Dynamic Update of CSP Allocations in Federation Orchestration

MARIEKE BEKE
MASTER´S THESIS
DEPARTMENT OF ELECTRICAL AND INFORMATION TECHNOLOGY
FACULTY OF ENGINEERING | LTH | LUND UNIVERSITY

Dynamic Update of CSP Allocations in Federation Orchestration

Marieke Beke
ma8112be-s@student.lu.se

Department of Electrical and Information Technology
Lund University

Supervisors: Emma Fitzgerald, William Tärneberg

Examiner: Christian Nyberg

June 14, 2024

Printed in Sweden
Tryckeriet i E-huset, Lund

Abstract

This paper discusses the project in which an algorithm was written to dynamically approach federation orchestration. This builds on and is part of the REINDEER project. In an environment with goods and online agents, and in this specific case, Contact Service Points (CSPs) and User Equipments (UEs), where the agents have different applications and make unpredictable changes such as moving around a room, finding a dynamic reallocation algorithm is a significant challenge.

The use of diverse case studies, encompassing both minor and significant adjustments to an allocation, has proven to be an effective approach in identifying more optimal solutions. An iterative methodology has been employed, wherein a randomly selected case is evaluated in each iteration. Only if a superior utility value is achieved is the new solution accepted. The utility values are calculated by a entity, designated as the oracle, which was not a part of the original project scope.

This approach has yielded higher, yet more consistent utility values compared to random reallocation. This is the case for different numbers of iterations, different numbers of CSPs and different numbers of UEs.

The runtime can become a bottleneck of the algorithm if more iterations have to be chosen. This is due to the fact that the oracle must be called every iteration. Overall, the addition of the dynamic reallocation algorithm is a relevant addition.

Popular Science Summary

Development of a Dynamic Reallocation Algorithm for federation orchestration in the RadioWeaves infrastructure

In today's rapidly evolving technological landscape, efficient resource management is crucial for the optimal performance of distributed systems. This project, part of the innovative REINDEER research project, focuses on dynamic resource allocation within federated networks, where multiple independent entities collaborate to share resources like processing power, storage, and connectivity. These networks must swiftly adapt to the unpredictable movements and varying demands of users. The intelligent algorithm continuously adjusts resource distribution, based on data input, saving computing time and improving efficiency. This research has implications for the future of networked systems, enabling technologies that support next-generation communication infrastructure.

In the realm of federated environments, efficient resource management is pivotal for optimal performance. The REINDEER project, funded by the European Union's Horizon 2020 programme, aims to revolutionize connectivity through RadioWeaves technology, focusing on resilience, interactivity, hyper-diversity, and energy efficiency.

A crucial part of the RadioWeaves infrastructure is federation orchestration, this implies managing Contact Service Points (CSPs) and User Equipments (UEs). CSPs are the resources, indivisible goods that serve the UEs, online agents. UEs with similar applications are grouped into federations to streamline resource allocation. The key challenge addressed in this project is the development of a dynamic reallocation algorithm. Unlike traditional static methods, this algorithm adapts to changes such as UEs coming online or moving within a room, ensuring optimal CSP allocation. It leverages an iterative approach where adjustments are made incrementally based on utility evaluations by an evaluation component, the oracle.

Testing and analysis demonstrate the algorithm's effectiveness across varying scenarios, including different numbers of CSPs and UEs. Results consistently show improved utility values compared

to static allocation methods, confirming its suitability for dynamic environments.

Looking ahead, future enhancements could relax constraints on application homogeneity among UEs within federations, explore weighted case prioritisation, and optimise computational efficiency to further refine allocation results. These efforts will pave the way for more efficient and adaptable network infrastructures in the future.

Table of Contents

List of Figures

Chapter 1
Introduction

Modern distributed systems landscape requires effective resource management and optimization for it to be efficient and scalable. This paper explores federated environments in the REINDEER project focusing on orchestration of resources to support dynamic workload demands and operational challenges. Fundamentally, REINDEER aims at defining a network framework supported by RadioWeaves technology using advanced algorithms, and dynamic methodologies to enhance performance[10].

Within REINDEER, federation orchestration is emerging as a critical component that serves as a focal point of resource allocation of Contact Service Points (CSPs) among different User Equipments (UEs)[9]. This paper navigates through intricate levels of federation orchestration by clarifying the abstract problem formulation and investigating techniques aimed at optimizing resource utilization.

The primary concern addressed in this thesis is how goods can be reallocated dynamically to agents operating in a federated environment. Through an intermediate exposition that discusses issues arising from indivisible goods distribution, online agent behavior, dynamics of federations, this manuscript lays the foundation for a dynamically driven approach towards allocating resources different from traditional static approaches.

The core of this thesis project is the development and utilisation of a sophisticated algorithm that is capable of addressing the specific characteristics of federated environments. The algorithm is iterative, operating through nine distinct cases that implement adjustments to the allocation. The Oracle, an existing tool within REINDEER, is employed to assess and contrast the allocations.

This thesis reviews, through extensive testing and analysis, the efficacy and performance of the proposed algorithm. With graphs and numbers positive results of the algorithm are evaluated and weaknesses are exposed.

Finally, this research brings the project's achievements to a conclusion by discussing the significant advances made in the field of resource management and optimisation in federation orchestration. The findings of this thesis work demonstrate the efficacy of a working algorithm for the continued development of the

REINDEER project. This thesis also suggests potential areas for further investigation and development with a view to improving the efficiency and adaptability of resource allocation strategies within distributed computing settings.

Chapter 2

Degree project as part of REINDEER project

2.1 REINDEER

To achieve a clear problem formulation, it is necessary to consider the bigger picture. Although the problem itself is abstract and can be separated from the REINDEER project, understanding this project is useful as it demonstrates how a larger research project arrives at the point of requiring an allocation algorithm. If such an algorithm is not readily available in existing research resources, it must be developed. That is why the abstraction is only made in the thinking process. However, during the implementation and testing phases, the naming and concepts of REINDEER will be used.

2.1.1 The next Generation of Connectivity

The REINDEER project is a research initiative that aims to develop an advanced connectivity platform. The project's name is an acronym for REsilient INteractive applications through hyper Diversity in Energy Efficient RadioWeaves technology, encapsulating several key concepts. In this context, resilience refers to the framework's ability to adapt and maintain effective functionality in the face of challenges or disturbances[10].

The project includes interactive applications, such as software or systems that dynamically engage with real-time user input and feedback. The term 'hyper-diversity' emphasises an exceptionally wide variety or range, highlighting the project's commitment to versatility. The project prioritises energy efficiency and optimising the use of energy resources within the infrastructure. This emphasis on sustainability ensures that the technology can operate efficiently over long periods without depleting energy resources. The RadioWeaves technology is at the heart of the project, enabling smart wireless and battery-free devices.

The European Union's Horizon 2020 research and innovation programme provides funding for this. The project will run for a period of 42 months and is divided into seven work packages. The workload of the project is divided into seven work packages. Lund University is responsible for Work Package 2 (WP2), which involves the RadioWeaves platform, including models, architectures, and topologies.

2.2 Federation Orchestration in REINDEER

The emerging field of 6G development aims to achieve unparalleled data rates, imperceptibly low latency, unparalleled dependability, and ultra-low power consumption. Additionally, there is a crucial emphasis on reducing the carbon footprint of network operations. To address these challenges, wireless access architectures are evolving, incorporating distributed radios. Computing resources are leveraged to their full potential through the emerging paradigm of cell-free networking. However, practical deployment of such architectures poses several challenges, including efficient resource coordination, infrastructure and service scalability, and the need for a robust foundation. REINDEER has introduced a part of this infrastructure. Dynamic federations are a crucial solution to address the challenges mentioned.

The landscape of 6G is rapidly evolving and requires support for a wide range of services, including an expected increase in connected devices and diverse applications. This requires infrastructures that can provide unparalleled capacity, ultra-reliable low-latency communication, and connectivity for a massive number of low-power devices. Furthermore, new features such as position-based applications and connections to energy-neutral devices highlight the need for advanced radio access architectures.

The proposed radio access concepts comprise Reflective Intelligent Surfaces (RIS), Cell-Free Massive MIMO (multiple input, multiple output), and Large Intelligent Surfaces (LIS). These innovations address wireless communication challenges. However, they do not cover the full spectrum of 6G services, including precise positioning and wireless energy transfer. The RadioWeaves system is introduced as a comprehensive solution that provides a distributed infrastructure capable of supporting a variety of services.

Recognising the limitations of current network architectures, dynamic federations are proposed as a solution for practical deployment. These federations consist of constellations of antennas, edge computing units, data storage, and other resources tailored to serve specific applications or application classes. The dynamic nature of federations, both in temporal and spatial domains, addresses the diverse requirements of applications. The term 'federation' refers to a cooperative group of resources that serve a common purpose, usually coordinated by an Edge Computing Service Point (ECSP).

To aid in the design and implementation of these systems, a specific set of terminology is introduced to differentiate between logical entities and physical elements. Logical entities consist of Contact Service Points (CSPs), ECSPs, and Federations, each with a distinct role in the dynamic federation concept. Physical elements include Sensing Elements, Data Storage Elements, Processing Elements, Charging Elements, and Radio Elements, which form the hardware foundation of the proposed architecture[9].

