Technische Universität München
TUM School of Computation, Information and Technology

# Design, Implementation, and Evaluation of Network Resource Allocation and Reconfiguration Algorithms

*Amir Varastehhajipour*

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften (Dr.-Ing.)

genehmigten Dissertation.

| | | |
|---|---|---|
| Vorsitz: | | Prof. Dr.-Ing. Klaus Diepold |
| Prüfer*innen der Dissertation: | 1. | Priv.-Doz. Dr.-Ing. habil. Carmen Mas-Machuca |
| | 2. | Prof. Dr.-Ing. Andreas Kirstädter |

Die Dissertation wurde am 08.06.2022 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 30.01.2023 angenommen.

# Design, Implementation, and Evaluation of Network Resource Allocation and Reconfiguration Algorithms

Amir Varastehhajipour

30.01.2023

# Abstract

Communication networks have become one of the primary critical players of our society, connecting users to data centers, data centers to users and each other, and more recently, even *things* to the Internet. These advancements have brought the *connectivity* to every corner of our lives, homes, streets, and even trains and airplanes. This vast and boundless *connectivity* can bring many online and interactive services to our everyday life, such as (ultra-) high-definition video streaming, gaming, and data-driven analytics. However, the communication systems were initially designed for providing simple best-effort connectivity with long-term manual changes, if necessary. However, this approach does not seem flexible enough for the modern new applications and services, which now impose additional requirements for the underlying infrastructure, such as strict Quality of Service (QoS) requirement, and dynamicity of traffic, and high user mobility.

Recently, there was a shift from statically and manually-configured networks towards *softwarized networks*. In this context, promising technologies like Software-Defined Networking (SDN) and Network Function Virtualization (NFV) govern the network by providing flexibility, and manageability while allowing it to cope with the heterogeneity and high variability of today's applications and communications. Generally, emerging applications require this existing underlying infrastructure to provide computational and network resources to more dynamic and heterogeneous services, applications, and requirements. The resource allocation and management tasks can have significant effects on the delivered QoS (e.g., end-to-end delay) to the end-users, as well as the related Operational Expenditures (OPEX) to the network providers (e.g., power consumption). Therefore, in this thesis, we focus on the design and optimization of such approaches, focusing on cost, end-to-end delay, and reconfigurability.

The user requests arrive on the fly from a specific node in the network and need to be routed through the network to reach the required service endpoint. These services are usually deployed virtually (e.g., as Virtual Machines (VMs)) by geographically-distributed Data Centers (DCs). Some services even require more complex setups, such as Service Function Chaining (SFC) where the traffic should pass through a sequence of Virtual Network Functions (VNFs). Thus, to offer QoS-guaranteed and cost-efficient services, determining the location of VNFs, as well as routing decisions, seem a challenging problem to address. Therefore, designing, modeling, and optimizing such resource allocation strategies is the focus of this work. In this thesis, we present a cost-efficient and QoS-aware algorithm, with low computational complexity, that can perform online resource allocation for SFC requests. Our comprehensive evaluation results confirm the performance of the proposed mechanisms in a realistic online network setup with heterogeneous infrastructure and user requirements.

Communication networks are becoming more and more dynamic every day. Given the different network endpoints (e.g., humans, machines) and the exponential growing variety and use of services and applications, the networks show more dynamic behaviors. This dynamicity can have different means and can create challenging problems for the network operators. A form of the dynamic behavior of networks can be due to unexpected traffic changes. As a result, the infrastructure resources can get overloaded, leading to inevitable performance loss for the users, and eventually, decreasing profits for network providers. Network reconfiguration decisions such as the deployment of new Lightpaths (LPs) on the network can help to cope with the traffic increases. Another source of dynamicity can be the mobility of users. In scenarios such as Mobile Edge Computing (MEC), the services are deployed in the proximity of the users. Hence, the movement of the user can have an undeniable impact on the received QoS. In practice, when a user moves farther away from the currently used MEC server, the routing cost can increase, inducing more delay. A network reconfiguration can be triggered, for instance, migration of the service endpoint to a closer MEC server towards the movement of the user. This action can lead to improving the QoS of the user. In these cases, the resource allocation framework's responsibility is to trigger and perform network reconfiguration decisions, accounting for their cost and characteristics in the account. Hence, on top of the resource allocation approaches, this dissertation aims to design and optimize network reconfiguration algorithms in situations with unexpected traffic changes and user mobility. Existing algorithms are investigated and evaluated against the newly proposed algorithms. The evaluations in various scenarios and settings indicate improvements in the runtime, optimality, and completeness compared to the state-of-the-art algorithms.

# Kurzfassung

Kommunikationsnetzwerke sind zu einem der wichtigsten kritischen Akteure unserer Gesellschaft geworden und verbinden Benutzer mit Rechenzentren, Rechenzentren mit Benutzern und untereinander und in jüngster Zeit sogar *Gegenstände* mit dem Internet. Diese Fortschritte haben die *Konnektivität* in jeden Winkel unseres Lebens gebracht, in Häusern, Straßen und sogar in Zügen und Flugzeugen. Diese riesige und grenzenlose *Konnektivität* kann viele Online- und interaktive Dienste in unseren Alltag bringen, wie (Ultra-) High-Definition-Video-Streaming, Spiele und datengesteuerte Analysen. Die Kommunikationssysteme waren jedoch ursprünglich so konzipiert, dass sie eine einfache Best-Effort-Konnektivität mit langfristigen manuellen Änderungen bei Bedarf bieten. Dieser Ansatz scheint jedoch nicht flexibel genug für die modernen neuen Anwendungen und Dienste zu sein, die nun zusätzliche Anforderungen an die zugrunde liegende Infrastruktur stellen, wie z. B. strenge Dienstgüte-Anforderungen, Dynamik des Datenverkehrs und hohe Mobilität der Benutzer.

In letzter Zeit gab es eine Verschiebung von statisch und manuell konfigurierten Netzwerken hin zu *softwarized networks*. In diesem Zusammenhang steuern vielversprechende Technologien wie Software-Defined Networking (SDN) und Network Function Virtualization (NFV) das Netzwerk, indem sie Flexibilität und Verwaltbarkeit bieten und es ihm ermöglichen, die Heterogenität und hohe Variabilität heutiger Anwendungen und Kommunikationen zu bewältigen. Im Allgemeinen erfordern neue Anwendungen diese vorhandene zugrunde liegende Infrastruktur, um Rechen- und Netzwerkressourcen für dynamischere und heterogenere Dienste, Anwendungen und Anforderungen bereitzustellen. Die Ressourcenzuweisungs- und Verwaltungsaufgaben können erhebliche Auswirkungen auf die gelieferten Quality of Service (QoS) (z. B. End-to-End-Verzögerung) an die Endbenutzer sowie auf die damit verbundenen Operational Expenditures (OPEX) für die Netzwerkanbieter (z. B. Stromverbrauch) haben. Daher konzentrieren wir uns in dieser Arbeit auf das Design und die Optimierung solcher Ansätze und konzentrieren uns dabei auf Kosten, End-to-End-Verzögerung und Rekonfigurierbarkeit.

Die Benutzeranforderungen kommen spontan von einem bestimmten Knoten im Netzwerk an und müssen über das Netzwerk geleitet werden, um den erforderlichen Dienstendpunkt zu erreichen. Diese Dienste werden in der Regel virtuell (z. B. als Virtual Machines (VMs)) von geografisch verteilten Data Centers (DCs) bereitgestellt. Einige Dienste erfordern sogar komplexere Setups, z. B. Service Function Chaining (SFC), bei denen der Datenverkehr eine Sequenz von Virtual Network Functions (VNFs) durchlaufen soll. Um QoS-garantierte und kosteneffiziente Dienste anzubieten, scheinen die Bestimmung des Standorts von VNFs sowie Routing-Entscheidungen ein herausforderndes Problem zu sein. Daher steht das Entwerfen, Modellieren und Optimieren solcher Ressourcenallokationsstrategien im Mittelpunkt dieser Arbeit. In dieser Arbeit stellen wir einen kosteneffizienten und

QoS-fähigen Algorithmus mit geringer Rechenkomplexität vor, der die Online-Ressourcenzuweisung für SFC-Anforderungen durchführen kann. Unsere umfassenden Evaluierungsergebnisse bestätigen die Leistungsfähigkeit der vorgeschlagenen Mechanismen in einem realistischen Online-Netzwerk-Setup mit heterogenen Infrastruktur- und Nutzeranforderungen.

Kommunikationsnetze werden von Tag zu Tag dynamischer. Angesichts der unterschiedlichen Netzwerkendpunkte (z. B. Menschen, Maschinen) und der exponentiell wachsenden Vielfalt und Nutzung von Diensten und Anwendungen zeigen die Netzwerke ein dynamischeres Verhalten. Diese Dynamik kann unterschiedliche Mittel haben und den Netzbetreibern herausfordernde Probleme bereiten. Eine Form des dynamischen Verhaltens von Netzwerken kann auf unerwartete Änderungen des Datenverkehrs zurückzuführen sein. Infolgedessen können die Infrastrukturressourcen überlastet werden, was zu unvermeidlichen Leistungseinbußen für die Benutzer und schließlich zu sinkenden Gewinnen für Netzwerkanbieter führt. Entscheidungen zur Neukonfiguration des Netzwerks, wie z. B. die Bereitstellung des neuen Lightpaths (LPs) im Netzwerk, können helfen, den Anstieg des Datenverkehrs zu bewältigen. Eine weitere Quelle der Dynamik kann die Mobilität der Nutzer sein. In Szenarien wie Mobile Edge Computing (MEC) werden die Dienste in der Nähe der Benutzer bereitgestellt. Daher kann die Bewegung des Benutzers einen unbestreitbaren Einfluss auf die empfangenen QoS haben. In der Praxis, wenn sich ein Benutzer weiter vom derzeit verwendeten MEC-Server entfernt, können die Routingkosten steigen, was zu mehr Verzögerung führt. Eine Netzwerkneukonfiguration kann ausgelöst werden, z. B. die Migration des Dienstendpunkts zu einem näheren MEC-Server in Richtung der Bewegung des Benutzers. Diese Aktion kann dazu führen, dass die QoS des Benutzers verbessert wird. In diesen Fällen besteht die Verantwortung des Ressourcenzuweisungsframeworks darin, Entscheidungen zur Netzwerkneukonfiguration auszulösen und durchzuführen, wobei deren Kosten und Merkmale im Konto berücksichtigt werden. Zusätzlich zu den Ansätzen der Ressourcenzuweisung zielt diese Dissertation darauf ab, Netzwerk-Rekonfigurationsalgorithmen in Situationen mit unerwarteten Verkehrsänderungen und Benutzermobilität zu entwerfen und zu optimieren. Bestehende Algorithmen werden untersucht und gegen die neu vorgeschlagenen Algorithmen bewertet. Die Auswertungen in verschiedenen Szenarien und Einstellungen zeigen Verbesserungen in der Laufzeit, Optimalität und Vollständigkeit im Vergleich zu den State-of-the-Art-Algorithmen.

# Contents

# Chapter 1

# Introduction

As predicted by Moore's Law, the number of transistors in a dense integrated circuit doubles every two years. Today, the number of online electronic devices is growing exponentially, reaching up to 50 Billion in 2050 [222]. In addition, with the recent advances in communication networks such as the revolution of the Internet, and the fifth/sixth-generation (5G/6G) cellular systems, millions of users are employing online services in their everyday life. These services, such as high-definition video streaming, cloud desktops, online gaming, require high computing capabilities and network bandwidth. To make these services accessible to thousands of users, they are hosted by distributed Data Centers (DCs) e.g., over a country, continent. Moreover, these services require different Quality of Service (QoS) parameters, such as end-to-end delay and data rate guarantees. Traditionally, communication networks have been planned and operated manually by human personnel, leading to a loss of resources and efficiency. However, with the appearance of promising technologies such as Software-Defined Networking (SDN), Network Function Virtualization (NFV), and Network Virtualization (NV), the task of network management is revolutionized. In particular, these technologies can assist in the automatic planning, management, and adaptation of the network resources considering today's networks' dynamic and heterogeneous characteristics.

Therefore, an undeniably essential yet challenging task for network providers is to allocate resources to their infrastructure. Network resource allocation can be performed on different parts of the network, e.g., allocation and scheduling of computational (servers, Virtual Machines (VMs)) and network (routing and bandwidth) resources to users and services. Adequately performing this task can be challenging, however crucial since it can have many impacts on the network providers' efficiency. On the one hand, a resource allocation consequence that can affect the users is the delivered QoS to the end-user, e.g., in terms of end-to-end delay. If a service endpoint (i.e., VM) is located on an over-utilized server, the processing time of the service can heavily increase, leading to a higher response time for the user. Similarly, if the VM is hosted at a DC which is very far from the user, the high routing delay can also remarkably reduce the delivered QoS to the user. On the other hand, resource allocation schemes impact also Capital Expenditures (CAPEX) e.g., infrastructure costs, and Operational Expenditures (OPEX), e.g., power consumption. For instance, a naive resource allocation scheme can use several powered-on devices while keeping them under-utilized, which can lead to a significant increase in the power consumption [38]. This is even more important in countries

with sustainability laws, such as CO2 emission taxes [120]. Therefore, designing, implementing, and evaluating such resource allocation techniques seems crucial.

The network resources and configurations need to be managed and adapted during the network runtime. In more detail, the state of the network should be carefully monitored, and when necessary, the resource allocation decisions should be reconfigured[1] accordingly. Further, if possible, the situation of the future must be foreseen and taken into account. Many situations can lead to network reconfiguration triggers, such as infrastructure breakdowns, human faults, un-/expected changes in the network traffic, and/or changing the user's location (i.e., user mobility). Specifically, in this thesis, we focus on these two cases:

**Traffic Changes** As the first case, upon certain un-/expected changes in the network (e.g., increase of user traffic, or a network failure), the affected parts of the infrastructure (e.g., network links) should be found, and proper reconfiguration actions should be performed. For instance, in a flex-grid optical network, in the event that a tenant's traffic increases, the affected Lightpath (LP)(s) should be recognized in the network. After that, based on different considerations such as available spectrum, the impact on the other users, a new LP can be launched to support the increased traffic. These decisions can be taken on the network controller, and then a new configuration is forwarded to the network devices

**Mobility of Users** With the overwhelming increase of the Internet penetration in our lives, it is now accessible anytime, every city corner, in pockets, and even in transportation systems like trains and airplanes. Further, in promising modern scenarios such as Mobile Edge Computing (MEC), the services are deployed nearby of the users to improve the delivered QoS. These advancements have brought the need for network reconfigurations based on the location of the users. For instance, if a road-assistant service belonging to a vehicle is hosted by a MEC server, with the movement of the vehicle, the location of the service might not remain optimal during the driving time. Therefore, the decision on when and where the service should be migrated is now a valid and challenging question to address by network operators.

Nevertheless, these reconfigurations do not come at no cost. Each type of reconfiguration has a cost, e.g., in terms of delay, and resource overheads [216], which needs to be accounted for. Hence, considering the network-wide parameters such as cost and/or QoS in different scenarios, the resource allocation, and management systems must offer efficient decision makings regarding the network reconfigurations during the runtime of the system. Furthermore, in some cases, these decisions can be calculated and pre-planned. However, in real-time scenarios, these decisions should be made in real-time. In other words, calculating and determining a set of reconfiguration actions should be taken fast by the network provider. Otherwise, it can decrease network performance and user satisfaction and eventually decrease profit.

[1]The terms reconfigure/reconfiguration are used interchangeably with adapt/adaptation.

## 1.1 Research Challenges

In this part, we summarize the main research challenges targeted in Chapters 2 to 5 of this dissertation. Modeling, optimizing, and analyzing the network resource allocation algorithms and mechanisms comes with various challenges, especially by considering the heterogeneous deployment of current networks, strict and dynamically changing requirements of services, and mobility of users. The main goal of this section is to advance the state-of-the-art approaches towards designing and optimizing fast and efficient network resource allocation and reconfiguration algorithms.

**Fast and Efficient Network Resource Allocation.**
The first objective of the thesis is to study the network resource allocation problem in a heterogeneous network with user requests arriving in an online manner. By employing the SDN and NFV concepts, it is important to be able to run these networks in a cost-efficient manner. Therefore, the first question is how to deploy the user requests on the physical network to minimize the network provider costs. Also, the resources of the network such as servers, network links are limited. Thus, the designed approach should be able to take these limitations into account. Further, one of the most important metrics to guarantee is the end-to-end delay of the users. However, it does not seem easy to perform the placement of service endpoints jointly with the delay-guaranteed routing decisions. The reason is, the relation of the placement and the routing decisions of the services does not seem straightforward. Last but not least, the designed solution needs to solve the problem fast. Since the user requests arrive online with no prior knowledge, the algorithm should allocate the resources to the user as soon as possible. Considering the above goals and constraints, designing a cost-efficient, delay-guaranteed, and fast algorithm seems a challenging problem to overcome.

**Necessary Network Reconfigurations Due to Traffic Changes.**
The volume of the network traffic can be changed over time. Ignoring this can cause the network to be overloaded, which can lead to significant packet drops and significant degradation of QoS. The user traffic is routed from source to destination and carry a specific amount of traffic in a network that works with flex-grid optical networks. Enabled by the novel Bandwidth Variable Transponder (BVT) technology, each node of the network can set up LPs with different configurations, defined by data rate, modulation format, etc. The initial challenge is to determine for each user request how many LPs, with which configuration should be launched, and on which path to support the requested traffic. Having that in hand, in a multi-period setting, the traffic of a specific user request increases over time. If the currently deployed LPs cannot support the increased traffic, one or more reconfiguration decisions must be made. We need to decide about the reconfiguration action(s) and their order. A challenge is that the state of the network during and after reconfiguration must not violate the constraints, such as spectrum capacity and Optical Signal to Noise Ratio (OSNR). Further, it is crucial that the network operates in a resource-efficient manner. Thus, certain objectives such as minimizing the over-provisioning and/or power consumption should be considered.

**Necessary Network Reconfigurations Due to Mobility of Users.**
Apart from the traffic changes in the network, in some scenarios, users' location might change over time. This mobility can have an impact on the delivered QoS to the end-users. We consider a multi-period scenario where a set of mobile users requests to access some online services. These services are deployed on VMs instances, hosted by distributed and capacitated DCs. Also, each of these VMs

can process only a limited amount of user traffic. The user requests are arriving in the system in an online manner. For each user request, the first challenge is to determine which VM to connect the user to. If the specific VM for the corresponding service does not exist or do not have enough available resources, a DC should be selected to launch a VM. The next challenge is to find a bandwidth- and delay-constrained routing path from the current location of the user to the selected DC. However, in future timeslots, the distance from the current location of the user to the selected DC might increase. It can increase the end-to-end delay, which is not suitable for the user (and hence, the network provider). In this situation, a VM migration (reconfiguration) can help to move the service instance closer to the current location of the user. Thus, determining when and from which to which DC to trigger a VM migration is yet another challenging question. Each of these decisions come with a cost, i.e., VM deployment (per-VM), routing (per-link), and reconfiguration costs. Thus, jointly answering the above questions in a cost-efficient manner forms a crucial yet difficult problem to tackle. In a similar scenario, another way to formulate the problem is to minimize the total delay (routing and reconfiguration) for all the users over time. In this case, at each timeslot, it is necessary to determine the user-to-DC assignment, routing, and reconfiguration(s).

