.41e+04_{3.2e+04}$	$1.32e+04_{4.9e+04}$	$1.03e+04_{3.6e+04}$	$1.84e+04_{6.6e+04}$	$3.02e+04_{7.6e+04}$
	$\mathcal{A}_4\mathcal{I}_4$	$1.46e+04_{3.4e+04}$	$1.22e+04_{3.2e+04}$	$1.12e+04_{4.2e+04}$	$1.72e+04_{4.9e+04}$	$3.22e+04_{8.6e+04}$
	$\mathcal{A}_5\mathcal{I}_1$	$2.39e+04_{6.4e+04}$	$1.85e+04_{5.3e+04}$	$1.71e+04_{6.6e+04}$	$2.67e+04_{8.8e+04}$	$4.06e+04_{1.0e+05}$
	$\mathcal{A}_5\mathcal{I}_2$	$2.63e+04_{7.7e+04}$	$2.02e+04_{6.3e+04}$	$1.68e+04_{6.1e+04}$	$2.93e+04_{1.1e+05}$	$4.20e+04_{1.1e+05}$
	$\mathcal{A}_5\mathcal{I}_3$	$2.90e+04_{9.5e+04}$	$2.26e+04_{7.9e+04}$	$1.63e+04_{5.2e+04}$	$2.68e+04_{7.9e+04}$	$4.75e+04_{1.3e+05}$
	$\mathcal{A}_5\mathcal{I}_4$	$2.92e+04_{1.1e+05}$	$2.46e+04_{9.2e+04}$	$1.84e+04_{6.7e+04}$	$2.66e+04_{7.4e+04}$	$4.73e+04_{1.3e+05}$
Median and Interquartile Range	$\mathcal{A}_1\mathcal{I}_1$	$3.87e+02_{1.5e+02}$	$3.70e+02_{9.7e+01}$	$4.02e+02_{1.2e+02}$	$3.31e+02_{1.2e+02}$	$7.31e+02_{6.4e+02}$
	$\mathcal{A}_1\mathcal{I}_2$	$4.23e+02_{1.7e+02}$	$4.09e+02_{1.2e+02}$	$4.33e+02_{1.2e+02}$	$3.57e+02_{1.4e+02}$	$7.70e+02_{7.2e+02}$
	$\mathcal{A}_1\mathcal{I}_3$	$4.62e+02_{1.7e+02}$	$4.02e+02_{1.1e+02}$	$4.25e+02_{1.1e+02}$	$3.57e+02_{1.3e+02}$	$7.56e+02_{6.7e+02}$
	$\mathcal{A}_1\mathcal{I}_4$	$4.98e+02_{2.2e+02}$	$4.31e+02_{1.2e+02}$	$4.48e+02_{1.2e+02}$	$3.73e+02_{1.5e+02}$	$7.88e+02_{6.7e+02}$
	$\mathcal{A}_2\mathcal{I}_1$	$1.21e+03_{3.8e+02}$	$9.03e+02_{1.6e+02}$	$8.76e+02_{1.8e+02}$	$9.22e+02_{2.8e+02}$	$1.35e+03_{8.0e+02}$
	$\mathcal{A}_2\mathcal{I}_2$	$1.30e+03_{3.4e+02}$	$9.86e+02_{1.7e+02}$	$9.49e+02_{1.8e+02}$	$9.87e+02_{2.6e+02}$	$1.42e+03_{8.1e+02}$
	$\mathcal{A}_2\mathcal{I}_3$	$1.34e+03_{3.5e+02}$	$1.05e+03_{1.8e+02}$	$1.02e+03_{1.8e+02}$	$1.05e+03_{2.6e+02}$	$1.50e+03_{8.4e+02}$