In this project we focus on a smaller part of this RadioWeaves implementation,

more specifically on routing resource allocation for the scope of a room or a hall. We can consider the network as consisting of UEs and CSPs. The UEs are the User Equipments. These have their own application and behavior. Communication is possible through the CSPs, Contact Service Points. To put it simple, UEs need CSPs to serve them. Given a set of requirements of the UEs there should be an allocation that meets these requirements. However, it is important to carefully consider the selection process to ensure the system is efficient, fast, and consumes minimal power. The REINDEER project team developed the federation orchestration setup with this in mind. This approach involves grouping UEs with the same application to be served together, resulting in a smarter allocation. A group of User Equipments (UEs) is referred to as a federation. Within a federation, UEs share the allocated CSPs. The challenge in making an allocation is to group UEs with similar applications, locations or behaviours and provide them with a set of CSPs that can fulfil the requirements of the federation. This setup is satisfactory for the project, but as soon as circumstances change, the usefulness of the federation orchestration may become questionable. Possible changes include UEs coming online or going offline, UEs changing location, or objects entering the room. Figure 2.1[9] provides an example of a deployment as described.

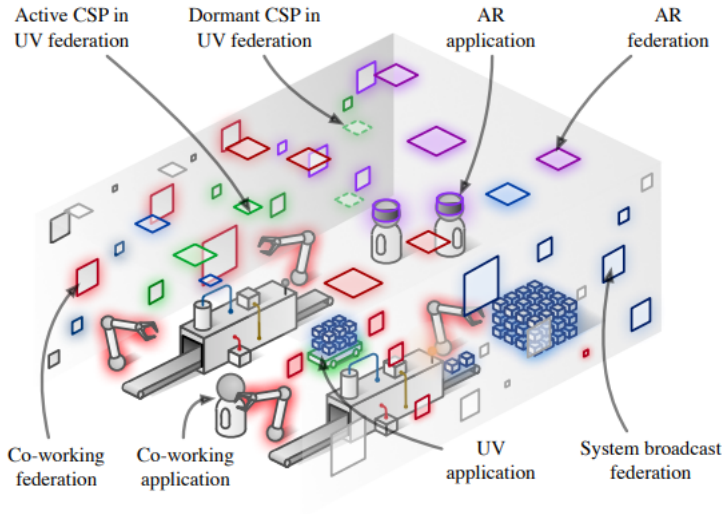


Figure 2.1: An example RadioWeaves deployment in a smart factory, with federations and their served devices colour coded. The four applications are augmented reality (AR) for professional applications (purple), tracking of robots and unmanned vehicles (UVs) (green), tracking of goods and real-time inventory (blue), and human-robot co-working (red)[9]

2.2.1 Dynamic reallocation

The current existing algorithm is insufficient for adapting to changes. Constructing a full new allocation for every change results in unnecessary calculations. A better approach is to keep the allocation and make minor modifications to adapt to new situations, such as switching CSPs or adding/removing UEs from the allocation. A new algorithm is required to enable the handling of UEs coming online and going offline, as this has not yet been developed in the REINDEER project. The algorithm should be able to adapt to changing positions of UEs and their varying needs for the number of serving CSPs. Additionally, it should conduct regular checks of the utility value of the full federation orchestration.

2.3 Abstract problem formulation

In the specific problem context of this paper, the term CSPs refers to a particular class of problems. In the broader literature, however, there are different approaches to solving similar problems. In abstract problem formulations, the term CSPs can be replaced by the concept of goods, which represent the entities or resources subject to allocation constraints.

Similarly, in a more generalised formulation, UEs can be likened to agents. Agents embody different roles and behaviours, similar to how UEs represent individual preferences and utilities in the context of CSPs.

The use of a more abstract problem formulation offers several advantages, including increased flexibility and generalisability. By adopting generic terms such as 'goods' and 'agents', researchers can apply methods and techniques to a wider range of problems beyond CSPs. This facilitates interdisciplinary collaboration and the transfer of knowledge between different domains.

Within the problem framework, the agents are organised into federations. These federations serve as groupings that aggregate agents with common goals or characteristics. This organisational structure allows for more efficient coordination, thereby increasing the overall effectiveness of the allocation process.

Furthermore, the goods involved are indivisible, meaning that each CSP is associated with a specific federation and cannot be divided among multiple federations. This constraint highlights the importance of strategic allocation decisions within each federation, as the indivisibility of the goods requires careful consideration of resource utilisation and distribution within the limits of each federation's capabilities.

In addition to abstracting CSPs into goods and UEs into agents, the use of an online algorithm further complicates the problem. Online algorithms are particularly valuable in dynamic environments where decisions must be made in real time or with incomplete information. In the context of resource allocation with UEs as agents, the use of online algorithms is unavoidable in order to respond adaptively

to changing conditions and unforeseen events. It is possible for agents to join and leave the system at any time. These algorithms are designed to continuously update their allocation strategies based on incoming data or feedback, thereby enabling agile decision-making even as the situation evolves. As agents interact within federations to allocate indivisible goods, online algorithms can dynamically adjust allocation decisions based on changing demands, emerging preferences, or new constraints. The utilisation of online algorithms, which are characterised by their flexibility and real-time decision-making capabilities, enables the enhancement of the overall efficiency and effectiveness of the allocation process in a dynamic and unpredictable environment.

Incorporating the concept of dynamic reallocation adds another layer of sophistication to an allocation strategy. Dynamic reallocation algorithms are designed to adapt and optimise allocation decisions in the face of changing circumstances. This type of algorithms are particularly valuable in environments where initial conditions can change unpredictably. Whether due to fluctuating demand, unforeseen events or evolving preferences, the ability to dynamically reallocate resources ensures that the algorithm remains robust and effective over time. Despite the uncertainties and complexities inherent in dynamic environments, these algorithms continue to operate effectively by continuously reassessing and adjusting allocation decisions based on the latest available information. The definition of reallocation conditions is essential to guide the behaviour of dynamic reallocation algorithms. These conditions act as triggers or thresholds that indicate when a reallocation event should occur. By establishing clear criteria for reallocation, can be ensured that resources are reallocated in a timely and appropriate manner, thereby optimising the overall performance of the allocation system.

There are many situations that may require a change in allocation. Here are a few to give an insight. Changes in the composition of federations, such as agents joining or leaving a federation, may require a reallocation to ensure that each federation remains adequately equipped to fulfil its objectives. Online agents within federations may experience shifts in their priorities or resource requirements over time. For example, an agent moving from one side of the room to the other may require reallocation to meet these evolving needs and ensure equitable distribution across federation members. Federations may seek to optimise their performance by reallocating indivisible goods based on real-time feedback and performance metrics. If one federation is under-utilising certain resources while another is experiencing resource shortages, reallocation can help balance resource distribution and improve overall federation efficiency.

In light of the above considerations, it is possible to explore the relevant literature in order to identify a solution that takes all of these factors into account. The required components are indivisible goods and online agents, which necessitate the use of an online algorithm. The agents form federations, and the goods are allocated to these federations. A dynamic approach is required. The above context needs to be incorporated into a dynamic reallocation algorithm.

Chapter 3

Dynamic reallocation for devision of goods to agents

3.1 Indivisible Goods

The article `Fair Allocation of Indivisible Goods`[6] discusses what it means to have indivisible goods. This refers to the division of a set of objects, goods, or items, where each object must be allocated as is, without being broken or divided into pieces. This concept is relevant in real-world situations such as divorce settlements involving physical objects like houses or cars. The fair allocation is complex, especially in determining whether a fair solution even exists for a given instance.

Several works talk about fairness. The authors in [8] explain fair division through the experimental exploration of two mechanisms, namely DYNAMIC DRF and CAUTIOUS LP. They discuss fairness in the context of the allocation of goods or resources among multiple agents in a manner that is perceived as equitable. The article analyses data to evaluate the effectiveness of mechanisms in achieving fairness objectives, specifically the sum of dominant shares (maxsum objective) and the minimum dominant share (maxmin objective) of the agents involved.

[1] serves as an introduction to the literature on discrete fair division, with a specific emphasis on additive valuation functions. Discrete fair division involves allocating indivisible goods or resources among agents, where each agent has a discrete valuation for the items being allocated. The article presents an overview of the advancements made in the field over the past decade, highlighting key findings and breakthroughs including the introduction of appropriate relaxations of envy-freeness and proportionality, such as envy-freeness up to one good (EF1) and envy-freeness up to any good.

The discussion on the dynamic adjustment of CSP allocations in Federation Orchestration is intertwined with the broader discourse on the fair division of resources. Fair division requires the development of mechanisms or algorithms capable of allocating indivisible resources in a manner that aligns with specified fairness criteria. The challenges faced in Federation Orchestration are similar to

those in resource allocation within federated networks.

However, in Federation Orchestration, the challenge extends beyond resource allocation to ensuring fairness among participating entities, such as federated service points and users. Federation Orchestration, like the fair division problem, must address factors such as asymmetric entitlements and dynamic settings.

Dynamic Adjustment of CSP Allocations in Federation Orchestration involves the adaptation of resource allocations to accommodate changing network dynamics and user demands. This task involves developing computational approaches that can dynamically adjust CSP allocations while adhering to fairness principles, such as maximin share fairness (MMS) and envy-freeness up to any good (EFX).

In summary, the discussion on equitable resource allocation provides insights that can be used in the dynamic adjustment of CSP allocations within federated networks but also demonstrates that achieving full fairness is hard to reach and prove.

3.2 Online agents

Three articles intersect in their discussion of the dynamic adjustment of CSP allocations in Federation Orchestration and the challenges and algorithmic approaches in the context of online resource allocation and fair division. They shed light on pertinent issues faced in dynamic online environments.