## 1.2 Contributions

This section summarizes the contributions of the dissertation in the research area of network resource allocation and reconfiguration. This thesis mainly covers the research in three areas: *i)* contribution to the power- and QoS-aware Service Function Chaining (SFC) resource allocation, *ii)* cost optimization of delay-guaranteed static and mobility-aware service placement and routing, and *iii)* modeling and optimization of the end-to-end delay of mobility-aware service assignment and routing.

The **first** major contribution of the dissertation is the formulation of the network resource allocation problem in a scenario in which the user requests are arriving online with a specific type of SFC, bandwidth, and end-to-end delay requirements. The network consists of nodes and links. At each node, a Physical Machine (PM) node is located, that can host a limited number Virtual Network Functions (VNFs). A sequence of VNFs forms an SFC. The network nodes themselves are responsible for forwarding the traffic between the source and destination of the request while passing through an ordered set of VNFs. The objective is the placement of VNFs on appropriate PMs, finding the delay- and bandwidth-constrained routing from source and destination while minimizing the cost of the network provider. Accounting for cost, we focus on the power consumption of the network switches and PMs. Since the current network switches and PMs are not power-proportional, to minimize the total power consumption of the network provider, the goal is to minimize the number of powered-on nodes and to increase the utilization of them. We firstly formulate this problem as an Integer Linear Program (ILP) optimization to derive the optimal solution. Since the problem is NP-Hard, the ILP solution does not work for large problem instances. Therefore, we present *Holu*, a fast and efficient heuristic framework to tackle the problem mentioned above. For a specific user request, *Holu* decouples the placement and routing sub-problems and solves them sequentially. In the placement problem, we employ a ranking system based on the centrality of the nodes in the network to prioritize the hosting PMs for the requested VNFs. After that, we find a delay- and bandwidth-constrained

routing path between the source and destination nodes while passing through the selected PMs from the placement sub-problem. This ranking mechanism works in a way that in the long-term, the VNFs are more accessible from all the other network nodes, which can improve the success rate of the second sub-problem. Therefore, it is expected that more user requests are admitted to the network, which can increase the profits of the network provider. Evaluation results show that *Holu* performs near-optimal while outperforming the state-of-the-art works in terms of power consumption and the acceptance rate. *Holu* does not the dynamicity of the user requests, neither the traffic increase nor the location of users. In the rest of the thesis, we focus on the operational phase of the network lifetime, considering the two aforementioned dynamic concepts, which makes the subsequent three contributions.

The **second** significant contribution of the thesis focuses on a scenario wherein a network, the location of users is still static. However, the traffic volume can change over time. Currently, most front and long-haul networks are working with optical fibers, especially with flex-grid technology. Thus, we consider a flex-grid optical network, where each node is equipped with a software-tunable transponder (i.e., BVT). For initial embedding of the user request, we use a LP configuration selection mechanism and perform the spectrum allocation with the best-fit approach on the k-shortest paths. However, when the traffic request of a user request increases, our proposed algorithm determines the proper reconfigurations and their order to cope with the increased traffic. To achieve this goal, it considers three reconfiguration methods: *i)* upgrading one or more number of currently deployed LPs, *ii)* adding one or more new LPs, and *iii)* rerouting neighboring LPs to free up enough spectrum for upgrading the LP of the user request in hand. These three reconfiguration methods are designed to reduce the blocked demands, while meeting particular network constraints such as spectrum capacity and OSNR. To make the algorithm cost-efficient, we optimize the proposed methods by considering two cost functions, resource over-provisioning and power consumption. We formulate different versions of the algorithm with various objective functions. To determine the most performant approach, we evaluate our algorithm with its different objective functions and compare it with state-of-the-art algorithms. We show that our algorithm can meet the increased traffic in a multi-period scenario in a power-efficient manner.

The **third** contribution of the dissertation addresses the cost-minimized resource allocation and reconfiguration with mobile users. In this scenario, we formulate the joint placement, assignment, routing, and reconfiguration decisions over a multi-period system. We firstly formulate this optimization problem as a static case, where the optimal solution is determined for each timeslot. In this case, it does not consider the previous and future positions of the users in account. In the next step, we formulate the mobility-aware (i.e., multi-period) optimization model for this problem. This model takes the future position of the user into account to determine the optimal VM deployments, routing, and migration decisions. Nevertheless, these optimization formulations work offline, while the nature of the problem in our hand is online (since the users are arriving during time). However, these optimization formulations are still relevant to evaluate the proposed heuristic algorithms. To efficiently solve the problem online, we propose two heuristic algorithms, solving the problem per timeslot and per user. Extensive simulations show that the heuristic algorithms outperform the baseline approaches while staying close to the optimal. We also develop a demonstration tool that shows the problem and solution in different settings, using a Graphical User Interface (GUI).

The **fourth** contribution is built on top of the previous one, with some key differences. Firstly, the VM placement decision is relaxed here; hence, it is assumed that the service is already deployed on all DCs. In this case, the number of VMs for each service, the up and downscaling decisions can be delegated to the cloud auto-scaling framework, which is available on modern cloud management platforms, such as Kubernetes [56]. Secondly, we focus on a different objective function: minimizing the end-to-end delay for all the users over the time horizon. Therefore, we formulate an optimization problem to determine the assignment of users to DCs, their bandwidth-constrained routing path, and the necessary reconfigurations over time. To solve the problem in an online manner, we propose two approaches. The first one is called *Homa*, which models the problem as a dynamic bipartite matching with special properties. To solve this matching problem, we transform each problem instance to a shortest-path problem, which in exceptional cases, it can even lead to optimal solutions within seconds. The evaluations show that the proposed approach can stay near-optimal, outperform state-of-the-art algorithms, and solve the problem in a time-efficient manner. Further, we show that *Homa* can balance the decisions that it is making towards achieving a *better* solution in the long-term. The second approach is called *Figo*, which is a method based on Machine Learning (ML), which models the problem with Deep Reinforcement Learning (DRL). Moreover, considering the future airplane positions, it overlooks the short-term delay improvements for the long-term ones. After a training period, the evaluations show that *Figo* exhibits near-optimal results in different settings.

## 1.3 Author's Relevant Publications

**A. Varasteh**, S. Akhoondian Amiri, and C. Mas-Machuca. "Homa: Online In-Flight Service Provisioning with Dynamic Bipartite Matching." In: IEEE Transactions on Network and Service Management (TNSM). 2022.

**A. Varasteh**, B. Madiwalar, A. Van Bemten, W. Kellerer, and C. Mas-Machuca. "Holu: Power-Aware and Delay-Constrained VNF Placement and Chaining." In: IEEE Transactions on Network and Service Management (TNSM) (2021).

**A. Varasteh**, M. De Andrade, C. Mas-Machuca, L. Wosinska, and W. Kellerer. "Power-aware virtual network function placement and routing using an abstraction technique." In: IEEE Global Communications Conference (GLOBECOM). IEEE. 2018.

**A. Varasteh**, S. Hofmann, N. Deric, M. He, D. Schupke, W. Kellerer, and C. Mas-Machuca. "Mobility-aware joint service placement and routing in space-air-ground integrated networks." In: IEEE International Conference on Communications (ICC). 2019.

**A. Varasteh**, H. Soares Frutuoso, M. He, W. Kellerer, and C. Mas-Machuca. "Figo:Mobility-Aware In-Flight Service Assignment and Reconfiguration with Deep Q-Learning." In: IEEE Global Communications Conference (GLOBECOM). 2020.

**A. Varasteh**, S. K. Patri, A. Autenrieth, and C. Mas-Machuca. "Evaluation of Lightpath Deployment Strategies in Flexible-Grid Optical Networks." In: Optical Fiber Communication Conference (OFC). 2021.

**A. Varasteh**, S. K. Patri, A. Autenrieth, and C. Mas-Machuca. "Towards Dynamic Network Reconfigurations for Flexible Optical Network Planning." In: 25th International Conference on Optical Network Design and Modelling (ONDM). 2021.

**A. Varasteh**, W. Kellerer, and C. Mas-Machuca. "Network in the Air." In: Proceedings of the 15th International Conference on emerging Networking EXperiments and Technologies (CoNEXT). 2019.

**A. Varasteh**, S. Hofmann, N. Deric, A. Blenk, D. Schupke, W. Kellerer, and C. Mas-Machuca. "Toward optimal mobility-aware VM placement and routing in space-air-ground integrated networks." In: IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). 2019.

## 1.4 Thesis Outline

The outline of this thesis is illustrated in Fig. 1.1, presenting the structure and mapping the main contributions of the dissertation to the corresponding Chapters.

Chapter 1 presents an introduction to the dissertation topic, the research questions, the challenges, and the key contributions in the area of network resource allocation and reconfiguration.

Chapter 2 addresses the online resource allocation problem for heterogeneous user requests. The considered scenario is aiming for deploying a sequence of VNFs (so-called SFC) in a Wide Area Network (WAN) environment, and determine a delay-constrained routing path between the source, SFC, and the destination nodes. We consider the bandwidth and end-to-end delay requirements of heterogeneous services. Further, we minimize total computation and network power consumptions. In addition to the optimal optimization formulation, a fast and efficient heuristic algorithm is presented and evaluated in different settings.

Chapter 3 focuses on the problem of network reconfiguration, where the traffic is increased in flex-grid optical network. A heuristic algorithm determines the necessary reconfiguration actions to cope with the increased traffic, considering different network reconfiguration methods at each timeslot.

Chapters 4 and 5 focus on addressing network resource allocation and reconfiguration optimization in a multi-period scenario with mobile users. In more detail, they present modeling and optimization of static and mobility-aware resource allocation and reconfiguration algorithms to minimize total operation costs (i.e., in Chapter 4), and a best-effort delay minimization approach (i.e., in Chapter 5) to solve the problem in an online manner. We propose different fast and efficient heuristic algorithms to cope with the low-complexity requirement of the solutions. Finally, Chapter 6 concludes this thesis by summarizing the main results and giving a brief outlook for interesting future work and research directions.

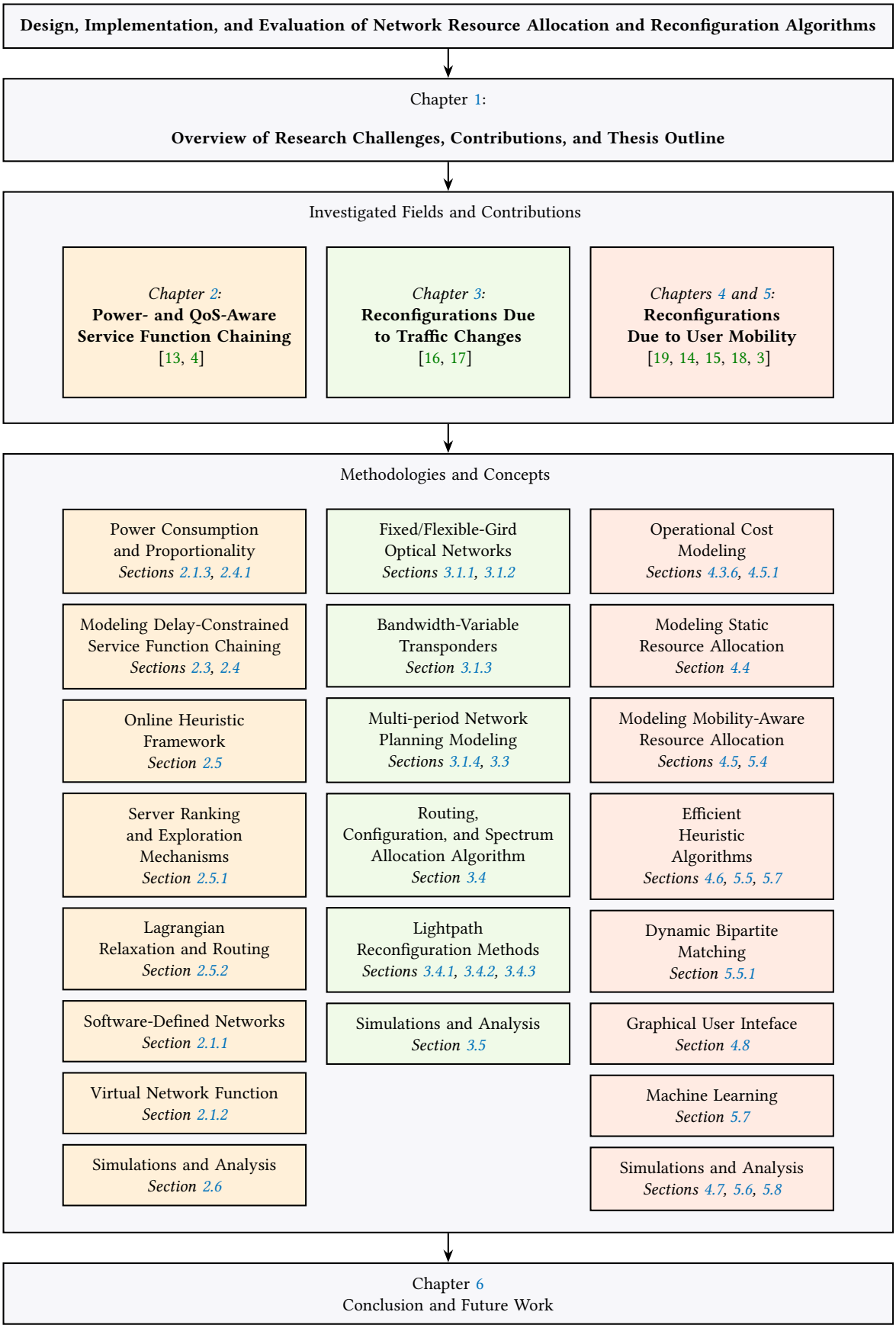



**Figure 1.1:** Outline of the thesis. The mapping of the main contributions to the corresponding thesis Chapters.

# Chapter 2

# Power-Aware and Delay-Guaranteed Service Function Chaining

## 2.1 Introduction

In this chapter, we present a study towards power-efficient and delay-guaranteed resource management in an Software-Defined Networking (SDN) and Network Function Virtualization (NFV)-enabled metro/core network. In this scenario, the users required services are deployed on physical resources using virtualization technologies. However, these resources are usually under-utilized and/or over-provisioned, resulting in power-inefficient deployments. To improve the power efficiency, the minimum number of Physical Machines (PMs) and network equipment should be utilized to deploy the services, which are not concomitant. Considering the existing PM and switch power consumption models and their resource constraints, we formulate the power-aware and Quality of Service (QoS)-guaranteed joint Virtual Network Function (VNF) placement and routing problem as an Integer Linear Program (ILP). Due to the NP-completeness of the problem, we propose *Holu*, a fast heuristic framework that efficiently solves this problem in an online manner. Our simulation results indicate that *Holu* outperforms the state-of-the-art algorithms in terms of total power consumption and acceptance rate. This chapter is based on the presented work in [13], and [4].

### 2.1.1 Software-Defined Networking (SDN)

In legacy networks, the Control Plane (CP) and Data Plane (DP) co-exist on the same forwarding device (e.g., network switch). In contrast, in SDN, these planes are decoupled [160]. In particular, the control plane logic of the network (e.g., path calculation) is implemented in a logically-centralized software controller, i.e., SDN controllers. The SDN controller programs the data plane (i.e., forwarding devices) to forward the packets in a specific manner. The comparison of the legacy and SDN networks is presented in Fig. 2.1.

SDN controllers are a piece of software that can be deployed in commodity servers, typically as a Virtual Machine (VM) or container. The controllers are written in different languages, e.g., Ryu in Python [70], and Open Daylight in Java [69]. The forwarding devices handle the incoming packets according to their forwarding table. This table is configured by the SDN controller, using protocols like OpenFlow [160], based on a *match-action* logic. In practice, the incoming packets are matched

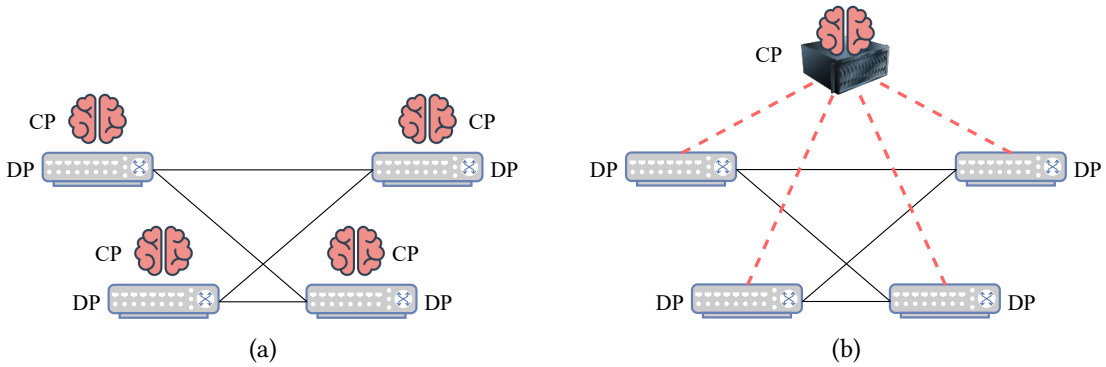



**Figure 2.1:** Comparison of the legacy and SDN networks. (a) Shows the co-existence of CP and DP at the same device. (b) Represents the SDN architecture where the CP is decoupled from DP and stored on an SDN controller.

with a rule based on the header data, e.g., destination IP address and TCP port number. After that, the packet is treated with the corresponding *action*, e.g., forwarding to a specific port number. SDN uses this programmability along with a logically centralized view of the network to bring a more flexible, manageable, and efficient network control. For instance, the traffic can be deployed at runtime based on the existing flows in the network to have potential cost savings through an optimized network operation and/or to improve the QoS.

### 2.1.2 Network Function Virtualization (NFV) and Service Function Chaining (SFC)

In legacy networks, network functions such as load balancer, proxies, and Intrusion Detection System (IDS) have been deployed in hardware middleboxes. In [190], it is stated that the number of middleboxes in an enterprise network is comparable to the number of routers. These middleboxes are usually deployed on vendor-specific and expensive hardware that is configured and maintained using trained human personel [36]. Therefore, deploying and maintaining these middleboxes come with a high Capital Expenditures (CAPEX) and Operational Expenditures (OPEX) for the providers. In addition, it is often impossible to add/remove functionalities to/from these middleboxes, which can make the flexibility of the networks very costly.

NFV [67], as an emerging and promising technology, can overcome these limitations. In NFV technology, the packet processing is moved from the hardware middleboxes to software instances. This software can be implemented in VMs or container technologies and be deployed on Commercial Off-The-Shelf (COTS) equipment. In this technology, these software middleboxes are referred to as VNFs. In addition, to provide secure and performant services, network providers often require their traffic to pass through a specific sequence of VNFs, referred to as an SFC [115, 116].