	$\mathcal{A}_2\mathcal{I}_4$	$1.45e+03_{3.4e+02}$	$1.13e+03_{1.9e+02}$	$1.08e+03_{2.0e+02}$	$1.12e+03_{3.1e+02}$	$1.56e+03_{9.0e+02}$
	$\mathcal{A}_3\mathcal{I}_1$	$2.67e+03_{9.7e+02}$	$1.87e+03_{3.3e+02}$	$1.70e+03_{2.8e+02}$	$2.06e+03_{6.8e+02}$	$2.51e+03_{1.1e+03}$
	$\mathcal{A}_3\mathcal{I}_2$	$2.83e+03_{9.3e+02}$	$1.98e+03_{3.3e+02}$	$1.81e+03_{3.3e+02}$	$2.17e+03_{7.3e+02}$	$2.62e+03_{1.0e+03}$
	$\mathcal{A}_3\mathcal{I}_3$	$2.93e+03_{9.2e+02}$	$2.10e+03_{3.5e+02}$	$1.94e+03_{3.4e+02}$	$2.26e+03_{7.5e+02}$	$2.73e+03_{1.2e+03}$
	$\mathcal{A}_3\mathcal{I}_4$	$3.04e+03_{9.1e+02}$	$2.19e+03_{3.2e+02}$	$2.03e+03_{3.4e+02}$	$2.36e+03_{7.3e+02}$	$2.85e+03_{1.1e+03}$
	$\mathcal{A}_4\mathcal{I}_1$	$3.34e+03_{1.2e+03}$	$2.31e+03_{4.0e+02}$	$2.09e+03_{3.6e+02}$	$2.61e+03_{9.1e+02}$	$3.06e+03_{1.1e+03}$
	$\mathcal{A}_4\mathcal{I}_2$	$3.50e+03_{1.2e+03}$	$2.45e+03_{4.2e+02}$	$2.24e+03_{4.5e+02}$	$2.72e+03_{9.6e+02}$	$3.19e+03_{1.2e+03}$
	$\mathcal{A}_4\mathcal{I}_3$	$3.62e+03_{1.2e+03}$	$2.58e+03_{4.6e+02}$	$2.34e+03_{4.0e+02}$	$2.84e+03_{8.9e+02}$	$3.33e+03_{1.2e+03}$
	$\mathcal{A}_4\mathcal{I}_4$	$3.84e+03_{1.2e+03}$	$2.71e+03_{4.4e+02}$	$2.46e+03_{4.0e+02}$	$2.95e+03_{9.6e+02}$	$3.45e+03_{1.2e+03}$
	$\mathcal{A}_5\mathcal{I}_1$	$6.45e+03_{2.6e+03}$	$4.31e+03_{9.3e+02}$	$3.70e+03_{7.2e+02}$	$4.98e+03_{1.7e+03}$	$5.53e+03_{2.2e+03}$
	$\mathcal{A}_5\mathcal{I}_2$	$6.67e+03_{2.6e+03}$	$4.49e+03_{9.2e+02}$	$3.88e+03_{6.7e+02}$	$5.17e+03_{1.7e+03}$	$5.73e+03_{2.2e+03}$
	$\mathcal{A}_5\mathcal{I}_3$	$6.82e+03_{2.6e+03}$	$4.66e+03_{8.7e+02}$	$4.07e+03_{7.6e+02}$	$5.30e+03_{1.8e+03}$	$5.90e+03_{2.1e+03}$
	$\mathcal{A}_5\mathcal{I}_4$	$7.10e+03_{2.6e+03}$	$4.85e+03_{8.7e+02}$	$4.25e+03_{7.8e+02}$	$5.45e+03_{1.6e+03}$	$6.10e+03_{2.0e+03}$