Banerjee, Gkatzelis, Gorokh and Jin [4] explored the complexities of online resource allocation and fair division. It discusses the challenges of distinguishing between agents that will be easy to satisfy later on and those that will be hard to satisfy, highlighting the limitations of online algorithms in achieving fairness in resource allocation. The text discusses the challenges faced in Federation Orchestration, where CSP allocations must be dynamically adjusted to meet varying resource demands and shifting user requirements while ensuring fairness among participating entities.

Additionally, online agents can be linked to the allocation of goods in online markets and considerations for revenue maximisation[3]. The authors elaborate about the dynamic nature of online resource allocation, specifically in adversarial arrival orders. This mirrors the need for dynamic adjustment of CSP allocations in Federation Orchestration to adapt to changing network conditions and demands.

Additionally, the paper `Online Algorithms for the Santa Claus Problem`[7] studies online assignment with a focus on the max-min objective. The text emphasizes the necessity of relaxing the problem in various ways due to significant gaps in approximation ratios. It highlights the intricate balance between fairness and efficiency in resource allocation. This resonates with the challenges faced in Federation Orchestration, where dynamic adjustments of CSP allocations must strike a balance between maximizing resource utilization and ensuring fair distri-

butions among federated entities.

The articles collectively contribute to the understanding of challenges and algorithmic approaches in online resource allocation. The research provides valuable insights applicable to the dynamic adjustment of CSP allocations in Federation Orchestration, ultimately enhancing efficiency of resource allocations in federated network environments.

3.3 Federations

The concept of federations in distributed computing infrastructures presents an intriguing approach to resource allocation. Dynamic adjustment of CSP allocations in federation orchestration is crucial for ensuring efficient resource utilization and fair distribution among users.

One of the key challenges in federation orchestration is managing resource allocation dynamically to adapt to changing demand and user preferences. This necessitates the development of robust allocation algorithms capable of dynamically adjusting resource allocations based on data and user feedback. Such algorithms should be able to optimize resource allocation while maintaining fairness and ensuring that all users receive their required resources in a timely manner.

Combinatorial auction bids serve as a mechanism for users to express their preferences for specific resources and their willingness to pay for them. The centralized auctioneer plays a critical role in facilitating these auctions and determining the allocation of resources based on the submitted bids. However, the dynamic adjustment of CSP allocations requires more than just auction mechanisms; it involves continuously monitoring resource usage, predicting demand trends, and adapting allocation strategies accordingly.[2]

Furthermore, the agreed-upon currency distribution policy mentioned in the abstract is essential for ensuring fairness and preventing resource starvation among users. By establishing clear guidelines for currency distribution, federations can mitigate the risk of heavy users monopolizing resources at the expense of lighter users. This policy can also incentivize users to accurately represent their resource needs and valuation, thus contributing to the overall efficiency of the allocation system.

The use of the Trusted-based Resource Allocation (TRA) algorithm in cloud federation environments is related to the ongoing discussion about the dynamic adjustment of CSP allocations in Federation Orchestration. The TRA algorithm is designed to address resource allocation challenges and improve the reliability of identity providers in cloud federations. However, its suitability for Federation Orchestration needs to be carefully considered.[12]

In Federation Orchestration, users operate independently of each other and rely on

the system for fair and reliable resource allocations. The TRA algorithm emphasizes leveraging mutual trust relationships between identity providers and clouds, which may not always align seamlessly. In contrast to traditional resource allocation systems, Federation Orchestration requires a system that users can trust implicitly, without the need for individual users to establish trust relationships.

Additionally, comparative analyses that demonstrate the effectiveness and superiority of the TRA algorithm over alternative approaches may not fully account for the unique requirements and dynamics of Federation Orchestration. Although the TRA algorithm may be efficient in allocating resources, enhancing security, and being cost-effective in traditional cloud federation environments, its suitability for the decentralized and disparate nature of Federation Orchestration is uncertain.

Considering these factors, it may be more appropriate for Federation Orchestration to prioritize decentralised decision-making and equitable resource distribution, without relying on mutual trust relationships. This approach would allow users to trust the system itself, rather than having to establish trust relationships with individual identity providers or clouds.

The TRA algorithm shows potential for addressing resource management challenges in cloud federations. However, its compatibility with the objectives and requirements of Federation Orchestration, particularly regarding user trust and decentralized decision-making, may require further evaluation and exploration of alternative approaches.

In conclusion, dynamic adjustment of CSP allocations in federation orchestration is a multifaceted problem that requires careful consideration of various factors, including user preferences, resource availability, and fairness concerns. By developing advanced allocation algorithms and implementing effective currency distribution policies, federations can achieve optimal resource utilization and ensure a balanced distribution of resources among users.

3.4 Dynamic approach

Tang[11] in their article argues for dynamic resource allocation mechanisms like what is required by multiple UAV tasks and emergent adjustments. Even though the article has emphasized UAV task management, the principles and methodologies behind it are useful for developing dynamic adjustment mechanisms within federated resource orchestration.

This emphasizes the need for adaptive resource allocation strategies that can adapt to changes in the workload's scenario. Also, when orchestrating federations, CSP allocations should be dynamically adjustable in response to variable demand, changing resources and end-user needs thereby optimizing resource utilization and ensuring efficient operation.

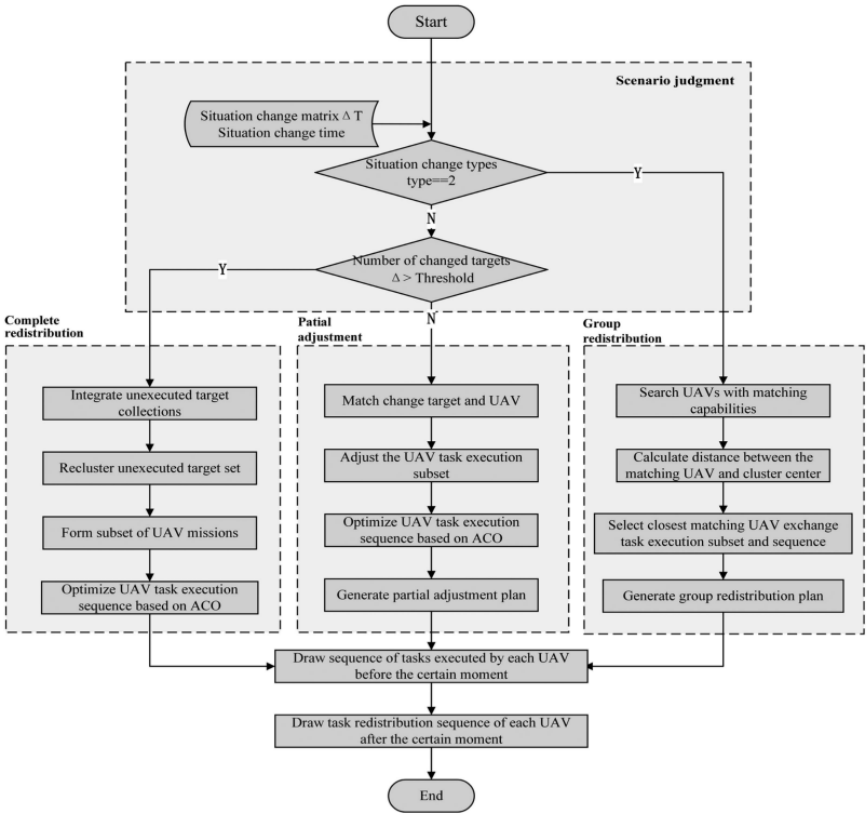


Figure 3.1: Architecture of the multi-UAV cooperative task reallocation algorithm in different scenarios[11]

These redistribution approaches involve complete reallocation; partial change; and pooling, which present a subtle way of dealing with changes occurring in task environments and state of UAVs. These techniques mirror the necessity for pliability and flexibility in a federated resource routing where there may be need to alter resource allocation dynamically in response to shifting demands of workloads and the constraints imposed by resources.In figure 3.1 the architecture scheme of UAV approach is shown. The idea of evaluating the situation and based on those results going for different kinds of redistributions or adjustments is relevant for federation orchestration.

Moreover, this research introduces the ant colony algorithm and FCM clustering based multi-UAV cooperative task reallocation algorithm which is a manifestation that it is possible to employ sophisticated optimization techniques for dynamic resource allocation. Similarly, in federation orchestration, incorporation of advanced allocation algorithms can improve the effectiveness and efficiency of resource dissemination procedures leading to enhanced system performance as well as user

satisfaction.

The diagrams produced from simulation showing how UAV tasks are carried out under different preallocation schemes also provide important information on how effective this model works after reallocation with various assumptions. Similarly, in federation orchestration, simulative feedback evaluations can offer insights into its dynamic adaptation mechanisms and further modify allocation algorithms looking at ever changing requirements of end-users' community as well as applications growth.
The dynamic reallocation model for multiple UAV tasks offers valuable insights and methodologies for managing resource allocation in specific contexts. However, it may not directly translate into an optimal solution for the dynamic adjustment of CSP allocations in federation orchestration.

One reason is the domain specificity of the model. It is tailored specifically for managing multiple UAV tasks in emergent adjustment scenarios, addressing challenges unique to UAV task management. These characteristics and constraints may differ significantly from those of federated resource orchestration, where a diverse range of resources and requirements must be considered.

Additionally, federated environments involve resource heterogeneity, encompassing various types such as computational resources, storage, and networking. The dynamic reallocation model may not adequately account for this heterogeneity and the diverse needs of users and applications in a federated context.

Scalability is another concern. Federated systems often comprise a large number of nodes and resources distributed across geographically diverse locations. Solutions for dynamic resource allocation must be capable of efficiently managing allocation at scale, a consideration not explicitly addressed in the UAV task management model.