The NFV architecture consists of several interconnected nodes which consist of network switches and PMs. The former forwards the traffic through the network, whereas the latter host VNFs. Since PMs can host several VNFs, it enables the physical consolidation of networks. Upon receiving user requests with a specific SFC to traverse, network providers have to find the best PMs to configure the required VNFs while taking the NFV architecture and its available resources into account. Fig. 2.2 shows this architecture with an exemplary deployed SFCs. In this figure, two demands are consid-

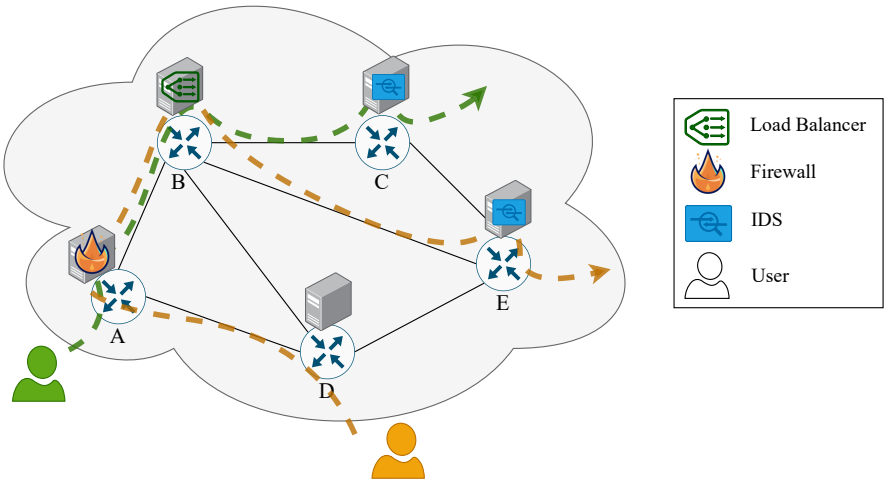



**Figure 2.2:** An example of two deployed user demands with the same required SFCs in a network.

ered: *orange* and *green*. Each of them is characterized by source and destination nodes, data rate, and a specific SFC. In this example, the source and destination nodes of the *orange*/*green* request are A/D and C/E, respectively. Both demands ask for the same SFC: {Firewall→Load Balancer→IDS} with a specific data rate requirement. The deployment of VNFs belonging to the requested SFC on PMs in the network is demonstrated, along with the setup of routing paths to provide users with the required SFCs. Thus, considering the heterogeneous and complex network and services of today, it is a challenging problem to determine the required number and placement of VNFs and the routing paths that optimize network operational costs and utilization. Further, guaranteeing the tight QoS parameters such as end-to-end delay makes the SFC deployment even more complicated.

### 2.1.3 Power Consumption and Proportionality

Despite its importance, the power-efficiency of the NFV-enabled networks has been only slightly considered. Today, around 7-12% of the total electrical power consumption is required for Internet technologies [23, 72, 129]. Network providers are challenged to increase their connectivity and services while being sustainable. Bolla *et. al.* [47] have shown the aggressive increase of power consumption in the networks operated by the major Telecom operators worldwide (e.g., AT&T, Verizon). Moreover, the global annual Internet traffic is expected to reach 4.2ZB, in 2022 (with an increase of almost 400% with respect to 2017 [129]). As a result, more and more operators work on the reduction of their carbon footprint and greenhouse gas emission by aiming at decreasing their power consumption [205, 77].

In order to reduce the power consumption, analysis and modeling of the power consumed by the main NFV components need to be done, particularly PMs and network equipment. According to Fig. 2.3, three power states can be considered for PMs and switches:

- Standby: the server is in low-power (sleep) mode and consumes a negligible amount of power.
- Idle: the device is powered on; however, its utilization is almost 0% (no traffic load).

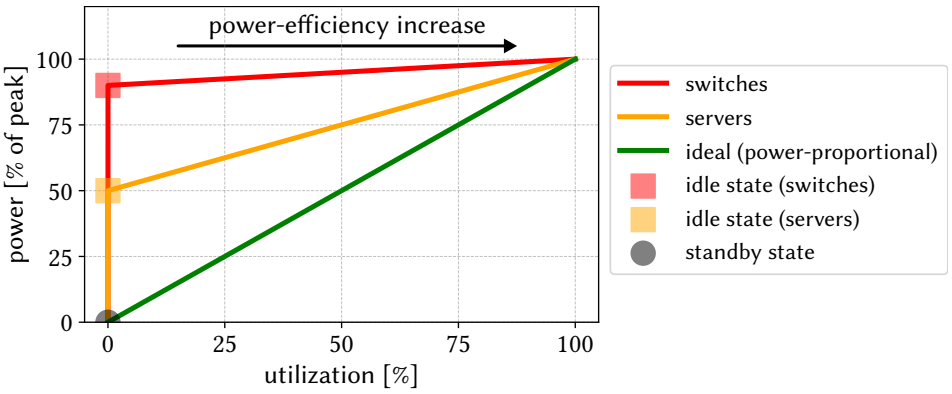



**Figure 2.3:** Power consumption vs. utilization (power-proportionality) comparison for online PMs and network switches with respect to the theoretical ideal power-proportional case. It is shown that both PMs and switches are not power-proportional, i.e., they consume a significant amount of power when they are in the idle state, while in the standby state, they consume a negligible amount of power. Therefore, to increase the power-efficiency, the number of online components should be reduced (increasing the resource utilization).

- Online: the device is powered on, and its utilization is higher than 0% (processing the traffic load).

We start with the PM power consumption. It has been shown that PMs can consume almost 50% of their maximum power when they are in the idle state [101, 131, 158]. Also, they consume a negligible amount of power when are in standby mode [198] (also referred to as the offline state in some works). Similar to PMs, network resources are usually over-provisioned to support the maximum traffic. However, their utilization rarely reaches the peak network capacity [118, 154]. It has been observed that the online network components such as switch chips and fans consume a significant amount of power even with low workload [118, 156, 219]. Consequently, idle networking devices are not power-efficient since an idle network switch can consume up to 90% of the peak power consumption [154].

Accordingly, the relation between power consumption and the device utilization is depicted in Fig. 2.3 for PMs and switches, assuming a linear power profile [59, 158]. It can be observed that PMs and network devices are not power-proportional, i.e., they do not consume power proportional to their utilization, which differs significantly from the ideal power proportionality case [158, 21, 131]. This difference causes a waste of power consumed by under-utilized devices. Thus, minimizing the number of online PMs and switches can improve the power-proportionality (see Fig. 2.3) and hence, improve the power-efficiency of the service provider.

### 2.1.4 Key Contributions

In this chapter, we study and address the Power-aware and QoS-guaranteed joint VNF Placement and Routing (PQ-VPR) problem. Considering the capacitated network resources, the main goal is to minimize the number of online PMs and network switches required to allocate the requested SFCs, while guaranteeing the end-to-end delay (i.e., sum of propagation and VNF processing delay) requirements. We first formulate this problem as an ILP based on our previous work [13]. The problem formulation has been improved by also considering the VNF processing time, which dynamically changes

depending on the traffic. Considering the NP-completeness of the problem, the ILP is not usable for solving real-world problem sizes. Therefore, we propose *Holu*, an efficient heuristic that solves the PQ-VPR problem in an online manner. In more detail, *Holu* decomposes the PQ-VPR problem into two sub-problems which are solved in sequence: *i)* VNF placement, *ii)* routing. In the first sub-problem, we rank the PMs in the network according to their centrality and the requested VNF types in the SFC. Thereafter, we employ a Delay-Constrained Least-Cost (DCLC) shortest-path algorithm to find the path between the selected VNFs in the previous step using a routing heuristic based on Lagrange Relaxation based Aggregated Cost (LARAC) [207]. As an essential feature, our routing algorithm is able to efficiently split the end-to-end delay budget between subsequent VNFs in an SFC. It can significantly increase the acceptance rate of requests, even with stringent end-to-end delay requirements.

Therefore, the main contributions of this solution can be summarized as:

- Presenting the PQ-VPR problem as an ILP optimization model to *i)* determine the optimal number of the VNFs and their mapping to the PMs, *ii)* allocate user requests to the VNF instances, *iii)* find a path to route the traffic through the allocated VNFs to form the SFC, *iv)* meet the resource capacities and end-to-end delay constraints including the link propagation and traffic-aware VNF processing delays, and *v)* minimize the total power consumption.
- Proposing *Holu*, an online heuristic framework to tackle the PQ-VPR problem by dividing it into two sub-problems: VNF placement and routing.
- Implementation and performance evaluation of the proposed heuristic and comparing with the state-of-the-art Cost-efficient Proactive VNF placement and chaining (CPVNF) and Betweenness Centrality Shortest-Path (BCSP) algorithms [80, 13] in terms of total power consumption, acceptance ratio, runtime, etc.

### 2.1.5 Organization

The rest of this chapter is organized as follows: The related work is reviewed in Subsection 2.2. Then, we present the system model and problem definition in Section 2.3 followed by the offline optimization formulation in Section 2.4. Thereafter, in Section 2.5, we introduce the *Holu* framework. Finally, we present the performance evaluation of the framework in Section 2.6, and summarize the chapter in Section 2.7.

## 2.2 Related Work

Although power-efficient VM placement is a well-studied field in the cloud computing environment [176, 211], VM placement and VNF placement in NFV/SFC paradigm problems differ in many ways. The former case is a problem that focuses on placing/packing a set of VMs on different PMs while the latter considers a specific ordered set of VNFs. In addition, the solution should contain routing and path allocation through these ordered VNFs, which makes it a fundamentally difficult problem to solve [184]. Therefore, the VM placement can be considered as a special case of the latter problem. There has been a large body of papers that have investigated the VNF placement and

**Table 2.1:** Comparison of state-of-the-art and the proposed solution, *Holu*.

| References | Decisions | | Power | | Features | | |
|---|---|---|---|---|---|---|---|
| | VNF Placement | Routing (chaining) | Physical Machines | Network | Delay-Constrained | Shared VNF Instance | Online |
| [207] | ✓ | ✓ | ✗ | ✗ | ✓ | ✓ | ✓ |
| [13] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ |
| [80] | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✗ |
| [175] | ✓ | ✗ | ✓ | ✗ | ✓ | ✗ | ✗ |
| [225] | ✓ | ✗ | ✓ | ✓ | ✗ | ✓ | ✗ |
| [96] | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | ✗ |
| [128] | ✓ | ✓ | ✗ | ✓ | ✗ | ✓ | ✗ |
| [85] | ✗ | ✓ | ✓ | ✗ | ✓ | ✓ | ✗ |
| [87] | ✓ | ✗ | ✓ | ✗ | ✗ | ✓ | ✓ |
| [198] | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✓ |
| [197] | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✓ |
| [136] | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✗ |
| [130] | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ |
| [104] | ✓ | ✓ | ✗ | ✗ | ✗ | ✓ | ✗ |
| [84][186] | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ |
| [88][193] | ✓ | ✓ | ✓ | ✗ | ✗ | ✓ | ✗ |
| [36][144] | ✓ | ✓ | ✓ | ✗ | ✓ | ✗ | ✗ |
| *Holu* | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

routing problem with different objectives, such as minimizing total deployment cost [68, 187, 203, 201], minimizing total end-to-end delay [108, 231, 75], minimizing network resources [112, 111] and routing costs [84, 186], maximizing reliability [138, 147, 179, 126].

Let us summarize the most recent and relevant studies addressing the power-aware VNF placement and routing problem [175, 225, 193, 96, 128, 85, 87, 198, 197, 36, 136, 144, 130, 88]. These works have been grouped into three categories: *i)* VNF placement: place a set of VNFs in order to meet an objective, *ii)* SFC routing: these papers assume the VNFs are already deployed in the network. Thus, they focus on finding the path for the traffic traversing through these VNFs, and *iii)* joint VNF placement and routing: in addition to VNF placement, the traffic path through these VNFs must be determined.

### 2.2.1 Virtual Network Function (VNF) Placement

In the first category, not concerned with the routing decisions, authors in [175, 225, 87] have tackled the VNF placement problem. In particular, Pham *et. al.* [175] aimed at deploying VNFs by using as fewer number of online PMs, such that the communication cost between them is optimized. They proposed a fast solution by using a sampling-based Markov approximation method combined with matching theory. Further, Yang *et. al.* [225] studied VNF chain placement in data centers. They provided an algorithm to save power in servers and network switches. A step further was taken by authors in [87], where they have proposed a dynamic server consolidation approach using VM live migration to achieve power-efficiency by maximizing the number of PMs in the standby state.

### 2.2.2 Virtual Network Function (VNF) Routing

The second category belongs to the works which tackle the challenges brought by the routing problem. There are several dimensions to consider in this category. Some references have focused on the routing problem and considered the power consumption of ternary content-addressable memory (TCAM) of network switches [135, 125, 71, 212, 64]. However, we consider the network power consumption based on the online network switches and the number of active ports and their utilization.

Assuming that the VNFs are already placed in the network, some works have focused on finding a path going through the required VNFs (i.e., SFC) considering some constraints, e.g., delay, capacity. For example, the authors in [85] formulated a problem to allocate and schedule traffic flows with deadlines to VNFs while minimizing the total PM power consumption. Recently, the graph layering technique has been proposed as an efficient way to find a path through an already placed set of VNFs [84, 186, 104, 207]. These papers transform the network into a layered graph based on the VNF of the SFC. The user traffic can be routed layer by layer from the top to the bottom layer. For instance, KARIZ, a local search heuristic proposed by [104], finds the path between two layers by solving the minimum cost flow problem. The objective of KARIZ is to minimize the network resource costs while disregarding the end-to-end delay constraint. As another work, after the graph layering transformation, authors in [84] use conventional shortest-path algorithms, e.g., Dijkstra, to calculate the path between the source and destination nodes. To reduce the computational time when using a shortest-path algorithm, Sallam *et. al.* in [186] propose a pruning algorithm to simplify the constructed layered graph. However, similar to [104] and [84], they did not consider the end-to-end delay constraint in their problem. Nevertheless, in our previous work [207], we proposed a heuristic to find a delay-constrained path, passing through a selected set of VNF nodes in a layered graph, achieving near-optimal performance.

### 2.2.3 Joint VNF Placement and Routing

Finally, as the third category, some references extend the problem to consider the routing jointly with the VNF placement decision [193, 96, 128, 198, 197, 36, 136, 144, 130, 88]. In more detail, their main goal is to place the VNFs on PMs, allocate them to the user requests, and route the traffic through these VNFs. These decisions can be constrained to capacity and/or delay requirements, optimizing PM and/or network power consumption. Specifically, the authors in [36] focused on determining the

required number and placement of VNFs to optimize the network utilization and operational costs (in terms of PMs power consumption) without violating service level agreements. They presented an efficient heuristic based on dynamic programming to solve this problem. Furthermore, Jang *et. al.* [130] formulated a multi-objective optimization model which maximizes the acceptance ratio and minimizes the power cost for multiple service chains. After transforming the model into a single-objective Mixed Integer Linear Program (MILP) problem, they proved that the problem is NP-hard and proposed an algorithm based on linear relaxation and rounding to approximate the solution of the MILP in polynomial time. In addition to optimizing the VNF placement and routing, the authors in [198] presented online algorithms to reconfigure the network based on traffic changes.

However, in these three categories, some missing considerations are addressed in this chapter. For example, in the first and third categories, the necessity of coordination between VNF placement and routing decisions is disregarded. Thus, in this section, we tackle the PQ-VPR problem. Moreover, opposed to this chapter, some approaches do not consider both PM and network power consumption into account, which can increase the OPEX of service providers. Moreover, unlike some reviewed related works, we guarantee the end-to-end delay. Also, some works [198, 87, 88] considered VNF migration and reconfiguration, which we do not focus on it in this chapter. A comparison of different state-of-the-art solutions to this thesis is summarized in Table 2.1. Further, comprehensive surveys on VNF chain placement are available for interested readers [44, 121, 145, 65]. In the remaining of this section, we present *Holu*, an online heuristic framework that presents an efficient VNF placement approach coupled with a fast delay-constrained routing algorithm to solve the PQ-VPR problem.

## 2.3 System Model and Problem Definition

Before presenting the mathematical modeling part, we note that the used notations and their definition are presented in Table 2.2.

### 2.3.1 Network Model

We represent the network as a unidirectional graph $G = (N, L)$, being $N$ the set of nodes, and $L$ the set of unidirectional links between pair of nodes. Each physical link $(i, j) \in L$ is characterized by the data rate capacity $B_{i,j}$ and the propagation delay $D_{i,j}$ (based on the distance of nodes $i$ and $j$). Every node $n \in N$ consists of a switch and a co-located PM (or a cluster of PMs). The switch can interconnect any input port with any output port as well as forward the connection to the PM when required. On the other hand, the PM is characterized by a set of resources $u \in U$, such as CPU, RAM, and storage. Also, each VNF type $k \in K$ has a resource requirement $\Delta_{k,u}$, processing delay $\varphi_k$, and processing capacity $\Phi_k$. A deployed instance of VNF $k$ can be shared among several user requests as long as its maximum processing capacity $\Phi_k$ is not surpassed. Otherwise, the new instance of VNF $k$ should be placed either on an online PM with enough available resources or on a PM which must be powered on. These three alternatives have a different impact on power consumption, as introduced in the later sections.

**Table 2.2:** Notation definition for the PQ-VPR problem.

| **Sets and Parameters** | |
|---|---|
| $G = (N, L)$ | Network graph |
| $\mathcal{R}$ | Set of requests |
| $\bar{G}_r = (\bar{N}_r, \bar{L}_r)$ | Virtual network graph of the request $r$ |
| $F$ | Set of VNF types |
| $\mathcal{V}_r^s$ | Source node of request $r$ |
| $\mathcal{V}_r^d$ | Destination node of request $r$ |
| $\mathcal{B}_r$ | Data rate of request $r$ |
| $\mathcal{D}_r$ | Maximum delay of request $r$ |
| $C_r$ | SFC of request $r$ |
| $B_{i,j}$ | Data rate capacity of physical link $(i, j)$ |
| $D_{i,j}$ | Propagation delay of physical link $(i, j)$ |
| $\varphi_k$ | Processing delay of VNF type $k$ |
| $\Phi_k$ | Processing capacity of VNF type $k$ |
| $U$ | Set of VNFs/PMs resource types |
| $\Delta_{k,u}$ | Required resource type $u \in U$ by VNF type $k$ |
| $C_{i,u}$ | Maximum capacity of resource type $u \in U$ in PM $i$ |
| $\theta_i^{CPU}$ | CPU utilization of PM $i$ |
| $P_{pm}^{idle}, P_{switch}^{idle}$ | PM and network switch idle power consumption |
| $P_{pm}^{max}$ | Maximum PM power consumption |
| $P_{port}$ | Network switch port power consumption |
| $P_{pm}^{T}, P_{net}^{T}$ | Total PM and network power consumption |
| $\Psi$ | Large positive integer |
| $\gamma_n^{r,k}$ | The ranking value of PM $n$ with respect to VNF $k \in C_r$ |
| $\alpha_n^{r,k}$ | The centrality impact of PM $n$ with respect to VNF $k \in C_r$ |
| $\beta_n^{r,k}$ | The power consumption impact of PM $n$ with respect to VNF $k \in C_r$ |
| $c_{i,j}$ | Routing cost function assigned to link $(i, j)$ |
| $\mathcal{P}_{i,j}$ | Routing power impact of using link $(i, j)$ |
| $\mathcal{Q}_j$ | Betweenness centrality of node $j$ |
| $\mathcal{S}_r$ | Set of candidate PMs for request $r$ |
| **Decision Variables** | |
| $x_i \in \{0, 1\}$ | =1, if PM $i$ is online |
| $y_i \in \{0, 1\}$ | =1, if switch $i$ is online |
| $q_{i,j} \in \{0, 1\}$ | =1, if link $(i, j)$ is online |
| $w_{h,l,r}^{i,j} \in \{0, 1\}$ | =1, if virtual link $(h, l) \in \bar{L}_r$ is mapped to $(i, j) \in L$ |
| $a_{i,k,r} \in \{0, 1\}$ | =1, if VNF $k$ for request $r$ is placed in PM $i$ |
| $n_{i,k} \in \mathbb{N}$ | Number of instances of VNF type $k$ on PM $i$ |

### 2.3.2 User Request Model

We define a user request $r \in \mathcal{R}$ as following 5-tuple:

$$r = (\mathcal{V}_r^s, \mathcal{V}_r^d, \mathcal{D}_r, \mathcal{B}_r, C_r) \tag{2.1}$$

where $\mathcal{V}_r^s$ and $\mathcal{V}_r^d$ are the source and the destination nodes, $\mathcal{D}_r$ is the maximum allowed end-to-end delay, $\mathcal{B}_r$ is the requested data rate, and $C_r = \{k_1, k_2, ..., k_{|c_r|}\}$ is the requested SFC, an ordered set of VNFs that the demand should be routed through.