Overview of Generative AI Tools Used

I acknowledge the use of ChatGPT[1] with model 4 and 4o as I conducted literature research by uploading the papers and letting it summarize and explain, e.g., scientific methods and formulas, to help understanding scientific papers and their key messages. I critically reviewed the ChatGPT feedback and revised the writing using my own words and expressions.

I acknowledge the use of DeepL Translate[2] to generate translations of key terms between English and German.

I acknowledge the use of QuillBot[3] for grammar checking at the final stage of the thesis.

[1] https://chatgpt.com/
[2] https://www.deepl.com/
[3] https://quillbot.com/

List of Figures

List of Tables

List of Algorithms

Bibliography

[1] Abolfazli, S., Sanaei, Z., and Gani, A. Mobile Cloud Computing: A Review on Smartphone Augmentation Approaches.

[2] Aceto, L., Morichetta, A., and Tiezzi, F. Decision support for mobile cloud computing applications via model checking. *Proceedings - 2015 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, MobileCloud 2015*, Mcc (2015), 199–204.

[3] Aceto, L., Morichetta, A., and Tiezzi, F. Decision support for mobile cloud computing applications via model checking. In *2015 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering* (2015), IEEE, pp. 199–204.

[4] Ali, F. A., Simoens, P., Verbelen, T., Demeester, P., and Dhoedt, B. Mobile device power models for energy efficient dynamic offloading at runtime. *Journal of Systems and Software 113* (2016), 173–187.

[5] Aral, A., and De Maio, V. Simulators and emulators for edge computing. In *Edge Computing: Models, technologies and applications*. Institution of Engineering and Technology, aug 2020, pp. 291–311.

[6] Araujo, E., Araujo, F. P., Becceneri, J. C., and Campos Velho, H. F. Particle Swarm Optimization with Turbulence (PSOT) applied to thermal-vacuum modelling. *IEEE International Conference on Fuzzy Systems* (2009), 344–349.

[7] Atashpendar, A., and Bouvry, P. A Parallel Cooperative Coevolutionary SMPSO Algorithm for Multi-objective Optimization. 713–720.

[8] Barba-González, C., Nebro, A. J., García-Nieto, J., and Aldana-Montes, J. F. A multi-objective interactive dynamic particle swarm optimizer. *Progress in Artificial Intelligence* (jul 2019), 1–11.

[9] Beiranvand, V., Hare, W., and Lucet, Y. Best practices for comparing optimization algorithms. *Optimization and Engineering 18*, 4 (2017), 815–848.

[10] Bezerra, L. C., López-Ibáñez, M., and Stützle, T. An empirical assessment of the properties of inverted generational distance on multi- and many-objective optimization. *Lecture Notes in Computer Science (including subseries Lecture Notes in*

Artificial Intelligence and Lecture Notes in Bioinformatics) 10173 LNCS (2017), 31–45.

[11] Blank, J., and Deb, K. pymoo: Multi-Objective Optimization in Python. *IEEE Access 8* (2020), 89497–89509.

[12] Brogi, A., and Forti, S. QoS-aware Deployment of IoT Applications Through the Fog. *IEEE Internet of Things Journal 4*, 5 (2017), 1185–1192.

[13] Brogi, A., Forti, S., and Ibrahim, A. How to Best Deploy Your Fog Applications, Probably. *Proceedings - 2017 IEEE 1st International Conference on Fog and Edge Computing, ICFEC 2017* (2017), 105–114.

[14] Brogi, A., Forti, S., and Ibrahim, A. Deploying fog applications: How much does it cost, by the way? *CLOSER 2018 - Proceedings of the 8th International Conference on Cloud Computing and Services Science 2018-Janua*, Closer 2018 (2018), 68–77.

[15] Brogi, A., Forti, S., Ibrahim, A., and Others. Predictive analysis to support fog application deployment. *Fog and Edge Computing: Principles and Paradigms* (jan 2019), 191–229.

[16] Cao, Y., Smucker, B. J., and Robinson, T. J. On using the hypervolume indicator to compare Pareto fronts: Applications to multi-criteria optimal experimental design. *Journal of Statistical Planning and Inference 160* (may 2015), 60–74.

[17] Chen, W. N., and Tan, D. Z. Set-based discrete particle swarm optimization and its applications: a survey. *Frontiers of Computer Science 12*, 2 (2018), 203–216.

[18] Chen, X., Jiao, L., Li, W., and Fu, X. Efficient Multi-User Computation Offloading for Mobile-Edge Cloud Computing. *IEEE/ACM Transactions on Networking 24*, 5 (2016), 2795–2808.

[19] Chun, B.-G., Ihm, S., Maniatis, P., and Naik, M. Clonecloud: boosting mobile device applications through cloud clone execution. *arXiv preprint arXiv:1009.3088* (2010).

[20] Coello, C. A. C., Luna, F., and Alba, E. SMPSO : A New PSO Metaheuristic for Multi-objective Optimization.

[21] Coello Coello, C. A., and Reyes Sierra, M. A study of the parallelization of a coevolutionary multi-objective evolutionary algorithm. In *MICAI 2004: Advances in Artificial Intelligence: Third Mexican International Conference on Artificial Intelligence, Mexico City, Mexico, April 26-30, 2004. Proceedings 3* (2004), Springer, pp. 688–697.