Moreover, the complexity and overhead introduced by the reallocation strategies and algorithms proposed in the UAV task management model may not be suitable for federation orchestration. Federated systems prioritize simplicity, efficiency, and low overhead in resource allocation mechanisms to ensure optimal scalability.

Lastly, federated systems operate in a distributed environment, where nodes may have limited communication and coordination capabilities. Solutions for dynamic resource allocation in federation orchestration need to account for the distributed nature of the system and ensure that allocation decisions can be made efficiently and effectively in such environments.

In conclusion, while the dynamic reallocation model for multiple UAV tasks provides valuable insights and methodologies for managing resource allocation in specific contexts, it may not provide an optimal solution for the dynamic adjustment of CSP allocations in federation orchestration. Tailored solutions specifically designed for federated environments, considering the unique characteristics and chal-

lenges of such systems, would be more suitable for addressing the complexities of resource allocation in federated environments.

3.5 Santa claus algorithm

The Santa Claus problem is a concept in the context of goods allocation, where Santa Claus has to allocate p gifts to n children with modular preferences. The goal is to maximize the utility of the unhappiest child, which corresponds to the maxmin allocation. This problem is relevant for goods allocation because it demonstrates that even in this restrictive setting, the problem remains NP-hard. This highlights the complexity of allocating goods in situations where the recipients' preferences are modular.[6]

When examining the dynamic adjustment of CSP allocations in federation orchestration, parallels can be drawn with the Santa Claus problem. This problem involves the allocation of gifts by Santa Claus to maximize the utility of the unhappiest child, a concept known as maxmin allocation.

The Santa Claus problem offers insightful parallels within dynamic adjustment of CSP allocations. Dynamic CSP allocations aim to distribute computational resources effectively among federated nodes to maximize system performance. However, the NP-hard nature of the problem suggests that there are formidable computational challenges inherent in dynamically adjusting CSP allocations, much like the task of Santa Claus distributing gifts to optimize the happiness of each child.

The potential for creating algorithms that can provide allocations with minimal envy or close approximations is existing. However, it raises a significant concern: the excessive amount of information transmission required by algorithms, especially in situations with general preferences. Additionally, any deterministic algorithm would require an exponential number of queries to calculate any finite approximation for the minimal envy problem or maxmin allocation.

3.6 A dynamic multiple traveling salesman problem

The comparison between the multiple Traveling Salesman Problem (mTSP) and dynamic adjustment of CSP allocations in federation orchestration reveals intriguing parallels. Both scenarios involve optimizing resource allocation within complex systems.
In the mTSP, the objective is to determine the most efficient sequence of visits for multiple salesmen to a set of cities, minimizing the total distance traveled.[5] Similarly, in federation orchestration, the objective is to dynamically allocate computational resources among federated nodes to optimize system user satisfaction.
The mTSP offers both exact and heuristic solution techniques, including Neural Network-based approaches. However, federation orchestration requires a combination of methods to handle the dynamic nature and scale of resource allocation

challenges.
The mTSP aims to minimize the total distance traveled by salesmen. In contrast, federation orchestration involves optimizing resource utilization, minimizing latency, and maximizing system throughput, among other objectives.
It is important to note that direct application of Traveling Salesman Problem (TSP) algorithms to federation orchestration is not feasible. The TSP is concerned with finding the shortest route that visits each city exactly once. This differs significantly from the dynamics and objectives of federation orchestration.
While insights from mTSP research can inform optimization techniques for dynamic allocation in federated systems, tailored approaches are needed to address the unique challenges of federation orchestration. To effectively manage resource allocation dynamics in federated environments, researchers must explore custom solutions as direct application of TSP algorithms is not suitable.

Chapter 4
The algorithm

4.1 The simulator

The algorithm is an integral part of an existing project called the simulator, serving as its backbone. Every aspect, from the workings of UEs, CSPs to the allocation algorithms, has been outlined and implemented.

The project's core is the initial setup, which provides a visually intuitive representation to understand the way CSPs are allocated and represent how UEs are moving. This visualization offers valuable insights into the allocation process, enabling them to make informed decisions and adjustments as necessary. Additionally, the visual representation is a powerful communication tool, facilitating seamless collaboration and idea exchange among team members.

The base project integrates the oracle, a mechanism designed to evaluate allocations. The oracle considers various parameters that contribute to the quality of an allocation, such as network capacity, resource availability, and user preferences. The oracle generates a utility value by analysing these parameters, serving as a quantitative measure of the allocation's effectiveness.

In order to facilitate analysis, this text abstracts the parameters under consideration, focusing solely on the results provided by the oracle. By simplifying the intricacies of individual parameters, we can adopt a more comprehensive perspective, enabling us to assess allocations in an objective manner. This simplifies the analysis process and enhances the algorithm's scalability and adaptability to diverse scenarios and environments.

4.2 Approach

The approach is predicated on the comparison of utility values, which serve as a foundational element in the assessment and evaluation of the efficacy of disparate allocation solutions. The objective is to identify optimal allocation strategies that maximize utility while minimizing resource consumption and operational costs. This is achieved through rigorous evaluation and benchmarking. The insights

gained from these comparisons can be used to improve and refine the algorithms continuously, ensuring optimal performance in real-world deployment scenarios.

One of the reasons why the random cases approach is preferred. It involves selecting simple random cases and tracking the best changes from them, which ensures that a wide range of potential improvements can be tested with relatively low complexity. This approach also includes resets to aid in escaping local optima and a more thorough exploration of the solution space. Indeed, this technique assists in examining scenarios holistically as well as uncovering unexpected insights and optimizations.

4.3 Structure

An overview of the code structure is given in figure 4.1. When the algorithm is called, it checks whether there are any UEs online at that time. If not, it simply returns an empty allocation. If there are online UEs, the algorithm can start searching for the best allocation. This is done iterative. The number of iterations is hard-coded in advance and can be set depending on how much runtime is available, as more iterations will provide a better allocation. For each iteration, a case is randomly selected. There are nine cases, each making a small or large adjustment to the current allocation. After performing the adjustment, the utility value of the new allocation is queried. If it is better than the previous best utility value, the new allocation is set as the chosen allocation. If there is no improvement, the new allocation is ignored. If the maximum number of iterations is reached, the allocation that currently achieved the highest utility value is returned. Listing 1 gives a code snippet to illustrate the working of this iterative executing random modifications approach as described.

4.4 The cases

4.4.1 Case 1: Swap two UEs between federations

This scenario facilitates the swapping of two UEs between two distinct federations within a network. If the network consists of only one federation, the algorithm remains inactive as there is no need for any swaps. However, when the network comprises multiple federations, the algorithm selects two federations from the pool at random, ensuring that they both host the same application. This selection criterion is crucial because in order for UEs to be swapped between federations, they must possess the same application as the federation they are destined for. Once two federations with identical applications are identified, the algorithm proceeds to the next step.

Within each selected federation, the algorithm randomly selects a UE. These UEs are then exchanged between the federations, effectively executing the swap operation. The algorithm maintains a fair and unbiased approach to the swapping

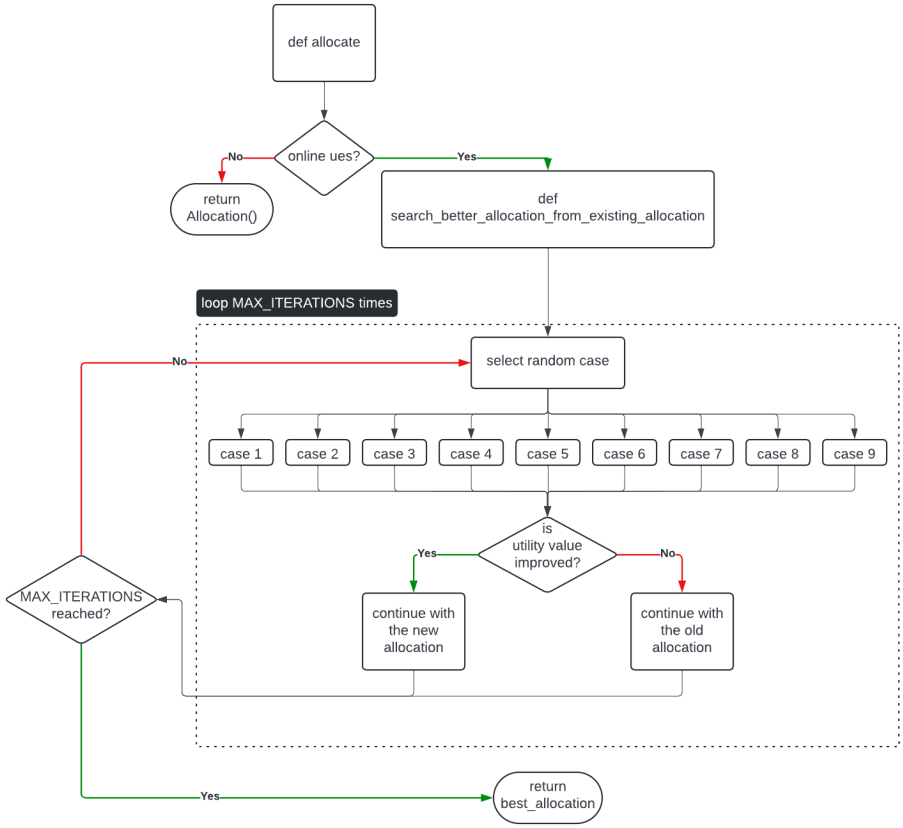


Figure 4.1: Visual representation of the code structure

process by ensuring randomness in both the selection of federations and UEs.