### 2.3.3 Service Function Chaining (SFC) Model

An SFC can be modeled as a virtual network represented as a graph $\bar{G}_r = (\bar{N}_r, \bar{L}_r)$, where for request $r$, $\bar{N}_r$ are the set of virtual nodes which are $C_r = \{k_1, k_2, ..., k_{|c_r|}\}$; and $\bar{L}_r$ are the set of links where the virtual link $(i, i+1)$ interconnects $k_i$ with $k_{i+1}$. In particular, for each request $r$, the virtual nodes $\bar{N}_r$ (i.e., the VNFs) and their interconnecting links $\bar{L}_r$ should be mapped to the physical network $G$. Hence, the link $(i, j) \in \bar{L}_r$ between two consecutive VNFs must be assigned to a path connecting physical links in $L$. Also, the ingress and egress virtual nodes match the physical source and destination nodes in the substrate physical network. In this way, $\bar{G}_r$ has to be mapped over $G$ such that the requirements of request $r$ are met in terms of delay and capacity while the consumed total power is minimized.

### 2.3.4 Network Power Consumption Model

The network power consumption refers to the amount of power consumed for the transmission, which includes the power consumed by the switches at the network nodes as well as the active interconnecting links. In particular, when a network switch is powered on, it consumes a base power $P_{switch}^{idle}$ Watts which is independent of the traffic load [50, 155, 219, 59]. Similarly, a network port consumes $P_{port}$ Watts if the port is powered on (otherwise, 0 Watts). Therefore, the power of a fully-utilized network switch can be computed as [155, 219]:

$$P_{switch} = \begin{cases} P_{switch}^{idle} + \sum_{ports} P_{port} & \text{, if it is } \textit{online} \\ 0 & \text{, otherwise} \end{cases} \tag{2.2}$$

### 2.3.5 PM Power Consumption Model

The most power-consuming factor of a PM is the CPU [93, 62, 63, 43]. Hence, the power consumption model for the PM is based on its CPU utilization. The power consumption of PM $i$ can be calculated as below [146, 40, 41]:

$$P_{pm} = \begin{cases} P_{pm}^{idle} + (P_{pm}^{max} - P_{pm}^{idle})\ \theta_i^{CPU} & \text{, if it is } \textit{online} \\ 0 & \text{, otherwise} \end{cases} \tag{2.3}$$

where $P_{pm}^{idle}$ and $P_{pm}^{max}$ are the consumed power when the CPU utilization is 0% and 100%, respectively. Also, $\theta_i^{CPU}$ is the CPU utilization of the $i^{th}$ PM.

### 2.3.6 Problem Statement

Given a network and a set of user requests: *i)* determine the optimal number of the VNF instances to be deployed, *ii)* Mapping of these VNF instances to the PMs, *ii)* allocating user requests to the deployed VNFs, *iv)* find a path to route the user traffic through the allocated VNFs (i.e., forming the SFC), *v)* guarantee the end-to-end delay required by the user requests, which should not exceed the sum of the network propagation delay and the processing delay of VNFs, *vi)* meet the PM and network capacity constraints, and finally *vii)* minimize the total PM and network power consumption.

## 2.4 Integer Linear Program (ILP) Formulation

In this section, we mathematically formulate the PQ-VPR problem as an offline ILP optimization model.

### 2.4.1 Total Power Consumption

As mentioned before, the objective function is to minimize the total power consumption, which is defined as the sum of the network and PM consumed power. According to the network power consumption model in Eq. 2.2, the total network power consumption denoted by $P^T_{net}$ can be calculated as:

$$P^T_{net} = P^{idle}_{switch} \sum_{i \in N} y_i + 2P_{port} \sum_{(i,j) \in L} q_{i,j} \tag{2.4}$$

where $y_i \in \{0, 1\}$ is a variable indicating if switch $i$ is online. Also, $q_{i,j} \in \{0, 1\}$ is defined as a variable that is equal to 1 if the physical link between nodes $i$ and $j$ is online. Note that an active physical link $(i, j) \in L$ (connecting node $i$ to $j$) requires two online ports (one per source/destination node).

Moreover, according to the PM power consumption model Eq. 2.3, the total PM power consumption $P^T_{pm}$ can be calculated as:

$$P^T_{pm} = \sum_{i \in N} x_i \left( P^{idle}_{pm} + (P^{max}_{pm} - P^{idle}_{pm})\, \theta^{CPU}_i \right) \tag{2.5}$$

where $x_i$ is a binary variable indicating if the PM $i$ is powered on. $\theta^{CPU}_i$ is the CPU utilization of PM $i$, which is calculated as the ratio between the all required CPU resources by the hosted VNFs and the available CPU resources on PM $i$, i.e., $\theta^{CPU}_i = \sum_{k \in K} \Delta_{k,u}/C_{i,u}$ for $u = CPU$.

In addition, some constraints need to be met by the ILP model. First, the set of resources on a PM is limited, and therefore, these resources cannot be exceeded by the hosted VNFs, i.e.,:

$$\sum_{k \in K} \Delta_{k,u} n_{i,k} \leq C_{i,u}, \forall i \in N, \forall u \in U, \tag{2.6}$$

where $\Delta_{k,u}$ is the amount of resource type $u$ used by the placed VNF $k$, $n_{i,k}$ is an integer variable indicating the number of placed VNFs of type $k$ on PM $i$, and $C_{i,u}$ represents the maximum capacity of resource type $u$ in PM $i$.

Secondly, each VNF $k$ has a limited processing capacity, which is denoted by $\Phi_k$. Therefore, the sum of the rates of all requests served by function $k$ in the PM $i$ must not exceed the processing capacity of VNF $k$, i.e.,

$$\sum_{r \in \mathcal{R}} a_{i,k,r} \mathcal{B}_r \leq n_{i,k} \Phi_k, \forall i \in N, \forall k \in K, \tag{2.7}$$

where $a_{i,k,r} \in \{0,1\}$ is a variable which equals to 1 if VNF $k$ of request $r$ is assigned to the PM $i$, and $\mathcal{B}_r$ is the requested data rate of $r$.

The last set of capacity constraints belong to the physical link capacities. The sum of the data rate required by all requests served by link $(i,j)$ should not be larger than the capacity of the link $(i,j)$:

$$\sum_{r \in \mathcal{R}} \sum_{(h,l) \in \bar{L}_r} w^{i,j}_{h,l,r} \mathcal{B}_r \leq B_{i,j}, \forall i \in N, \forall j \in N, \tag{2.8}$$

where $w^{i,j}_{h,l,r} \in \{0,1\}$ is a variable that equals to 1 if the physical link $(i,j)$ is used by the virtual link $(h,l)$ of $r$.

As introduced in Section 2.3.1, the nodes of the virtual network graph $\bar{G}_r = (\bar{N}_r, \bar{L}_r)$ demanded by the request $r$ must be embedded in the physical graph $G = (N, L)$. Firstly, the source and destination nodes of $\bar{G}_r$ must be mapped to the physical source and destination nodes in the $G$. Hence, the source and destination nodes of request $r$ should be mapped to the same node on the substrate physical node:

$$a_{i,k,r} = 1, \text{ if } i = f = \mathcal{V}^s_r, \forall r \in \mathcal{R}, \tag{2.9}$$

$$a_{i,k,r} = 1, \text{ if } i = f = \mathcal{V}^d_r, \forall r \in \mathcal{R}. \tag{2.10}$$

Moreover, the VNFs in SFC $C_r$ must be mapped to a node $n \in N$ that actually hosts an instance of VNF $k, \forall k \in C_r$:

$$a_{i,k,r} \leq n_{i,k}, \forall i \in N, \forall k \in K, \forall r \in \mathcal{R}. \tag{2.11}$$

The flow conservation law is expressed in flow states such that for each network switch $i \in N$, the difference of all outgoing and incoming physical links that are used for the virtual link between virtual nodes $h$ and $l$, and request $r$ must be equal:

$$\sum_{(i,j) \in L} w^{i,j}_{h,l,r} - \sum_{(j,i) \in L} w^{j,i}_{h,l,r} = a_{i,h,r} - a_{i,l,r}, \ \forall i \in N, \ \forall k \in \bar{N}_r, \ \forall l \in \bar{N}_r, \ \forall r \in \mathcal{R}, \tag{2.12}$$

The total delay of the embedded request $r$ is modeled as the sum of the VNF processing time of each VNF $k \in C_r$, denoted by $\varphi_{k,r}$, and the propagation delay $D_{i,j}$ in the physical links. This summation must be limited to the required end-to-end delay $\mathcal{D}_r$ of each request $r \in \mathcal{R}$, thus:

$$\sum_{i \in N} \sum_{k \in K} \varphi_k a_{i,k,r} + \sum_{(i,j) \in L} \sum_{(h,l) \in \bar{L}_r} D_{i,j} w^{i,j}_{h,l,r} \leq \mathcal{D}_r, \forall r \in \mathcal{R}, \tag{2.13}$$

where $\varphi_{k,r}$ denotes the processing delay of the VNF $k$ for request $r$, which is considered proportional to the data rate of request $r$ (i.e., $\varphi_{k,r} = \mathcal{B}_r / \Phi_k$).

Finally, let us introduce three indicator variables to control the operation status (i.e., *online* or *standby*) of PMs, physical links, and network switches):

$$\sum_{r\in\mathcal{R}}\sum_{(h,l)\in\bar{L}_r} w^{i,j}_{h,l,r} \leq q_{i,j}\Psi, \forall i \in N, \forall j \in N, \tag{2.14}$$

$$\sum_{j\in N}(q_{i,j}+q_{j,i}) \leq y_i\Psi, \forall i \in N, \tag{2.15}$$

$$\sum_{k\in K} n_{i,k} \leq x_i\Psi, \forall i \in N, \tag{2.16}$$

where $\Psi$ is a large positive number. Considering the aforementioned definitions and constraints, we can formulate the ILP optimization model as below:

$$\begin{aligned}
&\text{minimize } \left(P^T_{pm}+P^T_{net}\right), \\
&\quad \text{s.t. constraints } (2.6)-(2.16), \\
&\quad \text{vars: } x_i, y_i \in \{0,1\}, \forall i \in N, \\
&\qquad q_{i,j} \in \{0,1\}, \forall (i,j) \in L, \\
&\qquad w^{i,j}_{h,l,r} \in \{0,1\}, \forall (i,j) \in L, \forall (h,l) \in \bar{L}_r, \forall r \in \mathcal{R}, \\
&\qquad a_{i,k,r} \in \{0,1\}, \forall i \in N, \forall k \in K, \forall r \in \mathcal{R}, \\
&\qquad n_{i,k} \geq 0, \forall i \in N, \forall k \in K.
\end{aligned} \tag{2.17}$$

In this formulation, we jointly map a set of virtual nodes $\bar{N}_r, \forall r \in \mathcal{R}$ to PMs together with the mapping of virtual links $\bar{L}_r, \forall r \in \mathcal{R}$ onto paths in the underlying network $G$ connecting the individual servers. Also, this embedding must meet capacity and path length (i.e., delay) constraints. This problem can be reduced to the graph embedding problem, which is generally NP-hard and inapproximable [183, 184]. Thus, the presented ILP model (2.17) does not apply to practical-sized problems as it cannot be solved in a reasonable period. Therefore, in the following sections, we propose an online heuristic algorithm to solve the aforementioned problem in polynomial time.

## 2.5 *Holu*: The Online Heuristic Framework

In this section, we propose an efficient online heuristic framework named *Holu* to solve the PQ-VPR problem with much lower and more manageable complexity. Many of the heuristics presented in the state-of-the-art divide the problem based on each VNF, they solve the resulting sub-problems sequentially per VNF [94, 213, 13, 25, 136, 80, 193, 128]. In more details, considering a user request $r$ with $C_r = \{k_1, k_2, k_3\}$, the referred state-of-the-art papers would consider four sub-problems to determine a solution to the VNF placement and routing problems, one for each VNF and last one for routing from the last VNF to the destination, without having a fallback mechanism after a decision for a VNF is made. They sequentially solve these sub-problems, i.e., determining the VNF placement and the routing first for $k_1$, then for VNF $k_2$, and finally VNF $k_3$ and they build the SFC from the $\mathcal{V}^s_r$ to $\mathcal{V}^d_r$. Considering the previous example, the algorithm tries to place $k_3$ after placing the $k_1$, $k_2$, and find a path from the placed $k_2$ to the potential $k_3$ PMs. This might not be feasible anymore, since the constraint budgets (e.g., delay) might have been already consumed by the previous VNFs

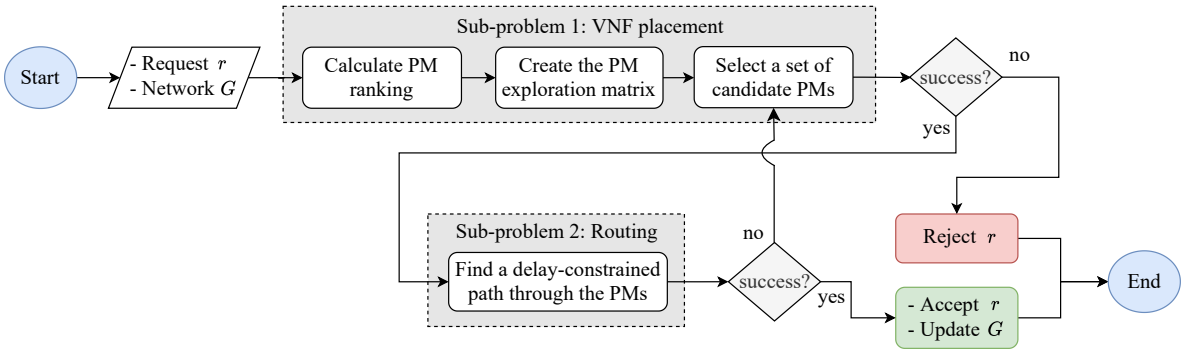



**Figure 2.4:** Flowchart of the *Holu* framework. Given the user request $r$ (Eq. 2.1) and the network graph $G$, *Holu* solves the PQ-VPR problem by collaboratively performing VNF placement (Sub-problem 1) and routing (Sub-problem 2). In Sub-problem 1, using a node ranking mechanism, the PM exploration matrix is formed. Then, a set of candidate PMs are picked using a PM selection mechanism. Thereafter, in Sub-problem 2, a routing algorithm attempts to realize the requested SFC by routing the traffic from the source node to the destination node through the candidate PMs returned by Sub-problem 1. If the path meets the required delay, the algorithm accepts the request $r$ and registers it in the network. Otherwise, in the next iteration, it attempts to find a new delay-constrained path using the second candidate PMs set. After a limited number of unsuccessful iterations, *Holu* rejects the user request $r$.

$k_1$ and $k_2$. Therefore, they cannot efficiently distribute the cost/constraint budgets between different segments (i.e., VNF-VNF) of a SFC (budget-division problem). This can potentially reduce the quality of found solutions dramatically as well as the acceptance rate. In contrast, *Holu* employs an end-to-end decision-making strategy to solve the PQ-VPR problem. This strategy consists of placing all the VNFs at once and then finding the best route, both considering the power consumption. In this way, we can solve the budget-division problem which exists in the state-of-the-art sequential solutions. Moreover, a fallback mechanism improves the quality of the solution in an iterative way.

To do so, we divide the PQ-VPR in two sub-problems, regardless of the length of the requested SFC: *i)* Sub-problem 1: VNF placement to minimize $P^T_{pm}$, and *ii)* Sub-problem 2: routing aiming at minimizing $P^T_{net}$ while meeting the delay constraints. As depicted in the flowchart in Fig. 2.4, having a request $r$ in hand, the VNF placement sub-problem finds a set of PMs to host the VNFs $k \in \mathcal{C}_r$. Particularly, *Holu* uses a ranking mechanism to assign a score to the candidate PMs for hosting the VNFs requested in the $\mathcal{C}_r$. Thereafter, it builds a matrix (referred as *exploration matrix*) to represent the candidate PMs that can host VNFs $k \in \mathcal{C}_r$ and sorts them based on the calculated ranking values. According to this sorting, a potential VNF placement solution (the PMs to host VNFs $k \in K$) is selected and passed as an input to Sub-problem 2, which attempts to find a routing through these PMs, considering the delay constraint $\mathcal{D}_r$. To do so, we employ our previous work [207] to find a delay-constrained path to route the traffic through the candidate PMs in a power-aware manner. If the routing is not successful (no path is found with respect to delay/capacity constraints), another set of candidate PMs is selected by Sub-problem 1, and the routing is recomputed. Note that since it is not straightforward to select a *good* set of PMs at once (as it would imply solving the whole problem at once), there is an iteration in order to improve the solution gradually. This loop is repeated either until a successful routing is found, which leads to accepting the request $r$, or a stopping condition (e.g., VNF placement returns no result), in which case the request $r$ is rejected. An illustrative example is presented in Fig. 2.5.