[22] Coello Coello, C. A., and Reyes-Sierra, M. Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art. *International Journal of Computational Intelligence Research 2*, 3 (2006).

[23] Cuervo, E., Balasubramanian, A., Cho, D.-k., Wolman, A., Saroiu, S., Chandra, R., and Bahl, P. Maui: making smartphones last longer with code offload. In *Proceedings of the 8th international conference on Mobile systems, applications, and services* (2010), pp. 49–62.

[24] Cui, Y., Geng, Z., Zhu, Q., and Han, Y. Review: Multi-objective optimization methods and application in energy saving. *Energy 125* (2017), 681–704.

[25] Dai, C., Wang, Y., and Ye, M. A new multi-objective particle swarm optimization algorithm based on decomposition. *Information Sciences 325* (dec 2015), 541–557.

[26] De Maio, V., and Brandic, I. First Hop Mobile Offloading of DAG Computations. *2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)* (2018), 83–92.

[27] De Rainville, F. M., Fortin, F. A., Gardner, M. A., Parizeau, M., and Gagné, C. DEAP: A Python framework for Evolutionary Algorithms. *GECCO'12 - Proceedings of the 14th International Conference on Genetic and Evolutionary Computation Companion* (2012), 85–92.

[28] Deb, K., and Datta, R. A fast and accurate solution of constrained optimization problems using a hybrid bi-objective and penalty function approach. *2010 IEEE World Congress on Computational Intelligence, WCCI 2010 - 2010 IEEE Congress on Evolutionary Computation, CEC 2010* (2010).

[29] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation 6*, 2 (2002), 182–197.

[30] Deb, K., Thiele, L., Laumanns, M., and Zitzler, E. Scalable multi-objective optimization test problems. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)* (2002), vol. 1, IEEE, pp. 825–830.

[31] Dinh, H. T., Lee, C., Niyato, D., and Wang, P. A survey of mobile cloud computing: Architecture, applications, and approaches. *Wireless Communications and Mobile Computing 13*, 18 (dec 2013), 1587–1611.

[32] Durillo, J. J., Fard, H. M., and Prodan, R. MOHEFT: A multi-objective list-based method for workflow scheduling. *CloudCom 2012 - Proceedings: 2012 4th IEEE International Conference on Cloud Computing Technology and Science* (2012), 185–192.

[33] Durillo, J. J., and Nebro, A. J. JMetal: A Java framework for multi-objective optimization. *Advances in Engineering Software 42*, 10 (2011), 760–771.

[34] Eberhart, R., and Kennedy, J. Particle swarm optimization. In *Proceedings of the IEEE international conference on neural networks* (1995), vol. 4, Citeseer, pp. 1942–1948.

[35] Fahimullah, M., Philippe, G., Ahvar, S., and Trocan, M. Simulation Tools for Fog Computing: A Comparative Analysis. *Sensors 2023, Vol. 23, Page 3492 23*, 7 (mar 2023), 3492.

[36] Feng, C., Han, P., Zhang, X., Yang, B., Liu, Y., and Guo, L. Computation offloading in mobile edge computing networks: A survey. *Journal of Network and Computer Applications 202*, April (2022), 103366.

[37] Fernando, N., Loke, S. W., and Rahayu, W. Mobile cloud computing: A survey. *Future Generation Computer Systems 29*, 1 (2013), 84–106.

[38] Fortin, F.-A., De Rainville, F.-M., Gardner, M.-A., Parizeau, M., and Gagné, C. DEAP: Evolutionary Algorithms Made Easy. *Journal of Machine Learning Research 13* (jul 2012), 2171–2175.

[39] Friedman, M. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the american statistical association 32*, 200 (1937), 675–701.

[40] Geng, Z., Wang, Z., Zhu, Q., and Han, Y. Multi-objective operation optimization of ethylene cracking furnace based on AMOPSO algorithm. *Chemical Engineering Science 153* (oct 2016), 21–33.

[41] Gu, X. L., Huang, M., and Liang, X. A Discrete Particle Swarm Optimization Algorithm with Adaptive Inertia Weight for Solving Multiobjective Flexible Job-shop Scheduling Problem. *IEEE Access 8* (2020), 33125–33136.