This method not only facilitates the efficient redistribution of UEs across federations but also ensures compatibility between the UEs and the applications hosted by their respective federations. Additionally, the random selection process helps to prevent favouritism or bias, promoting fairness and equity in the allocation of resources within the network.

4.4.2 Case 2: Swap two CSPs between federations

In this scenario, the algorithm facilitates the exchange of two CSPs between two distinct federations within a network. The algorithm remains inactive if the network consists of only one federation, as there would be no need for any CSP swaps. However, once the network expands to encompass multiple federations, the algorithm becomes active.

The process starts by randomly selecting two federations from the pool. The algorithm ensures an unbiased approach to the swapping operation by randomly selecting two federations. Once identified, the algorithm proceeds to the next step.

Within each selected federation, a CSP is randomly chosen. These CSPs are then exchanged between the federations, effectively executing the swap operation. The algorithm maintains fairness and impartiality throughout the swapping process by ensuring randomness in both the selection of federations and CSPs.

This method facilitates the redistribution of CSPs across federations and ensures diversity and equilibrium in resource allocation within the network. The random selection process also helps to mitigate the risk of bias or favoritism, promoting transparency and equality in the management of communication services within the network ecosystem.

4.4.3 Case 3: Delete a federation

To remove a federation, the process starts by randomly selecting one from the allocation. Next, a list of receiving federations is generated. This step is crucial because the UEs from the soon-to-be-removed federation require a new federation with CSPs to serve them. A receiving federation must meet the qualification of serving UEs with the same application. If no such federation is found, the removal process halts, and the randomly selected federation remains untouched.

However, if there are any receiving federations available, all UEs scheduled for relocation will be systematically removed from their original federation one by one, using a for loop. Subsequently, they will be added to a randomly selected federation from the list of receiving federations. Once the UEs have been relocated, the CSPs will be redistributed among the receiving federations to ensure optimal resource allocation and service delivery.

Finally, once the UEs and CSPs have been successfully reallocated, the federation can be safely removed from the network. This meticulous process ensures that the removal of a federation does not disrupt the continuity of service for the UEs and maintains the efficiency and functionality of the network as a whole.

4.4.4 Case 4: Add a new federation

The process starts by creating a new federation object. This object is then passed to a function that attempts to add UEs to it.

The function responsible for this task begins by randomly selecting the application of the new federation from the existing federations within the network allocation. The function identifies potential donating federations within the allocation that serve UEs with the same application as the newly created federation. If more than one such federation is available, the function proceeds with the allocation process.

To determine the number of UEs to allocate to the new federation, the function calculates the allocation parameters, considering both the total number of online UEs and the number of federations in the allocation. This method ensures a fair distribution of UEs across the network.

To allocate each UE, the function randomly selects a donating federation from a list of potential donors. It then randomly chooses a UE from the selected donating federation, removes it from the donor federation, and adds it to the new federation. Once the UEs have been allocated, the function updates the allocation by removing any donating federations that become empty. It then reallocates the CSPs that were originally assigned to the removed federation to the remaining federations.

If any UEs are found during this process, the federation also receives CSPs through the `allocate_csps_to_new_fed` function. The function first determines the total number of federations in the network allocation, which is crucial for subsequent calculations. If there are no federations present, indicating the initialization of the network or the absence of any federations, all CSPs are allocated directly to the newly created federation. Otherwise, the function calculates the number of CSPs to allocate to the new federation, ensuring a balanced distribution.

To allocate each CSP, the function randomly selects a donating federation from the existing ones in the allocation. It then randomly chooses a CSP from the selected federation, removes it from the donor federation, and adds it to the new federation while maintaining equilibrium. This process ensures that CSPs are only allocated from federations with available resources.

Finally, the newly created federation, which includes UEs and CSPs, is added to the allocation, completing the allocation process.

4.4.5 Case 5: Swap all CSPs between two federations

The function `swap_all_csps_between_federations` exchanges all CSPs between two federations within a given allocation. It operates by receiving an allocation object as input, which represents the current state of resource allocation among federations.

It operates by receiving an allocation object as input, which represents the current state of resource allocation among federations. It operates by receiving an allocation object as input, which represents the current state of resource allocation among federations. The function then calls `Reallocator.get_two_different_federations(allocation)`. The Reallocator class implements this method. It selects two distinct federations, fed1 and fed2, from the allocation.

The function then checks if fed2 is not None, ensuring that there are at least two different federations in the allocation. If this condition is met, the function proceeds with the CSP swapping operation.

To prevent any changes to the original CSP lists linked to fed1 and fed2, the function generates duplicates of these lists, referred to as `csps_fed1` and `csps_fed2`, respectively.

Then, the function exchanges the CSP lists between fed1 and fed2 by assigning `csps_fed2` to `fed1.csps` and `csps_fed1` to `fed2.csps`. This process effectively swaps all CSPs between the two federations.

4.4.6 Case 6: Move a CSP to another federation

The function `move_csp_from_one_federation_to_another` enables the transfer of a CSP from one federation to another within a new allocation.

The process involves selecting two distinct federations, fed1 and fed2, from the provided `new_allocation` object. To select the appropriate source and destination federations for the CSP transfer, the `Reallocator.get_two_different_federations(new_allocation)` method is to be called.

The function then checks two conditions before proceeding with the transfer. Firstly, it ensures that fed2 is not None, indicating the presence of at least two distinct federations in the allocation. Secondly, the function checks whether the source federation (fed1) has at least one CSP available for transfer, as indicated by the condition `len(fed1.csps) != 0`.

Once these conditions are met, the function initiates the CSP transfer process by randomly selecting a CSP from the list of CSPs associated with fed1 using the

random.choice function. This random selection ensures an unbiased choice, providing equal opportunities for all CSPs within fed1 to be considered for transfer.

The CSP is transferred from fed1 to fed2 by first removing it from fed1 using the `fed1.remove_csp(csp_to_move)` method, and then adding it to fed2 using the `fed2.add_csp(csp_to_move)` method. This ensures that the CSP is relocated without any loss of data or information. The CSP is transferred from fed1 to fed2 by first removing it from fed1 using the `fed1.remove_csp(csp_to_move)` method, and then adding it to fed2 using the `fed2.add_csp(csp_to_move)` method. This operation transfers the CSP from its original federation (fed1) to the designated destination federation (fed2), allowing for dynamic adjustments in resource allocation strategies within the federated environment.

4.4.7 Case 7: Remove a CSP from the allocation

In some cases, it may be advantageous to not assign a CSP to a federation. This choice could be driven by factors such as energy efficiency and resilience. By refraining from assigning a CSP to a federation, energy consumption can be reduced, which contributes to overall energy efficiency. In situations where resilience is crucial, avoiding CSP allocation can improve the federation's ability to withstand and recover from disruptions or failures. This can enhance system robustness and reliability.

The `remove_csp_from_allocation` function works on a `new_allocation` object with the goal of eliminating a CSP from the allocation. It begins by randomly selecting a federation from the list of federations within the new allocation and storing it in the variable `random_federation`. Next, it randomly selects a CSP from the list of CSPs associated with the chosen random federation. The CSP that was selected is removed from the random federation, effectively eliminating it from the federation's allocation. The function then checks if the selected federation has any remaining CSPs. If the length of the federation's list of CSPs equals zero, indicating that it has no CSPs, the federation itself is removed from the new allocation.

Essentially, this case 7 selects a federation at random and removes a CSP from it, chosen at random from the provided allocation. If this removal leaves the federation with no remaining CSPs, the function then removes the federation from the allocation.

4.4.8 Case 8: Add an unused CSP to the allocation

In the `add_unused_csp_to_allocation` function, a CSP that hasn't been allocated yet is added to a given allocation. Firstly, the function calls `Reallocator.get_unused_csps`, a method within the Reallocator class, to obtain a list of CSPs that haven't been allocated in the provided new allocation. This list is stored in the variable `unused_csps`.

If there are unused CSPs available (i.e., if the length of `unused_csps` is greater

than zero), the function proceeds with the allocation process. A CSP is randomly chosen from the list of unused CSPs using the `random.choice` function, and it's assigned to the variable `adding_csp`. Similarly, a federation is randomly selected from the list of federations within the new allocation object, and it's stored in the variable `adding_federation`.

The selected CSP is then added to the chosen federation using the `add_csp` method, effectively allocating the CSP to the chosen federation.

In summary, the `add_unused_csp_to_allocation` function ensures that an unused CSP is randomly assigned to a federation within the provided allocation, thus optimizing resource utilization.

4.4.9 Case 9: Add all unused CSPs to the allocation

Case 9 is executed by a function that accepts two parameters: `new_allocation` and `csps`. The former represents a recent allocation of resources, while the latter refers to the resource units or entities that require allocation.

The first step is to calculate the unused CSPs by using the `get_unused_csps()` method from the Reallocator class. This method identifies the resource units that have not been allocated in the new allocation. Then, the function iterates through each CSP in `unused_csps` and randomly selects one federation from the `new_allocation.federations` list. This is because new_allocation has a collection of federations to which resources can be allocated. Finally, the `add_csp()` method assigns the current CSP to the randomly selected federation. This step redistributes the unused resource units among the federations randomly.

4.4.10 Complexity

The algorithmic complexity of the given code can be understood by evaluating the efficiency of the reallocation algorithm implemented within the Reallocator class, focusing in particular on the `search_better_allocation_from_existing_allocation` method at the heart of the process.

The algorithm follows an iterative approach, constrained by a preset maximum number of iterations defined by `MAX_ITERATIONS`. This iterative nature is the primary driver of the algorithm's time complexity. In the worst case, where the loop runs for the maximum number of iterations, the time complexity becomes linear with respect to `MAX_ITERATIONS`, and is thus denoted as O(n), where n is the number of iterations.