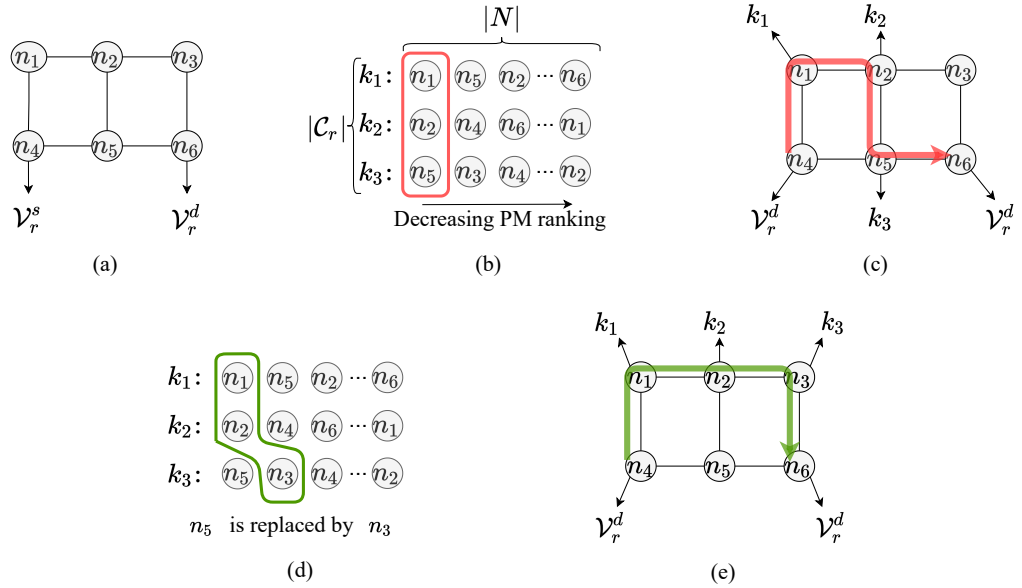



**Figure 2.5:** A high-level example that illustrates the steps that the *Holu* framework takes to find a solution for a request $r$. The request $r$ is defined with source node $\mathcal{V}_r^s = n_4$, destination node $\mathcal{V}_r^s = n_6$, and the requested SFC as $C_r = \{k_1, k_2, k_3\}$ (in this example we do not consider/show the data rate $\mathcal{B}_r$). (a) Physical network graph $G$ with $|N|$=6 nodes, where the request $r$ must be embedded. (b) Considering the graph $G$ and the request $r$, the PM exploration matrix $\mathbb{M}^r$ is formed in this sub-figure. There are $|N|$=6 columns and $|C_r|$=3 rows. Each row contains the candidate PMs for the respective VNF, sorted by their ranking value $\gamma_n^{r,k}$ (e.g., considering the first row, $\gamma_{n_1}^{r,k_1} > \gamma_{n_5}^{r,k_1} > \gamma_{n_2}^{r,k_1} > \ldots > \gamma_{n_6}^{r,k_1}$). It also shows the first set of candidate PMs that are chosen to host the VNFs in $C_r$ is $\mathcal{S}_r = \{n_1, n_2, n_5\}$. (c) As determined by the PM exploration matrix, the PMs in the first column $n_1, n_2, n_5$ are the candidate PMs to host $k_1$, $k_2$, and $k_3$, respectively. In the first iteration, the delay-constrained shortest-path algorithm (Sub-problem 2) should find a path meeting all the routing constraints. In this example, there is not such a path (e.g., the path $(n_4 - n_1 - n_2 - n_5 - n_6)$ does not meet the delay constraint). (d) After the unsuccessful path search in the first iteration, the PM selection mechanism selects the next best PM candidate set from the matrix $\mathbb{M}^r$ by replacing $n_5$ by $n_3$ for hosting $k_3$, having $\mathcal{S}_r = \{n_1, n_2, n_3\}$. (e) In the second iteration, the routing algorithm successfully finds a path $(n_4 - n_1 - n_2 - n_3 - n_6)$ that meets the delay and capacity requirements. Finally, this path as well as the VNF configuration on PMs $n_1$, $n_2$, and $n_3$ are registered in the network.

As a result, by identifying the VNF hosting PMs for the SFC (Sub-problem 1) and finding a path passing through these PMs (Sub-problem 2), we are able to distribute the cost (power) and the delay budget between different segments of the SFC. In below, the details of these two sub-problems and their respective solutions are presented.

### 2.5.1 Sub-problem 1: VNF Placement

As mentioned before, the VNF placement process identifies to map each of VNFs in the requested $C_r$ on a PM in network with the goal of minimizing the first element of the objective function, $P_{pm}^T$ (see the ILP model (2.17)). In order to improve power-efficiency, we focus on increasing the reusability of VNFs by sharing them between more SFCs, thus, increasing their utilization and minimizing the number of online PMs (i.e., improving power-proportionality, see Fig. 2.3). Therefore, it is critical for the VNF placement process to identify the PMs that maximize the VNF reusability factor. We achieve this goal by introducing a PM ranking strategy, as explained below.

#### 2.5.1.1 PM Ranking Mechanism

Given the request $r$, we determine a set of PMs that are the *best* candidates for hosting each VNF $k \in C_r$. To do so, we employ a mechanism that assigns a ranking value to each PM, according to the requested SFC $C_r$. We define the score of PM $n \in N$ to host VNF $k \in C_r$ as $\gamma_n^{r,k}$:

$$\gamma_n^{r,k} = \alpha_n^{r,k} + \beta_n^{r,k}, k \in C_r, n \in N, \tag{2.18}$$

In following, we define $\alpha_n^{r,k}$ and $\beta_n^{r,k}$ in detail.

*Node Centrality Impact ($\alpha_n^{r,k}$):* Given a network topology, a wide spectrum of centrality metrics can be defined in order to find the most *critical/important* node(s) in a network [49, 182]. Metrics relying only on the node degree have been shown not to improve the resource reusability as they incur higher network power consumption or too long delays [182]. Particularly, if the resource is available in *good* locations in the network, according to the properties of centrality measures, the probability of resource reusability in the network is expected to be increased. Hence, in this thesis, three centrality metrics relying on the length of shortest-paths between all node pairs are used to determine the value of $\alpha_n^{r,k}$: betweenness, closeness, and Katz centrality [49, 182]. Using one of the three chosen metrics, we calculate the centrality value for all the nodes $n \in N$ and then normalize the obtained values between 0 and 1 to obtain $\alpha_n^{r,k} \forall n \in N$.

*Power Consumption Impact ($\beta_n^{r,k}$):* This metric aims at prioritizing PMs which causes the minimum increase in the power consumption to process a new user request $r$ (either by creating a new VNF instance or using an existing one). The $\beta_n^{r,k}$ can take three different values based on the state of a PM $n \in N$ with respect to VNF $k \in C_r$ (the power consumption caused by hosting $k$ is increasing from first to the third case):

- $\beta_n^{r,k} = 1$: The PM $n$ is *online* and already hosts an instance of VNF $k$ with enough available capacity to process the data rate $\mathcal{B}_r$.
- $\beta_n^{r,k} = 0.1$: The PM $n$ is *online* and has sufficient resources to launch a new instance of VNF $k$.
- $\beta_n^{r,k} = 0$: The PM $n$ is *standby* and must be turned on to launch a new instance of VNF $k$.

The $\beta_n^{r,k}$ values can be tuned by the operator e.g., based on the power-proportionality of PMs and/or the size (impact) of specific VNF types that is being used.

After determining the values of $\alpha_n^{r,k}$ and $\beta_n^{r,k}$, upon receiving an user request $r$, for each $k \in C_r$, the values of $\gamma_n^{r,k}$ can be computed. Note that the weights of $\alpha_n^{r,k}$ and $\beta_n^{r,k}$ metrics can be changed according to the operator preferences/priorities.

#### 2.5.1.2 PM Exploration Matrix

As mentioned before, given a request $r$, we would prefer to place the VNF $k \in C_r$ on a PM with the highest $\gamma_n^{r,k}$ value. Here, the challenge is how to select this set of candidate PMs, which is denoted by $\mathcal{S}_r$, where $|\mathcal{S}_r| = |C_r|$. To do so, we define a PM exploration matrix $\mathbb{M}^r_{|C_r| \times |N|}$ which represents the search space of the candidate PMs: each row represents a set of candidate PMs to host a VNF $k \in C_r$, which are sorted in decreasing order according to their $\gamma_n^{r,k}$ value. Potentially, since all the PMs in

the network are candidate nodes, there are $N$ columns in the matrix $\mathbb{M}^r$. Fig. 2.5b shows an example of a PM exploration matrix $\mathbb{M}^r$ for a user request $r$. In this exemplary matrix, there are three (i.e., $|C_r|$) rows, one per VNF in $C_r$, and four (i.e., $|N|$) columns.

#### 2.5.1.3 PM Selection Mechanisms

The VNF placement problem consists in selecting one PM from each row in the matrix $\mathbb{M}^r$. Since the PMs in each row are sorted based on their $\gamma_n^{r,k}$ value, the first (and the *best*) choice of PM candidates is the first PM from each row in $\mathbb{M}^r$. For example, in Fig. 2.5b, the best (first) candidate PMs to host VNFs $\{k_1, k_2, k_3\}$ are supposed to be $n_1$, $n_2$, and $n_5$, respectively.

Having the set of candidate PMs $\mathcal{S}_r$, a path from $\mathcal{V}_r^s$ to $\mathcal{V}_r^d$ passing through the sequence of PM nodes $n \in \mathcal{S}_r$ should be calculated. However, such a path might not be found because of the routing algorithm's incompleteness or capacity/delay constraint violations. If this happens, we retry to find a path for an updated set of candidate PMs. In our approach, we select a PM from the current $\mathcal{S}_r$ and replace it with the next PM with the highest ranking value from the respective row in matrix $\mathbb{M}^r$. For example, in Fig. 2.5d, after no solution is found for the first candidate PM set $n_1, n_2, n_5$ (Fig. 2.5b), the PM $n_5$ is chosen to be replaced by the next PM $n_3$ to form the new candidate PM set $n_1, n_2, n_3$.

The question here is which PM from the current candidate PMs set $\mathcal{S}_r$ should be replaced. Since different candidate PM sets can lead to different solutions with different qualities, it is crucial to determine how to select the next set of candidate PMs, which is expected to reduce the power consumption while allowing a path through them guaranteeing the constraints. Having the PM exploration matrix in hand, we propose two candidate PM selection mechanisms to determine the next candidate PM set:

**i) Highest Node Ranking (HNR):**
The first approach uses the $\gamma_n^{r,k}$ metric to select and replace a PM from the candidate PMs set $\mathcal{S}_r$. HNR selects the candidate PM with the highest $\gamma_n^{r,k}$ value among the current set of PMs. Then the selected candidate PM is replaced by the next PM in the corresponding row of $\mathbb{M}^r$. Thereby, this approach relaxes the power savings to meet the end-to-end delay and, in turn, increases the acceptance ratio.

**iii) Least Sub-path Delay (LSD):**
The second approach uses the sub-path delay metric to identify and prune a PM from the candidate PM set. To do so, we first divide the shortest-path from $\mathcal{V}_r^s$ to $\mathcal{V}_r^d$ through candidate PMs of $\mathcal{S}_r$ into $|C_r|$ sub-paths: $\mathcal{V}_r^s$ to $k_2$ (here $k_1$ is the intermeddiate VNF), $k_1$ to $k_3$, and $k_{|C_r|-1}$ to $\mathcal{V}_r^d$. Each sub-path contains a single PM hosting a VNF as an intermediate PM. Then, for each sub-path, we calculate the shortest path from the start to the end node of the sub-path with the delay as the cost function. Afterward, we prune the PM which is an intermediate PM of the sub-path with the largest sub-path delay from the current PM set. As a result, the candidate PM with a high contribution to the end-to-end delay of the original path is replaced with the next PM from the respective row in the matrix $\mathbb{M}^r$. Fig. 2.6 shows an example where a path and its delay from $\mathcal{V}_r^s$ to $\mathcal{V}_r^d$ through $k_1$, $k_2$, and $k_3$ VNFs is shown.

The heuristic continues to try different candidate PM sets $\mathcal{S}_r$ until a solution meeting the delay and capacity constraints are found, or a stopping condition is reached (see Fig 2.4). This condition

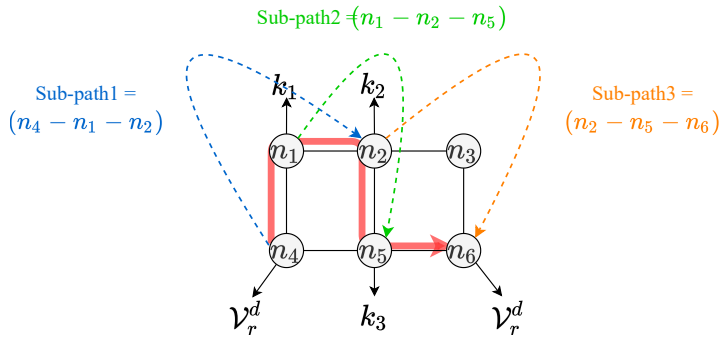



**Figure 2.6:** An example for the LSD PM selection method. Considering the failed solution from the example in Fig. 2.5c, a total $|C_r|$ sub-paths are formed, each corresponding to a VNF $k \in C_r$. Moreover, each sub-path has a certain total delay based on the propagation delay of each of its links. In this example, the delays of sub-paths 1, 2, and 3 are assumed to be 4, 6, and 8 ms respectively. As the third sub-path (corresponding to VNF $k_3$) has the highest delay among the sup-paths, the LSD removes the PM at $n_5$ hosting $k_3$, and looks for another PM, according to the matrix $\mathbb{M}^r$ (see Fig. 2.5d).

can be chosen by the operator based on the network size and the service provider's decision time constraints.

### 2.5.2 Sub-problem 2: Routing

In this section, we present the solution to the routing sub-problem which aims at minimizing the second element of the objective function, $P_{net}^T$ (Eq. 2.17). At this stage, the user request $r$ and an ordered set of candidate PMs $\mathcal{S}_r$ are given by Sub-problem 1. The problem now, is to find a path from source $\mathcal{V}_r^s$ to destination $\mathcal{V}_r^d$, that traverses the candidate PMs in $\mathcal{S}_r$, and that satisfies the delay constraint $\mathcal{D}_r$. Given a cost function that models the power consumption of a path, the problem is to find a DCLC path from a given source $src$ to destination $dst$, which can be modeled as:

$$z^*(src, dst) = \min_{\mathfrak{p} \in \mathbb{P}_{src,dst}} \sum_{(i,j)\in\mathfrak{p}} c_{i,j} \qquad (2.19)$$
$$\text{s.t.} \sum_{(i,j)\in\mathfrak{p}} D_{i,j} \leq \mathcal{D}_r$$

where $\mathbb{P}_{src,dst}$ is the set of paths from node $src$ to $dst$, and $c_{i,j}$ and $D_{i,j}$ are the cost (i.e., network power consumption) and delay functions of link $(i, j)$ respectively.

In the following, we show how to solve Sub-problem 2 by using the LARAC algorithm [132] and an extension of it, Lagrange Relaxation based Aggregated Cost with Specific Nodes (LARAC-SN) [207]. Let us first shortly describe these two algorithms.

#### 2.5.2.1 Lagrange Relaxation based Aggregated Cost (LARAC)

The DCLC problem is NP-hard [103]. Accordingly, numerous heuristics have been proposed to find close to optimal solutions [110] quickly. The LARAC algorithm finds a DCLC path in a graph by running several times the Dijkstra shortest-path algorithm. It is commonly considered as one of the best performing heuristics for the DCLC problem [110, 207]. The algorithm is based on the Lagrange relaxation, a mathematical optimization technique for solving constrained optimization problems.

Using the Lagrange relaxation principle, a heuristic solution $z^R(s,t)$ to the DCLC problem (Eq. 2.19) can be found by optimally solving [110]:

$$z^R(\mathcal{V}_r^s, \mathcal{V}_r^d) = \max_{\lambda \in \mathbb{R}^+} \left( \min_{\mathfrak{p} \in \mathbb{P}_{\mathcal{V}_r^s, \mathcal{V}_r^d}} \sum_{(i,j) \in \mathfrak{p}} c_{i,j} + \lambda \left( \sum_{(i,j) \in \mathfrak{p}} D_{i,j} - \mathcal{D}_r \right) \right). \tag{2.20}$$

Solving this maximization problem requires solving several times the inner minimization problem (called *relaxed problem*) by varying the $\lambda$ parameter. Interestingly, for this problem, the relaxed problem corresponds to a shortest-path problem with a modified cost function $c^\lambda = c_{i,j} + \lambda \times D_{i,j}$ [132]. As a result, LARAC finds a heuristic solution to the DCLC problem by successively running Dijkstra with the modified cost function $c^\lambda$ and by adapting the $\lambda$ at each iteration to converge to the maximum of Eq. 2.20.

LARAC starts by finding the least-delay and least-cost paths in the network from a given source node *src* to destination *dst*. If the least-delay path does not exist, it returns null since there is no path that meets the delay constraint. If the least-cost path ($\lambda = 0$) does not violate the delay constraint, the path is returned as the solution (which is the optimal solution for sub-problem 2). Otherwise, if the delay of the least-cost path is higher than $\mathcal{D}_r$, it performs subsequent Dijkstra runs with the modified cost function $c^\lambda$. At each iteration, $\lambda$ is adapted in order to give more or less importance to the cost and/or the delay. More details on the rationale behind the convergence strategy are available in the original reference [132] and the survey [110]. The algorithm is *complete*, i.e., it always finds a solution if it exists but is not optimal. However, it was shown that its optimality gap is quite low [110] while keeping a relatively low runtime. Thus, it becomes a perfect candidate to be used as part of the solution to Sub-problem 2 that we repeatedly solve to obtain a solution to our original problem.

#### 2.5.2.2 Lagrange Relaxation based Aggregated Cost with Specific Nodes (LARAC-SN)

LARAC-SN builds on top of LARAC to force the traversal of an ordered set of nodes [207]. To do so, the algorithm adapts the underlying Dijkstra procedure used by LARAC. Instead of running Dijkstra from the source to the destination node, LARAC-SN splits the Dijkstra run from the source node to the first VNF, from the second VNF to the next one, and so on until the destination. By doing so, the algorithm ensures that LARAC only considers paths that traverse the set of candidates PMs and hence that the final path satisfies this constraint. As we ensure that each segment path returned by LARAC always traverse the VNFs of $\mathcal{C}_r$ (i.e., the PM nodes in $\mathcal{S}_r$), the path returned by LARAC-SN will also traverse the VNFs. Splitting the underlying Dijsktra run rather than the LARAC run removes the need for splitting the delay constraint between the different VNFs. That is automatically handled by the LARAC procedure by adapting the weights of the aggregated cost function, which can significantly increase the acceptance rate of the requests. Inheriting the properties of LARAC, LARAC-SN is complete and close-to-optimal [207].

#### 2.5.2.3 Routing Metrics: Delay, Cost, and Capacity

The first metric is the delay. It is a global constraint metric. The delay of a link corresponds to its propagation delay (the VNF processing delay can be pre-computed and subtracted from the delay constraint budget $\mathcal{D}_r$), thus, its value for a link is static for a given routing request. The second

metric is the cost which is a global optimization metric. Our cost function essentially considers the power consumption impact of taking a given path. The cost of a link $(i, j)$ is defined as

$$c_{i,j} = \hat{\mathcal{P}}_{i,j} + \hat{Q}_j, \tag{2.21}$$

where the $\hat{\cdot}$ notation represents the fact that both values are normalized to a value between 0 and 1. As it can be seen, $c_{i,j}$ consists of two cost elements. The first one is the impact of power consumption $\hat{\mathcal{P}}_{i,j}$ for using a given link $(i, j)$ and it is computed as

$$\hat{\mathcal{P}}_{i,j} = \frac{1}{2} P^{idle}_{switch} \times (2 - y_i - y_j) + 2P_{port} \times (1 - q_{i,j}), \tag{2.22}$$

where $y_i$ and $q_{i,j}$ are binary variables reporting whether the switch $i$ or link $(i, j)$ are already powered on, respectively. In fact, variables $y$ and $q$ allow penalizing the usage of a switch or link that is not yet used (i.e., that is in the *standby* state). The second term forming the $c_{i,j}$ is the centrality metric $\hat{Q}_j$ of the destination node $j$ of the link. The reason for adding this term is that the power consumption of links can often be equal, as only six different values are possible (based on the $y$ and $q$ variables). As a result, the routing algorithm will often face equal-cost paths. Instead of letting the algorithm pseudo-randomly choose among these paths, the centrality term acts as a tie-breaker to direct the algorithm towards more central nodes that tend to be used more often and hence reduce the power consumption of subsequent requests. We use the betweenness centrality value of the node $j$ for the $\hat{Q}_j$. Computing the value of our cost function $c_{i,j}$ requires knowing all the links previously visited by the routing algorithm. Indeed, since the state $s_{(k,l)}$ of a link depends on whether it is used already (i.e., whether we already visited it), knowing all the previously visited links is necessary to compute $\hat{\mathcal{P}}_{i,j}$ and hence $c_{i,j}$. In [206], it is shown that such a metric as a global optimization metric increases the optimality gap of an algorithm. However, the algorithm stays complete.