[42] Gunantara, N. A review of multi-objective optimization: Methods and its applications. *Cogent Engineering 5*, 1 (jan 2018), 1–16.

[43] Gupta, H., Vahid Dastjerdi, A., Ghosh, S. K., and Buyya, R. iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments. *Software - Practice and Experience 47*, 9 (2017), 1275–1296.

[44] Hadka, D. MOEA Framework: A Free and Open Source Java Framework for Multiobjective Optimization (Version 4.5), 2024.

[45] Haibeh, L. A., Yagoub, M. C., and Jarray, A. A Survey on Mobile Edge Computing Infrastructure: Design, Resource Management, and Optimization Approaches. *IEEE Access 10* (2022), 27591–27610.

[46] Han, F., Chen, W. T., Ling, Q. H., and Han, H. Multi-objective particle swarm optimization with adaptive strategies for feature selection. *Swarm and Evolutionary Computation 62*, February 2020 (apr 2021), 100847.

[47] Han, H., Zhou, H., Huang, Y., and Hou, Y. Robust Multiobjective Particle Swarm Optimization with Feedback Compensation Strategy. *IEEE Transactions on Cybernetics 54*, 2 (feb 2024), 1062–1074.

[48] Helbig, M., Deb, K., and Engelbrecht, A. Key challenges and future directions of dynamic multi-objective optimisation. *2016 IEEE Congress on Evolutionary Computation, CEC 2016* (nov 2016), 1256–1261.

[49] Helbig, M., and Engelbrecht, A. P. Population-based metaheuristics for continuous boundary-constrained dynamic multi-objective optimisation problems. *Swarm and Evolutionary Computation 14* (feb 2014), 31–47.

[50] Hongbo, L., and Abraham, A. Fuzzy adaptive turbulent particle swarm optimization. *Proceedings - HIS 2005: Fifth International Conference on Hybrid Intelligent Systems 2005* (2005), 445–450.

[51] Hou, Y., Hao, G., Zhang, Y., Gu, F., and Xu, W. A multi-objective discrete particle swarm optimization method for particle routing in distributed particle filters. *Knowledge-Based Systems 240* (2022), 108068.

[52] Hrolenok, B. Multi-objective Optimization with PSO. 1–2.

[53] Ishibuchi, H., Masuda, H., and Nojima, Y. A study on performance evaluation ability of a modified inverted generational distance indicator. In *GECCO 2015 - Proceedings of the 2015 Genetic and Evolutionary Computation Conference* (jul 2015), Association for Computing Machinery, Inc, pp. 695–702.

[54] Jain, H., and Deb, K. An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part II: Handling constraints and extending to an adaptive approach. *IEEE Transactions on evolutionary computation 18*, 4 (2013), 602–622.

[55] Knowles, J. D., and Corne, D. W. Approximating the nondominated front using the Pareto Archived Evolution Strategy. *Evolutionary computation 8*, 2 (2000), 149–172.

[56] Kunde, C., and Mann, Z. Á. Comparison of simulators for fog computing. *Proceedings of the ACM Symposium on Applied Computing* (2020), 1792–1795.

[57] Lalwani, S., Singhal, S., Kumar, R., and Gupta, N. a Comprehensive Survey: Applications of Multi-Objective Particle Swarm Optimization (Mopso) Algorithm. *Transactions on Combinatorics ISSN 2*, 1 (2013), 2251–8657.

[58] Laumanns, M., Thiele, L., Deb, K., and Zitzler, E. Combining convergence and diversity in evolutionary multiobjective optimization. *Evolutionary Computation 10*, 3 (2002), 263–282.

[59] Lera, I., Guerrero, C., and Juiz, C. YAFS: A Simulator for IoT Scenarios in Fog Computing. *IEEE Access 7* (2019), 91745–91758.

[60] Lin, X., Luo, W., Gu, N., and Zhang, Q. A novel dynamic reference point model for preference-based evolutionary multiobjective optimization. *Complex and Intelligent Systems 9*, 2 (apr 2023), 1415–1437.