Within each iteration, the algorithm randomly selects from a set of predefined reallocation cases, each of which involves different operations on the federations, UEs and CSPs. The complexity of these operations may vary. For example, swapping UEs or CSPs between federations involves accessing and modifying sets, which typically has a time complexity of O(1) for each addition or removal operation.

However, if the operation involves searching these sets, the complexity could increase to O(k), where k is the number of elements in the set.

Furthermore, after each reallocation operation, the utility of the new allocation is recalculated using a potentially complex utility function provided by the Oracle class. The complexity of this calculation is will not be dived in this project, but it is crucial to be aware as it directly affects the overall efficiency of the algorithm. If the utility calculation is complex, it can dominate the time complexity of each iteration.

Another point to consider is the memory requirements of the algorithm. While space complexity is generally less of a concern in the context of reallocation algorithms, it is worth noting that this algorithm creates a copy of the allocation at each iteration for comparison with the best allocation found so far. This duplication implies a space complexity that is linear in the size of the allocation data structure, denoted as O(m), where m is the number of elements within the allocation.

Finally, the complexity of the helper functions used within each case should not be underestimated. For example, functions that determine whether a CSP is unused or that find federations with a particular property add their own layer of complexity, which may involve iterating over all federations or all CSPs.

In summary, the overall time complexity of the reallocation algorithm is primarily influenced by the number of iterations and the complexity of the utility computation, with a linear relationship to `MAX_ITERATIONS` and potentially to the size of the federations, UEs and CSPs involved in the utility computation. Space complexity, although secondary, is also linearly related to the size of the allocation data structure due to the need to maintain copies of allocations during the iterative process.

```
for i in range(Reallocator.MAX_ITERATIONS):
    random_case = random.choice(list(ReallocateCases))
    new_allocation = best_allocation.make_copy()

    match random_case:
        case ReallocateCases.SWAP_UE:
            Reallocator.swap_ue_between_federations(
            new_allocation)
        case ReallocateCases.SWAP_CSP:
            Reallocator.swap_csp_between_federations(
            new_allocation)
        case ReallocateCases.DELETE_FED:
            Reallocator.delete_federation(
            new_allocation)
        case ReallocateCases.ADD_NEW_FED:
            Reallocator.add_new_federation(
            env, new_allocation, ues, csps)
        case ReallocateCases.SWAP_ALL_CSPS:
            Reallocator.swap_all_csps_between_federations(
            new_allocation)
        case ReallocateCases.MOVE_CSP:
            Reallocator.move_csp(
            new_allocation)
        case ReallocateCases.REMOVE_CSP:
            Reallocator.remove_csp_from_allocation(
            new_allocation)
        case ReallocateCases.ADD_CSP:
            Reallocator.add_unused_csp_to_allocation(
            new_allocation, csps)
        case ReallocateCases.ADD_ALL_CSPS:
            Reallocator.add_all_unused_csps_to_allocation(
            new_allocation, csps)

    new_utility = Oracle.calculate_allocation_utility(
    new_allocation)
    if new_utility > best_utility + 0.001:
        best_utility = new_utility
        best_allocation = new_allocation.make_copy()
```

Listing 1: Loop in reallocator.py

Chapter 5
Testing

This chapter presents the testing conducted to evaluate the robustness and performance of the resource allocation system in our federated environment. The objective of the testing phase is to assess the responsiveness and adaptability of the system to a range of scenarios, including changes in the number of CSPs and the number of UEs in the environment.

The important value on which all tests are based is the utility value calculated by oracle. This value always determines the quality of the allocations and is thus a direct indication of whether an allocation is improved or not.

The greater part of the runtime is primarily attributed to the substantial computing time required by the oracle. Since each iteration requests a value from the oracle, a higher number of iterations is a significant addition to the runtime.

The tests were carried out with configuration files representing real situations as closely as possible. There are files for UEs, CSPs, applications, the environment and general variables. An example of an UE configuration file is listed in listing 2. The python project from which this project builds further on consisted already of certain configuration files. All these upcoming tests are executed with these or variations on these files so no new configuration files were constructed for this project.

5.1 General Results

In this part, the algorithm is tested on overall is tested for overall utility performance in two scenarios: one with 100 iterations and the other with 1000 iterations. It is expected that more iterations will result in higher utility value.

In the 100-iteration scenario, depicted in figure 5.1a, we observe significant fluctuations in utility values over time. The utility value at each time point after improved algorithm allocation is significantly higher. It is visible that after each time point, due to the dynamic nature of the UEs, there is a drop in the value so although it is possible to get the values higher, there is the same amount in utility to make up each time.

In the 1000 iteration scenario, figure 5.1b, a similar trend in the dynamics of the utility values is observed. Thanks to the increased number of iterations, the minima and maxima of the new utility value curve reach higher numbers.

For a direct visual confirmation of the effectiveness of a dynamic approach for reallocation, figure 5.2 is added. This test is executed with the `MAX_ITERATIONS` variable set on 1000.
The dynamic optimisation strategy shows immediately higher utility values compared to the non-dynamic approach. Since the non-dynamic algorithm keeps fluctuating in all directions, it is incidentally random-based, the curve of dynamic reallocation shows more consistency and invariably achieves high utility values.

5.2 Tests on different configuration files

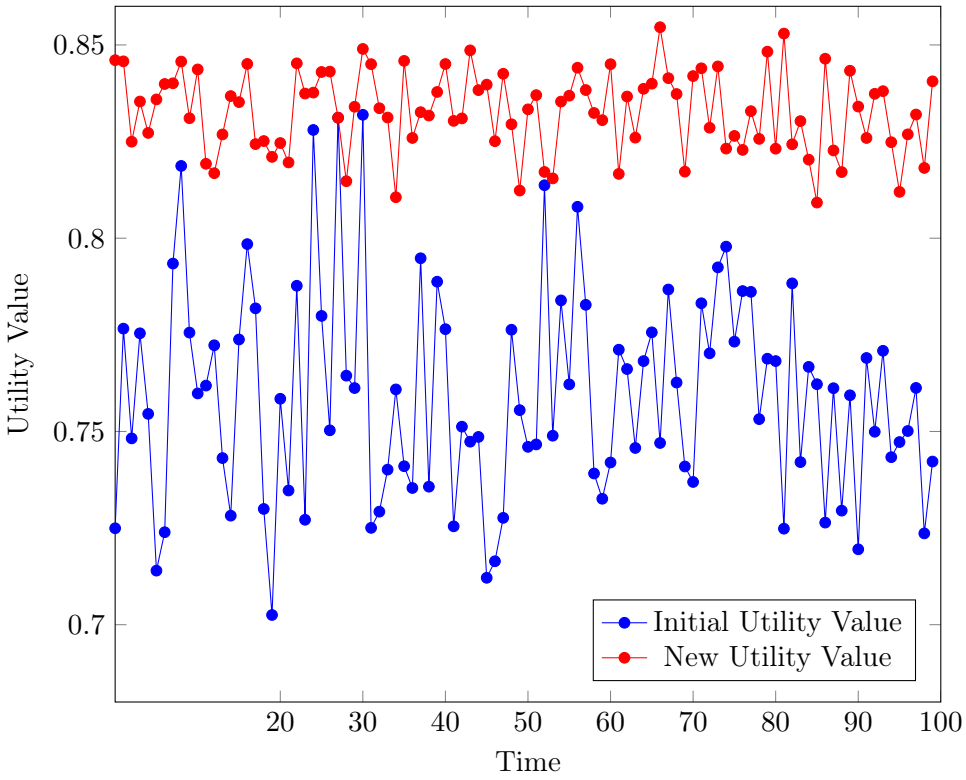
5.2.1 Increased amount of CSPs

It is hypothesized that an increase in the number of CSPs would lead to an increase in utility. This hypothesis is supported by figure 5.3, which demonstrates that when more goods are available, agents are served better.

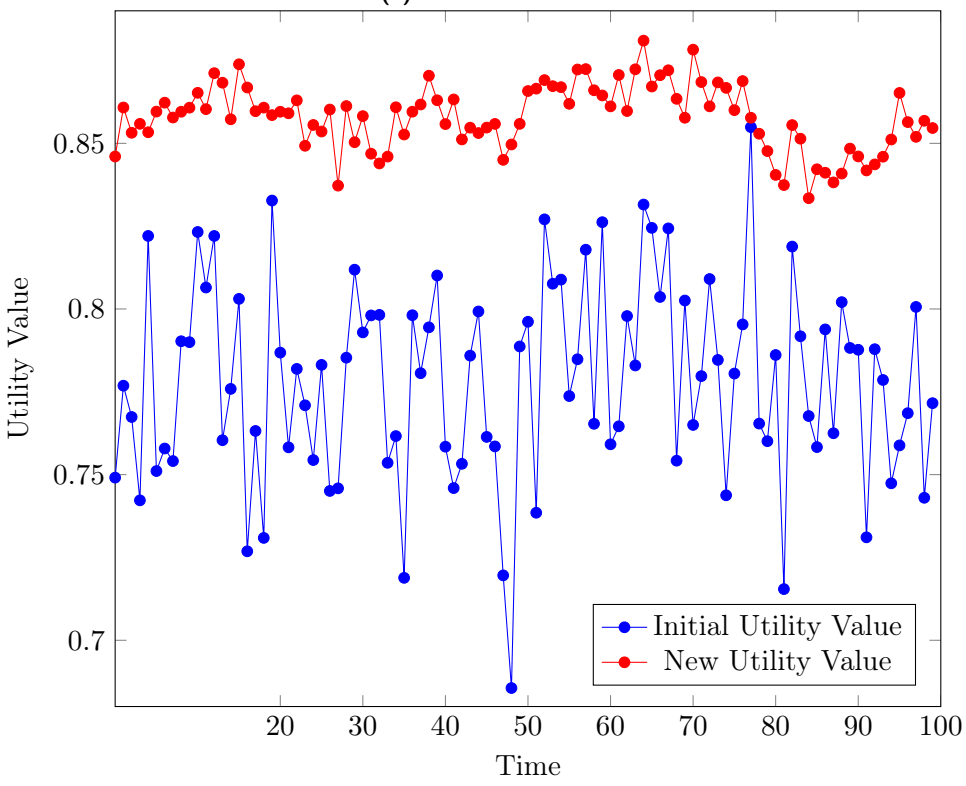
It is noteworthy that, despite the reduction in the quantity of CSPs, utility remained relatively stable. This stability persisted despite the inherently more challenging nature of the environment, where the algorithm encountered fewer available CSPs. Remarkably, we found that utility never plummeted to significant lows under these circumstances. This resilience in utility amidst decreased CSP quantity suggests several promising outcomes. Firstly, this underscores the algorithm's adaptability in navigating an array of problem instances. Secondly, it demonstrates the resilience of our decision-making processes, which were able to maintain utility levels despite increased complexity. Furthermore, the consistent performance of the algorithm highlights its reliability and effectiveness in handling a larger workload.