The third and last metric we use is the capacity of links. It is a local constraint metric. A link can only be used if it still has sufficient capacity to host the new flow. Similar to the cost function, checking if the new flow can be accommodated by the remaining capacity at a link requires knowing whether we visited this link already. In traditional routing, it is not the case because we know the remaining capacity of all links in advance, and we do not route into loops. However, when routing through a set of VNFs, we can potentially route through loops. Unfortunately, such a metric as a local constraint metric makes the routing algorithm *incomplete* [206]. Recovering the completeness of algorithms in such a situation leads to intractable runtime [206]. In fact, this is one of the reasons for which we have to iterate back and forth between sub-problems 1 and 2. Indeed, once $\mathcal{S}_r$ has been selected in Sub-problem 1, the algorithm used in Sub-problem 2 cannot guarantee (because it is not complete) to find a solution for the $\mathcal{S}_r$, even if it exists. As we want to maximize the acceptance of flow requests, we iterate between the two sub-problems until the routing algorithm finds a solution. In the evaluation section, we show that the incompleteness of the algorithm is not very high, and hence solutions can be found in relatively few iterations and hence, short runtime.

## 2.6 Performance Evaluation

### 2.6.1 Simulation Setup

This section provides details about the network topologies and the simulation parameters considered for the evaluation. The results presented in this section correspond to Nobel-EU [168], and Internet2 [196]. Each network node is equipped with a switch and a PM for hosting VNFs. The PM resources are characterized by the number of CPU cores, which has been set to 16. The PM power consumption is computed in accordance to Eq. 2.3. The idle and maximum power consumption of the PM is chosen as $P^{pm}_{idle}$=299 Watts and $P^{pm}_{max}$=521 Watts, respectively [194]. On the other hand, the network switch power consumption consists of static hardware power and network interface power (see Eq. 2.2). The static hardware power of the switch $P^{idle}_{switch}$ is set to 315 Watts [210]. The network interface power consumption of switch port varies based on the data rate configuration, which is modeled to operate at three different configurations: 100 Mbps, 1 Gbps, or 10 Gbps with 26 Watts, 30 Watts, and 55 Watts of power consumption, respectively [61, 155]. The physical link data rate capacity is set to a randomly generated value between 6 to 10 Gbps. To generate the user requests, the source, and destination pairs are generated randomly, such that $\mathcal{V}^s_r \neq \mathcal{V}^d_r$. We have considered commonly used service types: video streaming, web service, voice-over-IP (VoIP), and online gaming [128, 13]. The VNFs forming the SFCs have considered with different resource requirements and processing capacities as listed in Table 2.3 [36, 99]. Thus, as it is presented in Table 2.4, each of these service types requires a specific SFC, data rate, maximum end-to-end delay, and the share of the service type in the set of user requests.

**Table 2.3:** The number of CPU cores and their processing capacity (maximum throughput) of each VNF type.

| VNF Type | Required CPU Cores ($\Delta_{k,cpu}$) | Processing Capacity ($\Phi_k$) |
|---|---|---|
| Network Address Translation (NAT) | 2 | 500 Mbps |
| Firewall (FW) | 8 | 400 Mbps |
| Traffic Monitor (TM) | 1 | 200 Mbps |
| Video Optimization Controller (VOC) | 2 | 580 Mbps |
| WAN Optimization Controller (WOC) | 2 | 300 Mbps |
| Intrusion Detection and Prevention System (IDPS) | 8 | 600 Mbps |

**Table 2.4:** Overview of service types and their SFC set, data rate, end-to-end delay, and share of the total requests. We have considered request aggregation by considering a range of data rate to mimic the traffic coming from multiple users at the same time from a single network node.

| Type | Required SFC ($C_r$) | Data Rate Requirement ($\mathcal{B}_r$) | Delay Requirement ($\mathcal{D}_r$) | Share |
|---|---|---|---|---|
| Web | NAT-FW-TM-WOC-IDPS | U(0.6-1) Mbps | 500 ms | 18.2% |
| VoIP | NAT-FW-TM-FW-NAT | U(0.384-0.64) Mbps | 100 ms | 11.8% |
| Streaming | NAT-FW-TM-VOC-IDPS | U(24-40) Mbps | 100 ms | 69.9% |
| Gaming | NAT-FW-VOC-WOC-IDPS | U(0.24-0.5) Mbps | 60 ms | 0.1% |

To choose the best centrality metric and PM selection mechanism for *Holu*, we have extensively evaluated them in terms of power consumption and acceptance rate. Thus, for each node $n \in N$,

we use closeness centrality as the metric for the node centrality impact $\alpha_n^{r,k}$ and the LSD as the PM candidate exploration strategy (supportive evaluations are presented in the results section). The proposed ILP optimization model (2.17) has been implemented using Gurobi solver [113] in Python. Besides, *Holu* framework and the state-of-the-art approaches are simulated using Java. The simulations have been performed on a machine equipped with Intel Core i7-6700HQ @2.60 GHz, 16 GB of RAM, running Ubuntu 18.04.

### 2.6.2 Algorithms to Compare

We compare the proposed ILP model and the *Holu* heuristic with two state-of-the-art algorithms: *i)* A state-of-the-art approach named CPVNF [80], and *ii)* The BCSP algorithm, which was presented in our previous work [13] (referred as the BC-Based algorithm). These two algorithms are briefly explained below:

#### 2.6.2.1 Cost-efficient Proactive VNF placement and chaining (CPVNF)

The CPVNF algorithm [80] addresses the VNF placement and routing problem with the objective of minimizing the number of online PMs and communication costs while meeting end-to-end delay and resource capacities.

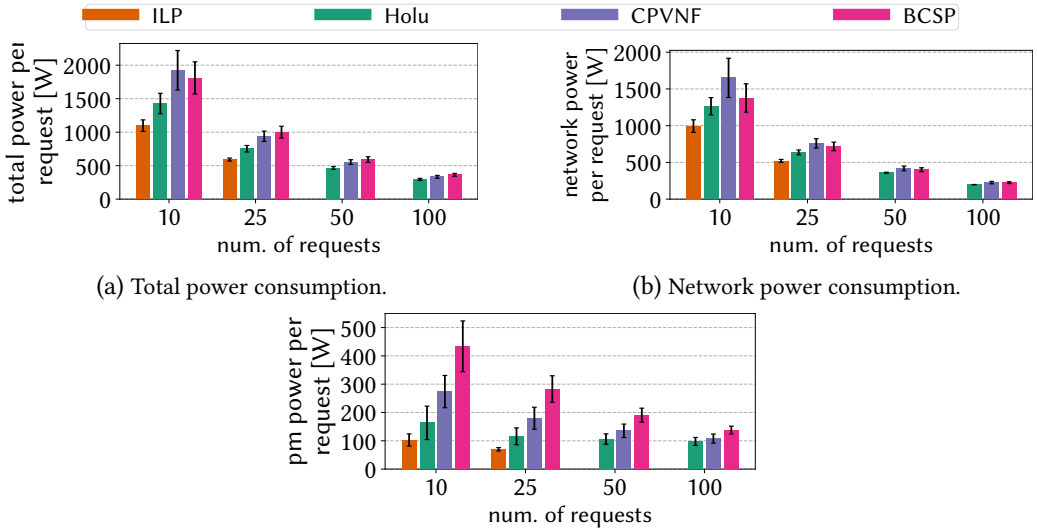
CPVNF acts sequentially, solving VNF per VNF. To select *high-quality* PMs in the network to host each VNF, it uses a ranking algorithm based on the Google PageRank method. PageRank is a variant of the Eigen Vector centrality measure [54] that assigns ranks to web pages based on both the quality and quantity of the web pages linked to it.

In comparison, the authors of CPVNF algorithms assign ranks to PMs based on the remaining capacity of the PM itself, the aggregated remaining bandwidth of its outgoing links, and the rank of its directly connected nodes [80]. Additionally, nodes are assigned personalized PageRank to bias towards PMs with the VNF instance under question.

For every request, the CPVNF algorithm uses the computed rank metric along with a delay function to select the VNF-hosting PMs. CPVNF uses the $k$-shortest-path algorithm to compute the routing at each stage and selects the path with the lowest delay value. If the end-to-end delay is less than $\mathcal{D}_r$, the user request is accepted in the network. Otherwise, a new set of VNF-hosting PMs is selected by increasing the importance of aggregate outgoing link bandwidth over the remaining CPU resource in node rank computation. The search continues until the iteration reaches the predefined search limit.

#### 2.6.2.2 Betweenness Centrality Shortest-Path (BCSP)

This algorithm represents the BC-based algorithm in our previous work [13]. This algorithm performs the VNF placement and routing sequentially for each VNF in the requested SFC. BCSP works based on the BC centrality metric for VNF placement and shortest-path for routing. It first calculates the BC metric for all the nodes in the graph. Then, for each request $r$, it calculates the shortest path between source $\mathcal{V}_r^s$ and destination $\mathcal{V}_r^d$ nodes using Dijkstra with delay as the cost function. Then, the BCSP algorithm tries to place the required VNFs of the $\mathcal{C}_r$ on the nodes with higher BC value on the shortest-path in a greedy manner. According to the BC definition, a higher BC value means a




(a) Total power consumption.

(b) Network power consumption.

(c) PM power consumption.

**Figure 2.7:** Comparison of total, network, and PM power consumption per user request for Internet2 topology.

higher number of shortest paths that are passing through a node $n$ and hence, a higher probability of VNF re-usability for future requests, which can lead to power-efficiency.

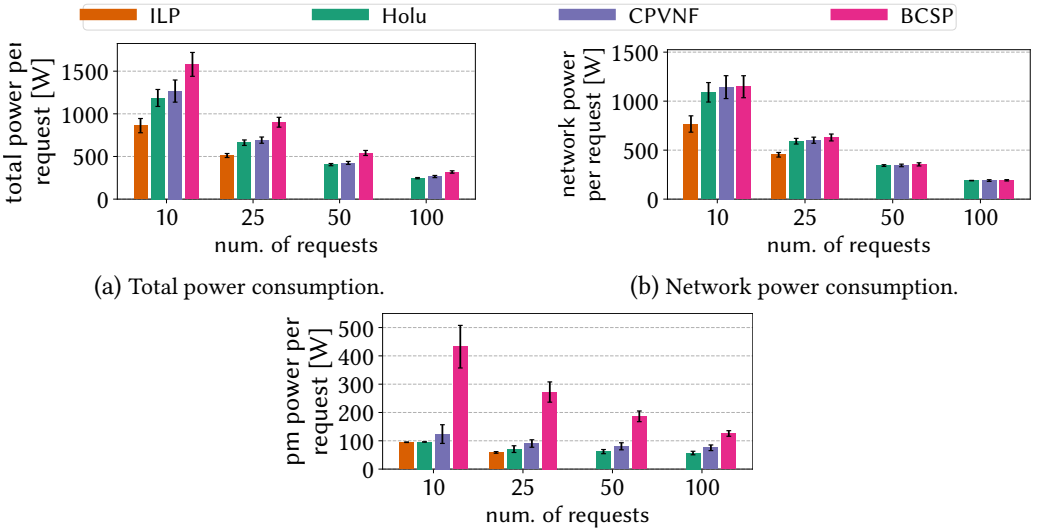
### 2.6.3 Simulation Results

This subsection summarizes the results of our extensive performance evaluation. Before presenting the results, we note that due to the scalability issues, we were able to run the ILP model for up to 25 user requests (more than 48 hours of computation time for a single problem instance with 50 requests).

#### 2.6.3.1 Power Consumption

Let us start comparing the total, PM, and network power consumption per accepted user request for the optimal case and the three heuristics for a different number of user requests. Fig. 2.7 and 2.8 show the mean power consumption per accepted request value for Internet2 and Nobel-EU topologies, respectively. Among all the heuristics, it can be observed that the *Holu* has the lowest total power consumption per request values in all the topologies. In more detail, *Holu* is performing on average 19.3% worst than the optimal solution, and outperforming CPVNF and the BCSP algorithms by 19% and 24.7%, respectively. However, the network gets saturated for a higher number of requests, and hence, the gap between the different heuristics decreases.

All the three heuristics utilize centrality measures to determine the PMs to place VNFs. However, it is evident from the results that the effectiveness of all the centrality measures is not the same. The performance of the CPVNF algorithm decreases when the network contains densely connected regions away from the graph center (e.g., more connected at the edges of the network). That is, unlike the ranking method of *Holu*, the PMs with higher ranks have more distance from other high-rank nodes, making it difficult to meet the required SFC delay constraint. For instance, Internet2 is from this group of graphs, which the power consumption of CPVNF is even worse than the BCSP




(a) Total power consumption.

(b) Network power consumption.




(c) PM power consumption.

**Figure 2.8:** Comparison of total, network, and PM power consumption per user request for Nobel-EU topology.

algorithm, because BCSP is always using the shortest-path (cost is considered as delay) between source and destination (see Fig. 2.7).

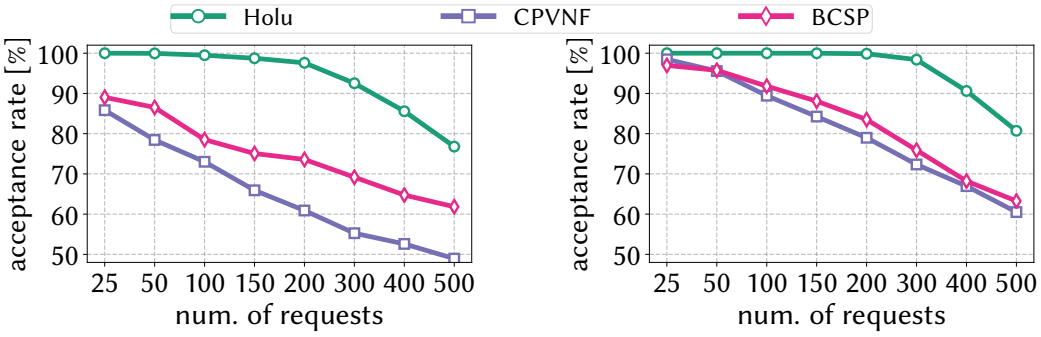
On the bright side, the *Holu* algorithm first ranks the PMs by using the centrality and their power consumption impacts. The closeness centrality value of a node depends on its position in the network for all other nodes. As a result, *Holu* can identify the PMs that increased the reusability of resources, leading to lower power consumption. The average number of online PMs per request have been summarized in Table 2.5 for 25 user requests. The lower number of online PMs per user request for *Holu* compared to the other heuristics reflects the effectiveness of the proposed PM ranking and selection approach. Concerning BCSP, the disadvantage when solving the VNF placement problem is that the search space is limited to the nodes along the shortest path between the request's source and destination. As a result, the resource reusability decreases, which leads to an increase of online PMs in both topologies.

**Table 2.5:** Comparison of the average number of online PMs per request for 25 user requests. *Holu* utilizes a fewer number of online PMs to serve the user requests compared to state-of-the-art approaches, thus improving the power-proportionality of the network.

| Topology | ILP | *Holu* | CPVNF | BCSP |
|---|---|---|---|---|
| Internet2 | 0.14 | 0.25 | 0.38 | 0.62 |
| Nobel-EU | 0.12 | 0.14 | 0.19 | 0.6 |

#### 2.6.3.2 Acceptance Rate

In our next analysis, we run *Holu*, CPVNF, and BCSP for up to 500 user requests to compare the acceptance rate for the two topologies. As we were able to run the ILP for a maximum of 25 requests, the feasibility of the problem for the larger inputs is unknown to us and requires further investigations. Also, we note that the ILP model either accepts all the requests or nothing; therefore, it is omitted




(a) Internet2 (b) Nobel-EU

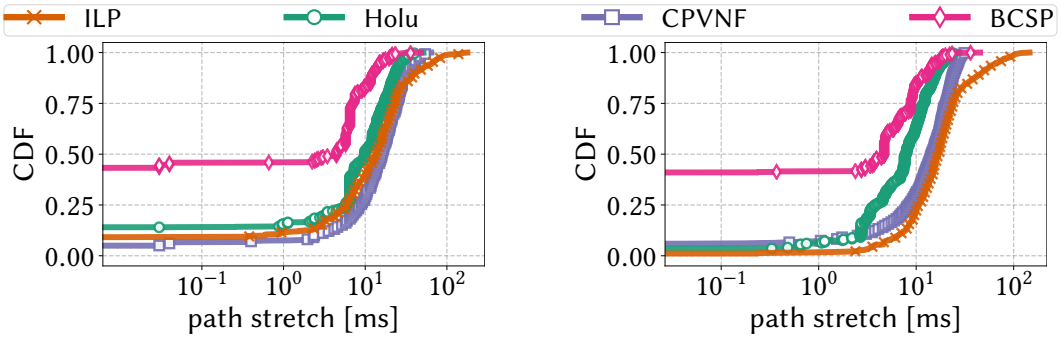
**Figure 2.9:** Comparison of the acceptance rate of *Holu* and baseline algorithms algorithms for different number of user requests. This figure shows that on average, *Holu* achieves 31% and 20% higher acceptance rate for Internet2, and 16.3% and 14.1% for Nobel-EU topology compared to the CPVNF and BCSP, respectively.

from the plots. As it can be seen in Fig. 2.9, *Holu* can accept 31% and 20% more requests for Internet2, and 16.3% and 14.1% for Nobel-EU topology compared to the CPVNF and BCSP, respectively. An essential parameter in successfully accepting a request is meeting its delay requirement. The results in Fig. 2.9 indicate that opposed to the other algorithms, by employing LARAC, *Holu* can divide the delay budget between the VNFs in the chain. In this way, it can increase its delay-awareness, which leads to a significant improvement in the acceptance ratio.

The CPVNF algorithm sorts the PMs of the network based on their ranking (i.e., according to the PageRank metric), ignoring the delay role. Thus, the node with a high ranking may be located far from the request's source and/or destination. As a result, the delay requirement of the user request may not be guaranteed, and hence, the request is rejected. Besides, the BCSP algorithm uses the BC metric for the PMs along the shortest-path to select the candidate nodes for VNFs. Therefore, a node with a high BC value may be closer to the user request's destination node. As a result, the BCSP algorithm needs to make a loop along the shortest path to realize the SFC. Under such circumstances, the actual SFC path computed by the BCSP is longer than the shortest path, which can lead to missing the delay requirement and rejecting the request. Also, we can see that the acceptance rate for BCSP is higher than CPVNF for the Internet2 topology and almost similar for the Nobel-EU. The Internet2 topology contains high-ranked nodes (according to PageRank metric) far away from each other and edges with considerable lengths (i.e., higher delay). As a result, the CPVNF algorithm performs poorly as compared to the BCSP algorithm in meeting the delay requirement, hence reducing the user request acceptance rate. Therefore, it can be seen that the coordination of an efficient VNF placement and a power and delay-aware routing can play an essential role in the acceptance rate. Thereby, *Holu* can balance the power and delay requirement, achieving a higher acceptance rate.

#### 2.6.3.3 Path Stretch

The path stretch metric represents the end-to-end delay difference between the path computed by each algorithm and the actual shortest path (based uniquely on delay) between the source and destination request. This metric is important because a lower path stretch can be translated to lower network resource consumption, i.e., lower OPEX costs. Fig. 2.10 shows the CDF of path stretch val-




(a) Internet2 (b) Nobel-EU

**Figure 2.10:** Path stretch comparison in case of 25 user requests. It can be seen that by outperforming the CPVNF and ILP, the average path stretch of *Holu* stands higher than the BCSP algorithm (which places the VNFs on the shortest-path). In both topologies, ILP produces the largest path stretch, since it focuses on maximizing the reusing of resources, which can lead to taking longer paths. Also, due to the properties of Internet2 topology, the maximum path stretch value is higher for Internet2 compared to Nobel-EU.