[61] López-Ibanez, M., Devi Prasad, T., and Paechter, B. Multi-objective optimisation of the pump scheduling problem using SPEA2. *2005 IEEE Congress on Evolutionary Computation, IEEE CEC 2005. Proceedings 1* (2005), 435–442.

[62] Luan, T. H., Gao, L., Li, Z., Xiang, Y., Wei, G., and Sun, L. Fog Computing: Focusing on Mobile Users at the Edge. *Journal of Network and Computer Applications 52* (2015), 11–25.

[63] Mach, P., and Becvar, Z. Mobile Edge Computing: A Survey on Architecture and Computation Offloading, 2017.

[64] Mavrotas, G. Effective implementation of the ϵ-constraint method in Multi-Objective Mathematical Programming problems. *Applied Mathematics and Computation 213*, 2 (jul 2009), 455–465.

[65] McCrum-Gardner, E. Which is the correct statistical test to use? *British Journal of Oral and Maxillofacial Surgery 46*, 1 (jan 2008), 38–41.

[66] Naouri, A., Wu, H., Nouri, N. A., Dhelim, S., and Ning, H. A Novel Framework for Mobile-Edge Computing by Optimizing Task Offloading. *IEEE Internet of Things Journal 8*, 16 (2021), 13065–13076.

[67] Nebro, A. J., Barba-González, C., Nieto, J. G., Cordero, J. A., and Montes, J. F. A. Design and architecture of the jMetaISP framework. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion on - GECCO '17* (New York, New York, USA, 2017), ACM Press, pp. 1239–1246.

[68] Nebro, A. J., Durillo, J. J., García-Nieto, J., Barba-González, C., Del Ser, J., Coello Coello, C. A., Benítez-Hidalgo, A., and Aldana-Montes, J. F. Extending the speed-constrained multi-objective PSO (SMPSO) with reference point based preference articulation. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (2018), vol. 11101 LNCS, Springer Verlag, pp. 298–310.

[69] Nebro, A. J., Durillo, J. J., and Vergne, M. Redesigning the jMetal multi-objective optimization framework. *GECCO 2015 - Companion Publication of the 2015 Genetic and Evolutionary Computation Conference* (jul 2015), 1093–1100.

[70] Nebro, A. J., Ruiz, A. B., Barba-González, C., García-Nieto, J., Luque, M., and Aldana-Montes, J. F. InDM2: Interactive Dynamic Multi-Objective Decision Making Using Evolutionary Algorithms. *Swarm and Evolutionary Computation 40* (jun 2018), 184–195.

[71] Nshimirimana, R., Abraham, A., and Nothnagel, G. *A multi-objective particle swarm for constraint and unconstrained problems*, vol. 33. Springer London, 2021.

[72] Pandis, N. The chi-square test. *American journal of orthodontics and dentofacial orthopedics 150*, 5 (2016), 898–899.

[73] Peng, W., and Zhang, Q. A decomposition-based multi-objective particle swarm optimization algorithm for continuous optimization problems. *2008 IEEE International Conference on Granular Computing, GRC 2008* (2008), 534–537.

[74] Puliafito, C., Gonçalves, D. M., Lopes, M. M., Martins, L. L., Madeira, E., Mingozzi, E., Rana, O., and Bittencourt, L. F. MobFogSim: Simulation of mobility and migration for fog computing. *Simulation Modelling Practice and Theory 101* (2020), 102062.

[75] Qiao, N., You, J., Sheng, Y., Wang, J., and Deng, H. An efficient algorithm of discrete particle swarm optimization for multi-objective task assignment. *IEICE Transactions on Information and Systems E99D*, 12 (2016), 2968–2977.

[76] Raquel, C. R., and Naval, P. C. An effective use of crowding distance in multiobjective particle swarm optimization. *GECCO 2005 - Genetic and Evolutionary Computation Conference* (2005), 257–264.

[77] Rostami, S., Neri, F., and Gyaurski, K. On Algorithmic Descriptions and Software Implementations for Multi-objective Optimisation: A Comparative Study. *SN Computer Science 1*, 5 (sep 2020), 1–23.

[78] Rudenko, A., Reiher, P., Popek, G. J., and Kuenning, G. H. Saving portable computer battery power through remote process execution. *ACM SIGMOBILE Mobile Computing and Communications Review 2*, 1 (jan 1998), 19–26.