5.2.2 Increased amount of UEs

In the subsequent testing phase, where the algorithm's performance was assessed with a reduced number of UEs, analogous conclusions were reached. Notably, it was observed in figure 5.4 that when fewer agents needed to be served, the algorithm consistently yielded higher utilities. However, a distinctive pattern emerged when comparing the impact of varying quantities of CSPs versus UEs. The stability of the system remains unaltered when the quantity of CSPs is modified, whereas a change in the quantity of UEs results in a notable shift in stability. The graph illustrates a greater disparity in measurements for four UEs than for fewer UEs. It is noteworthy that during the testing of varying CSP quantities, four UEs were consistently employed.



(a) 100 iterations



(b) 1000 iterations

Figure 5.1: Comparison of initial and new utility values over time for 100 and 1000 iterations

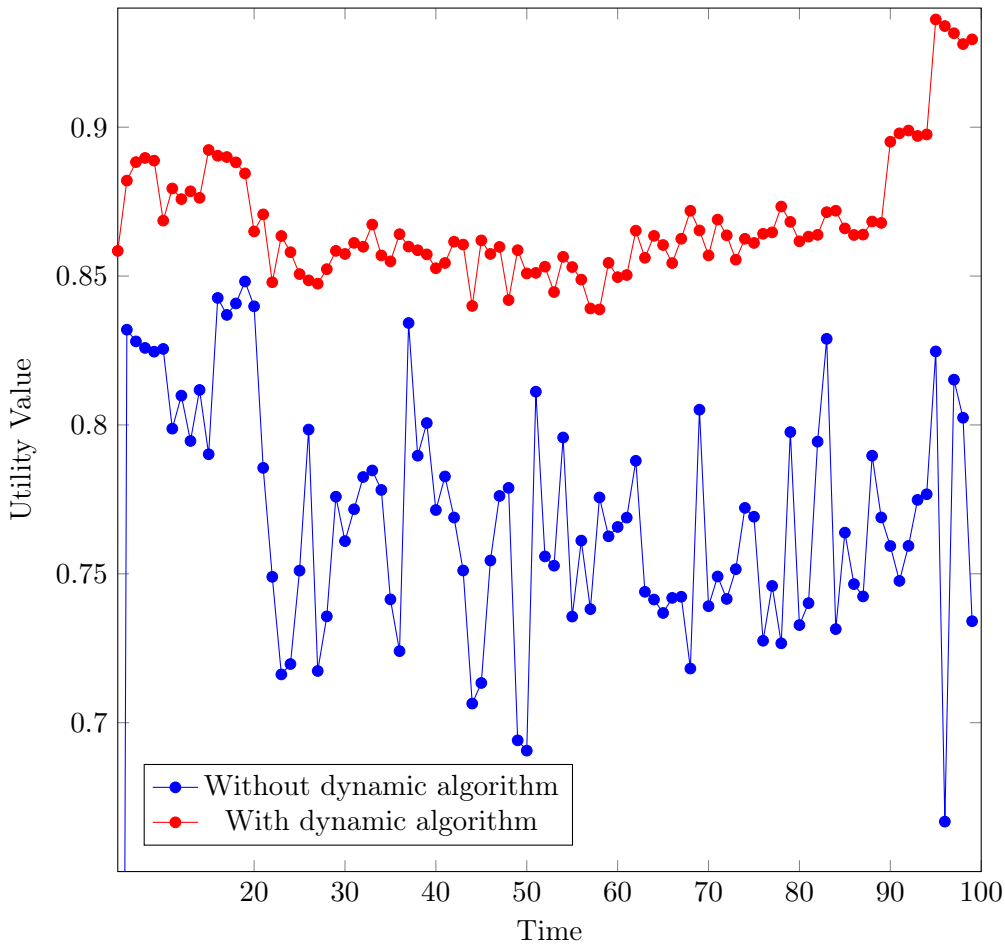


Figure 5.2: Utility values over time for 1000 iterations with and without the dynamic algorithm

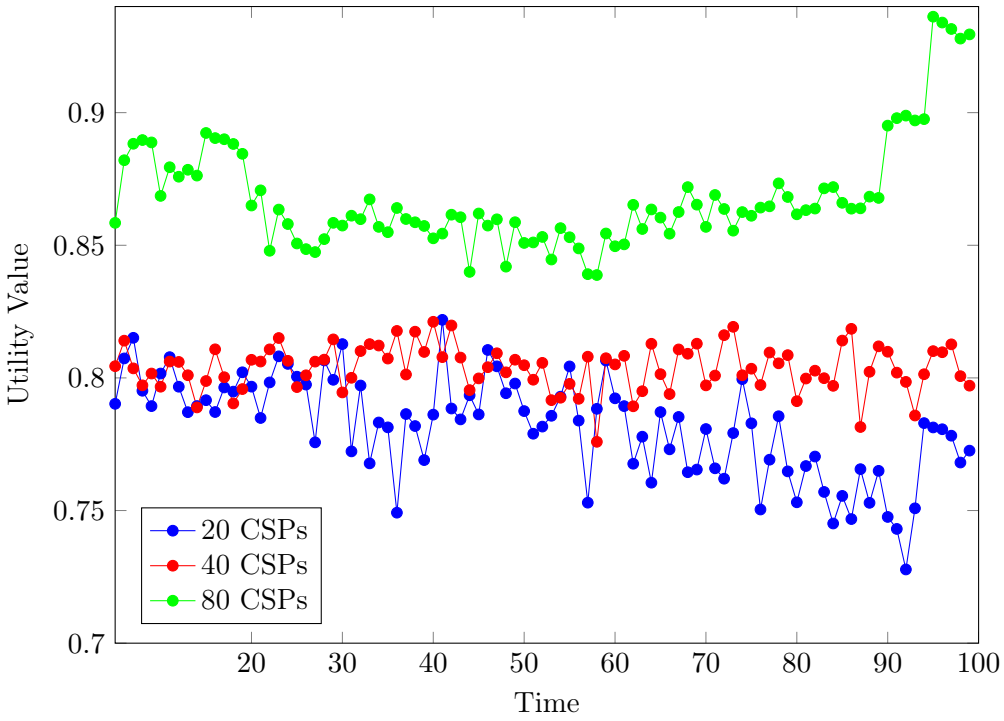


Figure 5.3: Utility values over time for 1000 iterations for 20, 40 and 80 CSPs

This observation suggests that the number of UEs plays a defining role in determining the complexity of the allocation process when employing this algorithm. Despite the increased complexity associated with a higher number of UEs, it is noteworthy that even in scenarios with the maximum number of UEs, utility consistently remained above a certain threshold, never dropping below 0.8. This resilience indicates the algorithm's robustness and its capacity to maintain satisfactory levels of utility across a spectrum of challenging conditions.