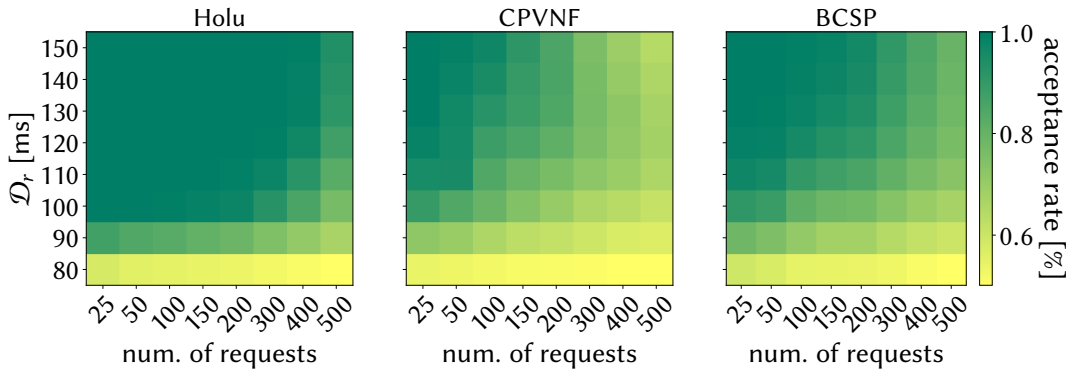
ues for ILP, *Holu*, CPVNF, and BCSP approaches for 25 user requests. We omit to show these results for other numbers of user requests since their behavior is similar.

It can be seen that in both topologies, the proposed *Holu* heuristic achieves the minimum-maximum path stretch compared to all the approaches, followed by CPVNF, BCSP, and the ILP. The reason is that the delay-aware routing algorithm can efficiently balance the importance of the power vs. the delay metrics. That is, the ILP shows an extremely high path stretch due to focusing on power minimization, while *Holu* causes a sound path stretch while having a lower power consumption compared to the other heuristics. Also, the plots indicate that the ILP solution has many instances with high path stretch values. The reason is that the ILP makes the requests traverse through VNF hosting nodes that are not necessarily close to the source and/or destination of the request to reduce the total power.

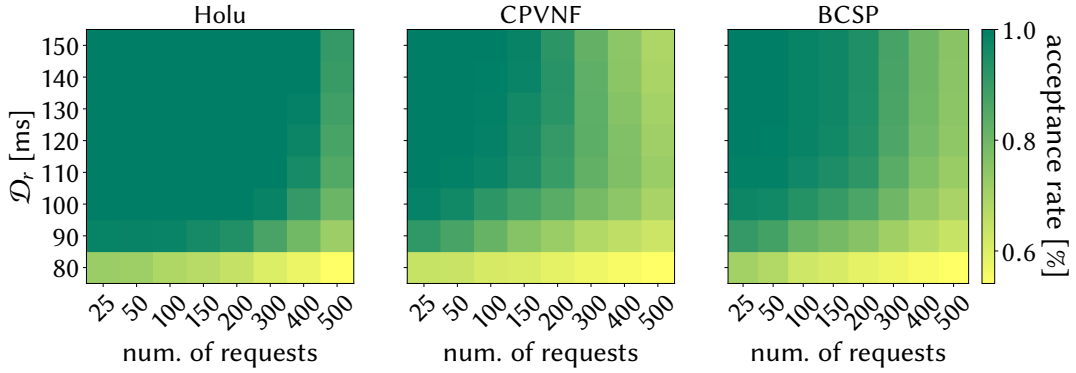
#### 2.6.3.4 Delay Tolerance

This study aims to compare the acceptance rate of *Holu*, CPVNF, and BCSP algorithms for user requests with varying delay requirements. The goal is to explore how different algorithms can handle user requests with tight or loose delay requirements. For this purpose, we simulate different sets of user requests, ranging from 25 to 500. Each of these user requests is generated with random source and destination pairs, 4 Mbps data rate, and a specific SFC type (see Table 2.4). Moreover, we select the delay values of user requests from 80 to 150 ms in steps of 10 ms. The goal is to check which algorithm is more sensitive to the delay variations in terms of acceptance rate. Fig. 2.11 shows the heatmap of the acceptance rate for a different number of user requests with varying delay requirements for both topologies.

These plots exhibit that *Holu* has a generally higher acceptance rate for all the delay ranges and the different number of requests. *Holu* emphasizes the delay parameter during the selection of VNF hosting nodes and routing process. Thereby, it succeeds in achieving a greater acceptance rate even under strict delay conditions. The CPVNF has the second-best performance after *Holu* and BCSP has




(a) Internet2



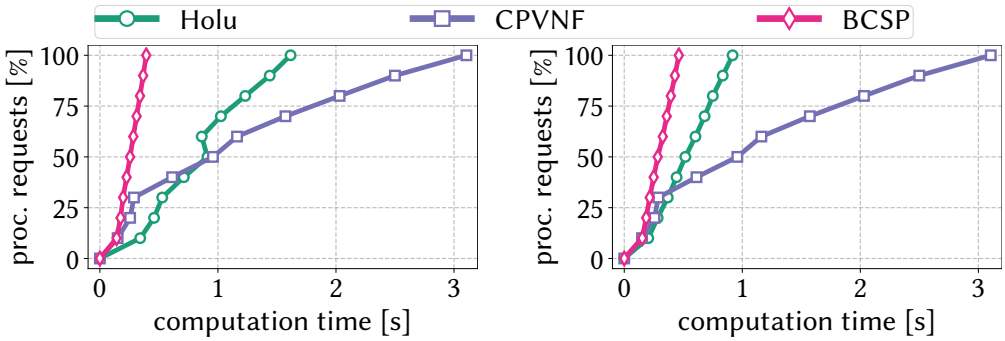

(b) Nobel-EU

**Figure 2.11:** Comparison of acceptance rate of *Holu*, CPVNF, and BCSP algorithms with respect to the number of user requests and the value of the delay constraint. This figure indicates that compared to others, *Holu* accepts more requests, even the ones with tight delay requirements. Also, due to the higher graph connectivity, more requests can be embedded into the Nobel-EU topology, compared to Internet2.

the worst performance. The reason is CPVNF applies a more sophisticated ranking method and also explores a more extensive solution space compared to the BCSP approach.

#### 2.6.3.5 Runtime

In this subsection, we compare the runtime of the algorithms. To do this, we record the computation time of each algorithm at the end of processing every 10% of the user requests. Due to space limitations, we only show the comparison of the runtime of different approaches for 100 user requests. We note that we omitted the ILP results from the plot because its runtime is in order of hours. Fig. 2.12 shows that *Holu* can make the admission and solve the PQ-VPR for a given user request in 10 ms for Nobel-EU, and 15 ms for Internet2 network topology. We note that the implementation of *Holu* can be optimized, e.g., by using more efficient data structures to achieve even lower runtime for practical use-cases. Besides, the results in Fig. 2.12 indicate that the BCSP algorithm is the fastest approach. The main reason behind it is the lower number of times required to run Dijkstra (only once per user request compared with several times for the other heuristics). For instance, *Holu* needs to compute the shortest-path several times iterative for specific requests (depending on the PM selection mechanism).




(a) Internet2 (b) Nobel-EU

**Figure 2.12:** The comparison of the runtime efficiency for *Holu*, CPVNF, and BCSP for processing 100 user requests.

On the other hand, the CPVNF and *Holu* approaches require to calculate PM ranking values before processing a user request resulting in higher runtime. After processing some of the initial 40-50% of the user requests, the CPVNF algorithm requires more time to process the remaining user requests compared to the other approaches. It is because it saturates the high-ranked PMs for serving the initial requests. As a result, it requires more iterations to find the VNF hosting PMs which meets the delay requirements for remaining user requests. Moreover, unlike our approach, the CPVNF needs to recompute the node ranks at every iteration, which leads to a higher runtime.

## 2.7 Summary

In this chapter, we have focused on resource allocation of an NFV/SDN-enabled network, with the use case of SFC. Motivating by low power-proportionality of both PMs and network switches, we formulate the PQ-VPR problem. Since the resource is usually under-utilized and/or over-provisioned, minimizing the number of online devices can improve resource utilization and more power-proportionality. To achieve this goal, we first proposed an ILP model. Due to its scalability issues, we presented an online heuristic framework named *Holu* which tackles the problem by breaking it into two sub-problems which are solved sequentially: *i) VNF placement*, and *ii)* routing. The VNF placement suggests a set of candidate PMs to host the requested SFC. It uses a PM ranking mechanism considering the centrality of the PM and the power consumption impact by hosting a VNF. We show that centrality is an important metric to consider for PM ranking since it can improve resource utilization and power efficiency in the long term. Also, to find a path traversing through the set of suggested candidate PMs (returned by the VNF placement sub-problem), we employed a complete algorithm based on LARAC heuristic to solve a DCLC shortest-path routing problem. We showed that our algorithm can efficiently split the delay budget between two consecutive VNFs in an SFC, leading to a high acceptance ratio compared to state-of-the-art algorithms.

Our simulation results indicated that compared to existing approaches, the end-to-end SFC placement, and routing approach, employed by *Holu*, can determine the VNF to PM mapping. Also, it can accurately split the delay budget among the routing paths between the consecutive VNFs in the SFC, improving its power consumption and acceptance rate up to 24.7% and 31%, respectively.

In contrast to this chapter where the user traffic is static, we consider the case where the traffic increases dynamically in the next chapter. The goal is to efficiently adapt and reconfigure the network to cope with the traffic increase.

# Chapter 3

# Reconfiguration Due to Traffic Changes

## 3.1 Introduction

In the previous chapter, based on the traffic requests of the users, we discussed the resource management of a programmable metro/core network, with a focus on delay-guaranteed placement and routing for Service Function Chaining (SFC) to save power (i.e., Operational Expenditures (OPEX)) and guarantee end-to-end delay. It was assumed that the traffic requests from users were static. However, according to the acknowledgment of both operators and analysts, the telecommunication industry is facing a drastic increase in the bandwidth request every year [152, 172]. In this chapter, by employing the modern dynamicity capabilities of flexible optical networks, e.g., Bandwidth Variable Transponders (BVTs), the network can be reconfigured in order to meet the traffic changes of the users/services. Thus, it is possible to minimize the resource over-provisioning and reduce the overall network costs, such as power consumption. The content of this chapter is based on the work published in [17], and [16].

### 3.1.1 Fixed-Grid Optical Networks

Due to their high transmission data rate over high distances, optical fibers are deployed in long-haul communication networks. Within the fiber, the C-band causes the lowest losses of the whole fiber spectrum; thus, it is being used for traffic transmission over long distances. Dense Wavelength-Division Multiplexing (DWDM) technology enables the transmission of several optical signal carriers over a single optical fiber by using different wavelengths. Currently, as a cheap commodity solution, the DWDM systems divide the C-band into equally-spaced discrete bands (usually by 50 or 100 GHz), which can deliver up to 100 Gbps of traffic, already standardized by International Telecommunication Union Telecommunication Standardisation Sector (ITU-T). An optical transponder then sets up an individual wavelength carrying the user traffic.

However, this approach is inflexible (i.e., called fixed grid optical networks) in two ways. Firstly, the transponder has a fixed bit rate, e.g., 10 or 40 Gbps. Indeed, in this case, it is possible to change some of the transponder configurations, such as Forward Error Correction (FEC); however, this does not change the overall transmission data rate. Although further transponders can be installed to support additional data rates, this is a very slow process, as new devices need to be installed and

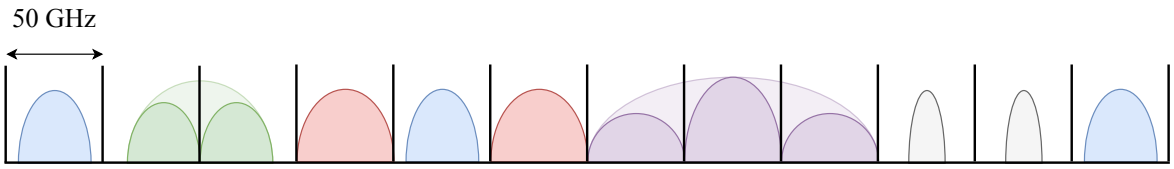



**Figure 3.1:** An example of a fixed-grid optical link with equal-sized 50 GHz slots.

provisioned. Additionally, very high bit rate transponders (i.e., 400+ Gbps) need a very broad optical spectrum; hence, they cannot be fitted into the grid.

Secondly, the spectral width of each wavelength signal is fixed, i.e., it cannot be extended beyond the ITU-T-defined width (e.g., 50 GHz). It means that the network is mainly inflexible to support changes in traffic requests. Therefore, in fixed-grid networks, requests with large bandwidth requirements are divided up and carried over multiple channels in the grid, leading to inefficient use of the network capacity and high number of transceivers. Fig. 3.1 shows a fixed-grid network, where it serves a range of user requests with different data rate requirements (bandwidths). It can be seen that some of these requests can fully fit between the 50 GHz grid boundaries, whereas others (e.g., *green* and *purple* requests) are too broad.

### 3.1.2 Flexible-Grid Optical Networks

In the flexible-grid[1] approach, unlike fixed-grid, the optical spectrum can be divided into more arbitrarily chunks of spectrum. This approach can be implemented by utilizing two principles: *i)* small chunks of bandwidth (e.g., 6.25 or 12.5 GHz), and *ii)* enabling concatenation of individual chunks to provide a more significant contiguous portion of the spectrum. Thus, flexible-grid networks are more manageable, cost-efficient, and dynamic. These networks, hand in hand with software-tunable transponders, can fulfill the capacity and dynamicity requirements of future long-haul networks by combining two technologies.

Fig. 3.2 shows the capability of flex-grid to pack the channels to make efficient use of the spectrum. In these networks, each node is a Reconfigurable Optical Add-Drop Multiplexer (ROADM) which can add, drop, or pass through the wavelength, enabling reconfigurable management of Lightpaths (LPs).

[1] Also called Flex-Grid or Elastic Optical Network (EON).

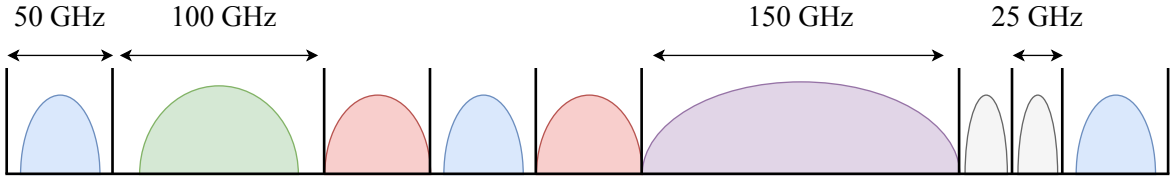



**Figure 3.2:** An example of a flex-grid optical spectrum, adaptable to requests width different bandwidth requirements.

### 3.1.3 Bandwidth Variable Transponders (BVTs)

In order to realize Flex-grid networks, by exploiting the capabilities of Software-Defined Networking (SDN), next-generation optical transponders can allocate channels over a flex-grid network, which are called BVTs. BVTs can reconfigure their channel configuration, e.g., modulation format, and data rate, to the actual traffic request by expanding or contracting the bandwidth of an optical LP (i.e. varying the number of sub-carriers). Also, they can dynamically change the other channel configuration parameters, such as modulation format and baud rate. Table 3.1 shows some examples of these different channel configurations with their characteristics.

By using BVTs, buying transponders with different bandwidths can be avoided, which can lead to huge cost savings [152]. Therefore, simply one transponder can deliver a wide range of bit rates if required. Moreover, BVTs can trade-off between optical reach and spectrum utilization. For example, if a request asks for 100 Gbps over 400 km distance, a highly spectrum-efficient modulation format, e.g., 16 QAM, can be used, which occupies only 25 GHz of bandwidth.

**Table 3.1:** Channel configuration examples of a software-tunable BVT [173].

| Data Rate (Gbps) | Modulation Format | Bandwidth (GHz) | Optical Reach (km) | Minimum OSNR (dB) |
|---|---|---|---|---|
| 100 | QPSK | 50 | 5000 | 10.5 |
| 250 | 8 QAM | 37.5 | 2000 | 19.8 |
| 400 | 32 QAM | 62.5 | 500 | 24 |
| 600 | 32 QAM | 75 | 150 | 29 |

### 3.1.4 Routing, Configuration, and Spectrum Assignment (RCSA)

A well-known problem in legacy fixed-grid optical networks is the Routing and Wavelength Assignment (RWA). Typically, the goal of RWA is to maximize the number of optical connections (i.e., LPs). In this problem, each connection goes through the same wavelength over the entire path (i.e., spectrum contiguity constraint). Unlike RWA that deals with fixed channel sizes, Routing and Spectrum Allocation (RSA) has to set up the LPs on a network with variable channel sizes (i.e., flex-grid networks). Thus, RSA is a more challenging problem to tackle. However, the increased flexibility introduced by the flex-grid networks and software-tunable BVTs has made the resource allocation and planning problems more complex to solve. In particular, in flex-grid networks, in addition to the conventional routing and spectrum allocation decisions (i.e., RSA problem), the configuration of the BVTs along with the physical layer constraints (e.g., minimum Optical Signal to Noise Ratio (OSNR)) need to be considered. This problem is called RCSA which has three main constraints:

1. The allocated spectral resources must be the same along with the links in the candidate path.
2. These allocated resources should also be contiguous in the spectrum.
3. The physical layer constraints such as minimum OSNR must be considered.

In more detail, a planning algorithm that solves the RCSA problem first finds the path (e.g., Dijkstra's shortest path) for source-destination pairs in the network. After that, for each of these requests,

it finds the set of channel configurations for BVTs, which can satisfy the physical transmission constraints (e.g., minimum OSNR). Then, using spectrum assignment, empty channel slots which can fit the configurations are found on all links of the chosen path to obtain a central channel frequency. Finally, the software-tunable BVTs at both the source and destination are tuned to this frequency, and a LP is established.

The RCSA problem can be used for *in-operation* network planning. In *in-operation* planning, the goal is to optimize the resources for the network operator. The optimization is usually performed periodically, based on the changes in the network traffic.

### 3.1.5 Key Contributions

In this chapter, on the given a flex-grid network and a set of traffic requests (matrix), we focus on solving the RCSA problem in a multi-period *in-operation* network planning scenario. In particular, we consider a yearly planning period and focus on answering the following three questions:

1. For each request, how many LPs should be deployed, and on which paths?
2. At each planning period, how to reconfigure the LPs to meet the required requests data rate (if necessary)?
3. How to answer the above questions with a low over-provisioning and power consumption?

Based on the characteristics of the flex-grid networks, we present three reconfiguration methods that can be applied to cope with the increased request: upgrading, adding, and/or rerouting LPs. We present a framework that can determine the reconfiguration actions based on the amount of increased request traffic. The performance evaluation results indicate that the reconfiguration methods can achieve a high network throughput compared to the state-of-the-art and cope with the increased traffic with low over-provisioning and high power efficiency.

### 3.1.6 Organization

The rest of this chapter is summarized as follows: Section 3.2 presents the related work. The mathematical modeling and problem formulation are presented in 3.3. In Section 3.4, we introduce the RCSA algorithm along with the different network reconfiguration methods. Finally, we present the evaluation of the RCSA algorithm in Section 3.5, followed by the summary of the chapter in Section 3.6.