[79] Schott, J. R. *Fault tolerant design using single and multicriteria genetic algorithm optimization*. PhD thesis, Massachusetts Institute of Technology, 1995.

[80] Shi, W., and Dustdar, S. The Promise of Edge Computing. *Computer 49*, 5 (2016), 78–81.

[81] Simons, H. You told us: Better performance or battery life? This wasn't close at all, 2023.

[82] Sonmez, C., Ozgovde, A., and Ersoy, C. EdgeCloudSim: An environment for performance evaluation of edge computing systems. *Transactions on Emerging Telecommunications Technologies 29*, 11 (nov 2018), e3493.

[83] Taheri, S. M., and Hesamian, G. A generalization of the Wilcoxon signed-rank test and its applications. *Statistical Papers 54*, 2 (2013), 457–470.

[84] Tian, Y., Cheng, R., Zhang, X., and Jin, Y. PlatEMO: A MATLAB Platform for Evolutionary Multi-Objective Optimization [Educational Forum]. *IEEE Computational Intelligence Magazine 12*, 4 (2017), 73–87.

[85] Uma Maheswari, K. M., and Govindarajan, S. A survey of various cloud scheduling algorithms. *Journal of Advanced Research in Dynamical and Control Systems 9*, 7 (2017), 1–8.

[86] Veček, N., Črepinšek, M., and Mernik, M. On the influence of the number of algorithms, problems, and independent runs in the comparison of evolutionary algorithms. *Applied Soft Computing Journal 54* (2017), 23–45.

[87] Veldhuizen, D. A. V., and Lamont, G. B. Multiobjective Evolutionary Algorithm Research : A History and Analysis. *Technical Report TR-98-03, Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio 1998* (1998), 1–88.

[88] Venkatesan, S. P., and Kumanan, S. *A Multi-Objective Discrete Particle Swarm Optimisation Algorithm for supply chain network design*, vol. 11. 2012.

[89] Verma, A., and Kaushal, S. A hybrid multi-objective Particle Swarm Optimization for scientific workflow scheduling. *Parallel Computing 62* (feb 2017), 1–19.

[90] Wang, H., Olhofer, M., and Jin, Y. A mini-review on preference modeling and articulation in multi-objective optimization: current status and challenges. *Complex and Intelligent Systems 3*, 4 (aug 2017), 233–245.

[91] Wilson, K., and Rostami, S. *On the integrity of performance comparison for evolutionary multi-objective optimisation algorithms*, vol. 840. Springer Verlag, 2019.

[92] Wu, R., Li, Y., Guo, S., and Li, X. An Efficient Meta-Heuristic for Multi-Objective Flexible Job Shop Inverse Scheduling Problem. *IEEE Access 6* (2018), 59515–59527.

[93] Zhang, Q., Liu, Y., Han, H., Yang, M., and Shu, X. Multi-Objective Particle Swarm Optimization with Multi-Archiving Strategy. *Scientific Programming 2022*, 1 (jan 2022), 7372450.

[94] Zhang, W., and Hansen, K. M. An evaluation of the NSGA-II and MOCell genetic algorithms for self-management planning in a pervasive service middleware. *Proceedings of the IEEE International Conference on Engineering of Complex Computer Systems, ICECCS* (2009), 192–201.

[95] Zheng, W., and Sakellariou, R. A Monte-Carlo approach for full-ahead stochastic DAG scheduling. *Proceedings of the 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops, IPDPSW 2012*, November 2015 (2012), 99–112.

[96] Zhou, A., Jin, Y., Zhang, Q., Sendhoff, B., and Tsang, E. Combining model-based and genetics-based offspring generation for multi-objective optimization using a convergence criterion. In *2006 IEEE international conference on evolutionary computation* (2006), IEEE, pp. 892–899.

[97] Zitzler, E., Knowles, J., and Thiele, L. Quality Assessment of Pareto Set Approximations. 2008, pp. 373–404.

[98] Zitzler, E., and Thiele, L. Multiobjective Evolutionary Algorithms : A Comparative Case Study. *Nature 3*, September (1998), 257–271.

[99] Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M., and Da Fonseca, V. G. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation 7*, 2 (2003), 117–132.