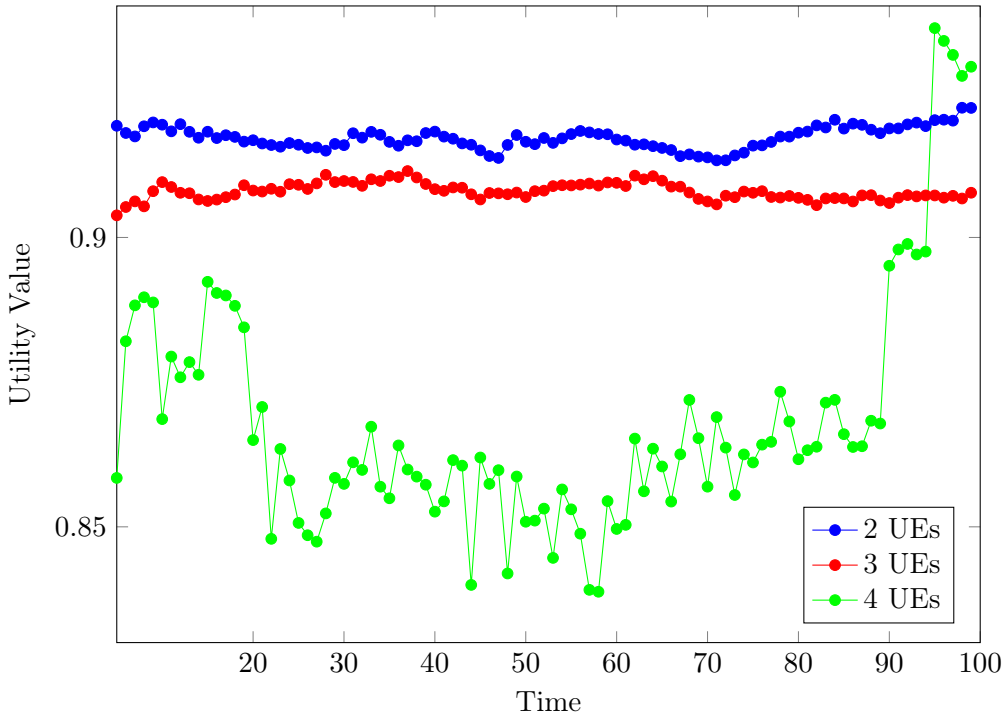


Figure 5.4: Utility values over time for 1000 iterations for 2, 3 and 4 UEs

```yaml
id: 0
coords: [6.394267984578837, 0.25010755222666936,
 ↪  0.4125439775536789]
application: application_0
speed: 0.1
online: 0
offline: 9999999999999
utility_check_interval: 60
trajectory:
    - [1.395379285251439, 1.024951761715075,
     ↪  1.1110016170015138, 8]
    - [0.8693883262941615, 4.2192181968527045,
     ↪  0.044695829157105516, 3]
    - [2.326608933907396, 6.020187290499804,
     ↪  0.8418675944079195, 10]
    - [7.01324973590236, 4.195198209616588,
     ↪  0.6738135694257804, 4]
    - [8.094304566778266, 0.06498759678061017,
     ↪  1.2087288777492118, 6]
    - [3.4025051651799187, 1.5547949981178155,
     ↪  1.4358196083101717, 5]
    - [1.022102765198487, 3.7992730063733737,
     ↪  0.5384690707269426, 5]
    - [6.037260313668911, 8.071282732743802,
     ↪  1.0945976800407269, 8]
    - [1.2482616285320935, 9.222953720281598,
     ↪  0.11820029711768726, 4]
    - [8.294046642529949, 6.185197523642461,
     ↪  1.2925603504661658, 9]
    - [1.922885902566599, 0.6955514882374092,
     ↪  0.9918949778515652, 4]
    - [9.852215206607578, 8.55317721015147,
     ↪  1.2997255001329044, 6]
    - [2.779736031100921, 6.356844442644002,
     ↪  0.5472482684551263, 5]
    - [3.552707002275215, 6.701751743776933,
     ↪  1.052730470187827, 10]
    - [6.480353852465935, 6.091310056669882,
     ↪  0.2567079722971455, 3]
    - [1.634024937619284, 3.794554417576478,
     ↪  1.4842850259548928, 10]

```

Listing 2: UE configuration file

Chapter 6
Conclusion

6.1 Project result

The dynamic adjustment of CSP allocations in federation orchestration has led to advancements in resource management and optimization within the REINDEER projects use case. The proposed solution has yielded several notable outcomes, including enhanced resource utilisation. The algorithm enables federations to reallocate CSP allocations in a manner that is responsive to changing workload demands and resource availability. This flexibility has resulted in a significant improvement in the efficiency of resource utilisation throughout the federation, ensuring optimal allocation of resources to meet operational requirements.
The simulator project serves as the foundation of the algorithm, seamlessly integrated to handle various aspects such as UEs, CSPs and allocation strategies. The oracle evaluates allocations based on parameters like network capacity and user preferences, generating utility values as a measure of effectiveness. The approach focuses on maximizing utility while minimizing resource consumption, employing random case selection for testing improvements efficiently. The algorithm iteratively selects random cases to modify allocations, optimizing utility values based on predefined criteria. This iterative process ensures flexibility and robustness in handling allocation scenarios. The algorithm's structure facilitates iterative adjustments to allocations, optimising utility values through random modifications. The cases encompass various allocation scenarios, including swapping UEs and CSPs between federations, adding or removing federations, and redistributing resources to optimise the allocation. The algorithm's complexity lies in its iterative nature, influenced by the number of iterations, the complexity of utility computation, and the size of the allocation data structure.

Extensive experimentation and evaluation have validated the effectiveness and efficiency of our dynamic adjustment approach. The testing assesses the system's responsiveness to various scenarios, such as changes in the number of CSPs and UEs. The utility value, determined by an oracle, serves as a benchmark for the quality of allocations. The runtime is primarily attributed to the substantial computing time required by the oracle and thus by the high amount of calling the oracle. Testing is conducted using configuration files that represent real situations as closely as possible. The results of the testing indicate that there are signifi-

cant fluctuations in utility values over time, with higher iteration amounts leading to higher utility values. Dynamic allocation strategies consistently outperform non-dynamic ones, showcasing higher utility values. Tests with varying CSPs and UEs demonstrate that there is a positive correlation between the number of goods (CSPs) and the quality of service for agents. Even with fewer CSPs, utility remains relatively stable, showcasing the algorithm's adaptability and robustness. Conversely, the number of UEs has an impact on system stability. In general, higher numbers of UEs result in greater fluctuations. However, utility consistently remains above a certain threshold, indicating the algorithm's effectiveness across diverse conditions.

In conclusion, the project results demonstrate the transformative impact of dynamic allocation adjustment in federation orchestration. These findings provide a foundation for further advancements in resource management and optimization within federated systems.

6.2 Future work

Although the current algorithm has demonstrated its effectiveness, there are several avenues for future enhancement to address existing limitations and improve overall performance. One significant limitation is the constraint that UEs within the same federation must have identical applications. Future iterations of the reallocation algorithm could explore methods to relax this constraint to allow heterogeneous applications between UEs while still ensuring system operation. This would add significant flexibility to construct robust allocations.

One approach to enhance the reallocation algorithm would be to introduce a mechanism to assign higher weights to critical cases. This would enable more informed resource allocation decisions, thereby optimising system efficiency. The objective would be to have the more improving cases being called more often than other cases.

Future iterations of the algorithm could explore alternative evaluation approaches, such as event-triggered evaluations, to better adapt to changing environments and optimise resource allocation over time.

As utility values increase, the computational complexity of the reallocation algorithm can escalate, resulting in longer run times and reduced responsiveness. Therefore, it is of paramount importance to investigate methods to reduce run times for computations with higher utility values per second. Techniques such as algorithmic optimisation, parallel processing or hardware acceleration can be employed to streamline computation and improve reallocation efficiency without compromising accuracy.

6.3 Endword

The project has successfully developed and evaluated a dynamic reallocation algorithm tailored for federation orchestration within the REINDEER project framework. The algorithm dynamically reallocates resources in order to ensure that the desired performance is maintained in response to fluctuating demands and conditions. The implementation of the solution demonstrated significant improvements in throughput and latency, thereby confirming the hypothesis that dynamic reallocation can effectively meet the challenges of next-generation network environments. As the project draws to a close, it is appropriate to reflect on the accomplishments and the lessons learned. Future work will concentrate on enhancing the algorithm's functionality and investigating more profound integrations with other components of the REINDEER project. This project not only advances our comprehension of federation orchestration but also provides a pragmatic solution to the ongoing challenges in managing complex network infrastructures.

References

[1] Georgios Amanatidis et al. "Fair division of indivisible goods: A survey". In: *arXiv preprint arXiv:2202.07551* (2022).

[2] Alvin AuYoung et al. "Resource allocation in federated distributed computing infrastructures". In: *Proceedings of the 1st Workshop on Operating System and Architectural Support for the On-demand IT InfraStructure*. Vol. 9. 2004.

[3] Yossi Azar, Niv Buchbinder, and Kamal Jain. "How to allocate goods in an online market?" In: *European Symposium on Algorithms*. Springer. 2010, pp. 51–62.

[4] Siddhartha Banerjee et al. "Online nash social welfare maximization with predictions". In: *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM. 2022, pp. 1–19.

[5] Tolga Bektas. "The multiple traveling salesman problem: an overview of formulations and solution procedures". In: *omega* 34.3 (2006), pp. 209–219.

[6] Sylvain Bouveret et al. *Fair Allocation of Indivisible Goods*. 2016.

[7] MohammadTaghi Hajiaghayi et al. "Online Algorithms for the Santa Claus Problem". In: *arXiv preprint arXiv:2210.07333* (2022).

[8] Ian Kash, Ariel D Procaccia, and Nisarg Shah. "No agent left behind: Dynamic fair division of multiple resources". In: *Journal of Artificial Intelligence Research* 51 (2014), pp. 579–603.

[9] REINDEER. "D3.2: Methods for Communication and Initial Access with RadioWeaves". In: 2022, pp. 48–52.

[10] REINDEER. *Resilient interactive applications through hyper diversity in energy efficient radioweaves technology*. URL: `https://reindeer-project.eu/`.

[11] Jun Tang et al. “Dynamic reallocation model of multiple unmanned aerial vehicle tasks in emergent adjustment scenarios”. In: *IEEE Transactions on Aerospace and Electronic Systems* 59.2 (2022), pp. 1139–1155.

[12] Kuo-Hui Yeh. “An efficient resource allocation framework for cloud federations”. In: *Information Technology and Control* 44.1 (2015), pp. 64–76.

Printed by Tryckeriet i E-huset, Lund 2024

LUND
UNIVERSITY

Series of Master´s theses
Department of Electrical and Information Technology
LU/LTH-EIT 2024-993
http://www.eit.lth.se