## 3.2 Related Work

Prior studies for RCSA algorithms have used Integer Linear Program (ILP) formulations and/or heuristics to dynamically provision new LPs in a flex-grid optical network. Recently, for a multi-period planning scenario, Moniz et al. in [164] have proposed two separate ILP formulations for provisioning 32 and 64 GBaud channels, respectively. For upgrading in-operation LPs to a configuration with a higher data rate, a push-pull technique has been introduced by [74]. In their method, they have a dynamic reconfiguration to allocate additional frequency slots for the future upgrading

of the LPs. However, they do not consider rerouting and also the different paths to route the traffic. Also, they have only tested their approach for LPs with only ten and 100 Gbps of data rates. However, since many modern software-tunable BVTs can achieve variable symbol rates with the same number of frequency slots, their approach does not consider these requirements.

Further, Rottondi et al. [185] have provided a comprehensive solution for routing, modulation format, baud rate, and spectrum assignment using the few-mode transmission in metro-ring networks. Although the goal of their study is to compare few-mode fibers with single-mode fibers, they have shown that they can improve the overall savings on the spectrum usage using an ILP formulation. However, they have not considered higher symbol rates, potentially reducing the cost per bit even in single-mode fibers. Also, the network architecture in their study was an eight-node ring network with varying network radius. Therefore, it is not clear how their approach performs on long-haul optical networks with a high number of nodes and links.

Additionally, Ahmed et. al [24] have presented a novel RCSAs heuristic for LP provisioning. However, the effect of Nonlinear Interference Noise (NLIN) on LPs is not considered, which may lead to inaccurate configuration selection. Moreover, the assumption of semi-static traffic profiles with fixed distribution and a Poisson arrival of traffic requests may not hold when catering to unexpected traffic growth in the network.

On the other hand, several works have studied energy-efficient RCSA algorithms which try to minimize the total power consumption of transponders while meeting the simulated traffic requests [181, 92, 86]. For example, El-Mirghani et al. in [86] have presented a comprehensive power consumption analysis for different optical transponders based on their data rates. However, the assumption of approximately 1 W/Gbps power consumption for 100 Gbps, 400 Gbps, and 1 Tbps transponders does not hold for software-tunable BVTs [223]. However, in this chapter, we consider a base and dynamic power consumption for a BVT. In particular, the base power is required for operating the tunable laser, and the dynamic one is consumed for different data rates [223].

In [173], the authors have introduced a configuration selection algorithm considering NLIN. Also, an RCSA optimization for a multi-period network planning scenario has been presented in [172]. We have observed that in this optimization problem, in addition to meeting the increased yearly traffic, it is essential for network operators to reduce the total number of deployed LPs, which is proportional to the power consumption of the networks and plays a role in making strategic decisions for effectively planning optical networks [109]. However, deploying a low number of LPs can lead to under-provisioning in the future, which can lead to the need for more LPs later on. Therefore, considering the channel configuration of the LPs (data rate) can play a significant role in the efficiency of the RCSA algorithms. In this chapter, we present an algorithm for optimized RCSA using different objective functions and determine the number of LPs and their channel configurations in a multi-period *in-operation* planning scenario. Moreover, we provide an evaluation of the power consumption of our algorithm with different objective functions.

## 3.3 Mathematical Formulation

### 3.3.1 System Model

We define the network as a graph $G = (N, L)$ with $N$ nodes and $L$ links. Each node is an optical ROADM with add/drop capabilities. Also, each link consists of single-mode fiber pairs with heterogeneous span lengths [171]. Considering $T$ planning periods (years, in our case), the set of requests per year is defined as $\mathcal{R}^t, t \in T$. We define each yearly request set as $r \in \mathcal{R}^t$ as:

$$r_{i,j,t} = (i_r, j_r, b_{r,t}), \tag{3.1}$$

where $i, j \in N$ are source and destination nodes respectively, and $b_{r,t}$ is the requested data rate at year $t$. The value of $b_{r,t}$ is calculated based on $b_{r,t-1}$ and the expected yearly increase of $\Delta_{r,t}$, i.e., $b_{r,t} = b_{r,t-1} + \Delta_{r,t}$. The data rate requested by a request should be fulfilled by deploying one or more LPs in the network.

The set of deployed LPs for request $r$ in year $t$ is defined as $L_{r,t}$. Each of these LPs $l_c \in L_{r,t}$ is associated with an LP configuration $c \in C_r$, where $C_r$ is the set of feasible configurations for request $r$. This set of feasible configurations per request is determined based on our previous work [173]. Each configuration $c$ is defined as:

$$c = (R_c, Q_c, B_c, SNR_c), \tag{3.2}$$

Where $R_c$ is the data rate, $Q_c$ is the modulation format, $B_c$ is the channel bandwidth, and $SNR_c$ is the minimum receiver sensitivity OSNR. We note that to calculate the OSNR of each LP, both linear and non-linear noise are taken into account. The linear noise comes from the in-line amplifiers, while the NLIN is due to the cross channel interference of adjacent channels. The OSNR is calculated using a closed form equation of the Enhanced Gaussian Noise (EGN) model, based on [178].

The LPs in the network can be assigned to a path $p \in KSP_r$, where $KSP_r$ is the set of paths (e.g., k-shortest-path) for request $r$.

We consider the power consumption of the network based on the number of deployed LPs (using transponders) and their configured data rate [223]. When an LP is configured, a static amount of power $P_s$ is consumed. Additionally, there is a dynamic power consumption $P_d$ which depends on the data rate $R_c$ for the configuration $c$. Thus, the total power consumption $P_T$ of the network in each year is considered as:

$$P_T = \sum_{r \in \mathcal{R}^t} \sum_{l_c \in L_{r,t}} \left(P_s + P_d \times R_c\right), \forall t \in T. \tag{3.3}$$

We note that $P_s \gg P_d$ [223]. For clarification purposes, the notations have been summarized in Table 3.2.

In the following, we explain how we deal with the RCSA problem, using an efficient reconfiguration heuristic with different objectives and configurations.

## 3.4 The RCSA Algorithm

At the beginning of the planning (i.e., $t = 0$), the LPs, their configurations, and the path to deploy them are pre-selected by using the approach from our previous work, HeCSON [173]. Using this

**Table 3.2:** Notation definition for the RCSA problem.

| Notation | Definition |
|---|---|
| $G = (N, L)$ | The physical network graph with $N$ nodes and $L$ links. |
| $T$ | The set of planning years. |
| $\mathcal{R}^t$ | The set of requests in year $t$. |
| $r_{i,j,t}$ | The request with source/destination node $i/j$ at year $t$. |
| $b_{r,t}$ | The requested data rate of request $r$ at year $t$. |
| $\Delta_{r,t}$ | The increased data rate for request $r$ at year $t$. |
| $L_{r,t}$ | The set of deployed LPs for request $r$ at year $t$. |
| $c \in C_r$ | The set of LP configurations. |
| $l_c$ | An LP with configuration $c$. |
| $R_c$ | The data rate of LP configuration $c$ |
| $Q_c$ | The modulation format of LP configuration $c$ |
| $B_c$ | The channel bandwidth of LP configuration $c$ |
| $SNR_c$ | The minimum receiver OSNR of LP configuration $c$ |
| $KSP_r$ | The set of k-shortest-paths for request $r$ |
| $\delta$ | The maximum over-provisioning bound |
| Decision Variables | |
| $x_{c,p} \geq 0$ | Number of deployed LPs with configuration $c$ on path $p$ |

method, for every source-destination request pair routed on any of the k-shortest paths in the network, HeCSON provides a list of feasible BVT configurations, taking into account the OSNR as well as the minimum receiver sensitivity OSNR. Thereafter, it uses the first-fit approach to place the LPs into the spectrum. Also, it ensures that the spectrum contiguity is fulfilled. For clarification, let us consider the example presented in Fig. 3.3. In this example, a network has six nodes and seven links, with three requests with the same requirements at the starting year ($t = 0$). The spectrum of each link and the assignment of each request LP is presented on the right side of the figure. The source and destination nodes for *red*, *green*, and *blue* requests are (A,E), (A,C), and (A,F), respectively. Let us assume that the determined LP configuration for all the requests is the same, each 1/4-th of the spectrum space. Since our approach is in request-by-request manner, we assume that we start first with the *red* request, followed by *green* and *blue* ones. The routing decisions here are also assumed to be the shortest path. The *red* request is routed through $A \rightarrow D \rightarrow E$ path, with a first-fit channel

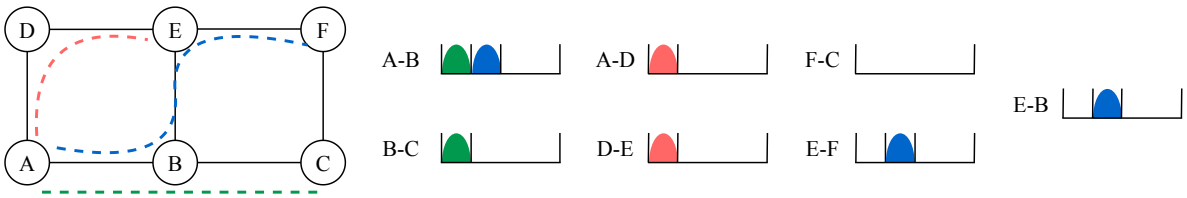



**Figure 3.3:** An example of a network with three requests (assumed with equal requirements here), and their placement in the network.

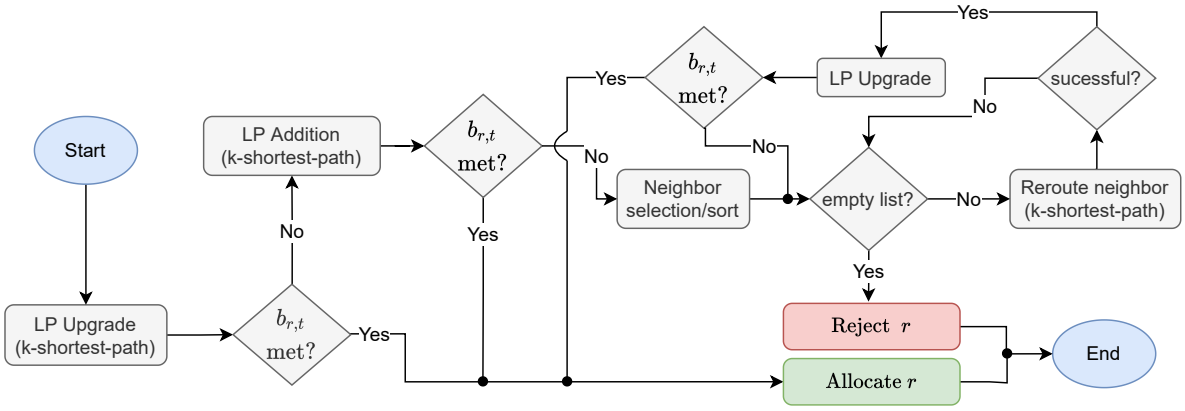



**Figure 3.4:** The flowchart of the proposed RCSA algorithm. It shows how the network reconfiguration methods work together to provision for the increased data rate.

allocation over the links $A-D$ and $D-E$, as appears in the figure. Similarly, the *green* and *blue* requests are also provisioned in the network. Note that the spectrum contiguity constraint is also met here. Also, we note that in this network, each request can have many LPs, each deployed on different paths, and with different configurations.

From the first year onward ($t \geq 1$), having the increased requests traffic in hand, the algorithm is triggered yearly, in a request-by-request manner. For each request, the algorithm is triggered if the sum of the data rate of the deployed LP(s) is lower than the total increased traffic:

$$\sum_{l_c \in L_{r,t-1}} l_c \times R_c \leq b_{r,t}, \forall t \in T, \forall r \in \mathcal{R}^t (t > 0). \tag{3.4}$$

In this case, to cope with the increased data rate, the proposed RCSA algorithm performs a three-step approach to cope with this traffic, with low over-provisioning and power-efficient manner. A high level flowchart of the proposed RCSA algorithm is depicted in Fig. 3.4: In particular, it combines three reconfiguration methods, to achieve the goal, namely *LP Upgrade* [173], *LP Addition* [172], and the proposed *LP Reroute*. In below, we explain the algorithm, starting by presenting the reconfiguration methods in detail.

### 3.4.1 Reconfiguration Method 1: *LP Upgrade*

The algorithm first tries to upgrade each LP in $L_{r,t-1}$ to a configuration (from the possible configuration set) with a higher data rate [173]. Considering the spectrum occupancy and the OSNR along each k-shortest-path of $r_{i,j,t}$ (where the LPs could have been deployed), the *LP Upgrade* method checks if the deployed LP(s) can be upgraded to a new channel configuration with higher data rate. If possible, one or more LPs of the current request are upgraded until the increased requested data rate $b_{r,t}$ is met. If neither enough frequency slots are available, nor the required OSNR can be guaranteed, the algorithm moves to the second reconfiguration method, *LP Addition*. Consider the example depicted in Fig. 3.5, which is based on the previous Fig. 3.3. In this example, the traffic of the *red* and *green* requests are increased at the current $t$. To cope with this, the deployed LP of the *red* request on path $A-D-E$ is upgraded to a configuration with a higher data rate, which consumes two times more channel space than the previous configuration (see Fig. 3.3). However, upgrading the LP of the *green*

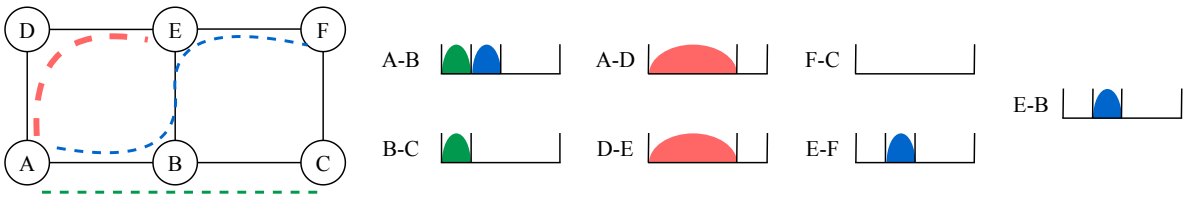



**Figure 3.5:** The *LP Upgrade* reconfiguration method for an exemplary scenario.

request is not possible in this case since there is no free space in the channel. In this case, the second reconfiguration method comes handy, which is introduced in the next part.

### 3.4.2 Reconfiguration Method 2: *LP Addition*

At this stage, when the current LPs of a request cannot be upgraded (e.g., due to lack of spectrum availability), the algorithm deploys one or more LPs on one of the available paths from $KSP_r$, to support the increased data rate [172]. Previously, in *LP Upgrade* example in Fig. 3.5, we showed that the *green* request cannot be upgraded due to the insufficient channel space. As it is shown in Fig. 3.6, a new LP for the *green* request can be deployed on the same path. Of course, in addition to the data rate, the OSNR requirements must be met.

To decide the configuration and the path of the new LPs, we formulate this reconfiguration problem as an ILP optimization model. To develop this method, we use an integer decision variable $x_{c,p} \geq 0$, where its value determines the number of LPs with configuration $c$ which should be deployed on path $p$.

Further, this ILP model must satisfy a set of constraints, which is described below. Firstly, the sum of the data rate of the newly deployed LPs must be greater than the amount of request increased data rate. This can be written as:

$$\sum_{c \in C_r} \sum_{p \in KSP_r} x_{c,p} \times R_c \geq \Delta_{r,t}. \tag{3.5}$$

Secondly, the allocated data rate should be bounded by a value of $\delta$. We use $\delta$ to balance the over-provisioning and the need for future LP deployments. More specifically, bigger $\delta$ means more over-provisioning and possibly fewer reconfigurations in the future. This set of constraints can be formulated as:

$$\sum_{c \in C_r} \sum_{p \in KSP_r} x_{c,p} \times R_c \leq \Delta_{r,t} + \delta. \tag{3.6}$$

However, due to many possible parameters (e.g., over-provisioning, power-consumption), determining the objective function for this optimization model does not seem straightforward. Therefore, we consider four objective functions based on the data rate and the number of deployed LPs in the network. In the following, we present two single and two multi-objective cases, which we can use to find a solution for the *LP Addition* reconfiguration method.

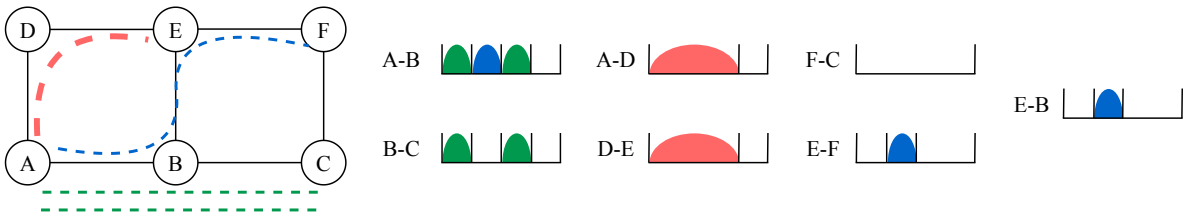



**Figure 3.6:** The *LP Addition* reconfiguration method for an exemplary scenario.

#### 3.4.2.1 Objective 1: Minimize LP (*MinLP*)

The first objective is to minimize the number of newly deployed LP for the request[2]. This objective might be able to reduce the over-provisioning over time. Also, it can help to reduce the power consumption of the network incurred by deploying new LPs. In this case, the optimization formulation can be presented as:

$$\text{Minimize} \sum_{c \in C_r} \sum_{p \in KSP_r} x_{c,p}, \tag{3.7}$$

$$\text{s.t. Constraints } (3.5), (3.6),$$

$$\text{vars. } x_{c,p} \geq 0, \forall c \in C_r, \forall p \in KSP_r.$$

#### 3.4.2.2 Objective 2: Maximize Data Rate (*MaxDR*)

The second model has the objective function of maximizing the configured data rates[3]. The idea behind this formulation is to deploy LPs with large enough data rates so that the need of deploying new LP in subsequent planning years is decreased. This model can be formalized as:

$$\text{Maximize} \sum_{c \in C_r} \sum_{p \in KSP_r} \left(x_{c,p} \times R_c\right), \tag{3.8}$$

$$\text{s.t. Constraints } (3.5), (3.6),$$

$$\text{vars. } x_{c,p} \geq 0, \forall c \in C_r, \forall p \in KSP_r.$$

#### 3.4.2.3 Objective 3: Maximize Data Rate, Minimize LP (*MaxDR, MinLP*)

This formulation is developed as a multi-objective case with hierarchical objectives, i.e., optimizing the first objective and then the second while keeping the first one as a constraint. In this case, the first objective (with higher priority) is to maximize the data rate, while the second one (with lower

[2]For simplicity, we use the term *Min LP* to refer to this model.
[3]For simplicity, we use the term *Max DR* to refer to this model.

priority) is to minimize the number of newly deployed LPs[4].

$$\begin{aligned} &\text{Maximize} \sum_{c \in C_r} \sum_{p \in KSP_r} \left(x_{c,p} \times R_c\right), \\ &\text{Minimize} \sum_{c \in C_r} \sum_{p \in KSP_r} x_{c,p}, \\ &\text{s.t. Constraints (3.5), (3.6),} \\ &\text{vars. } x_{c,p} \geq 0, \forall c \in C_r, \forall p \in KSP_r. \end{aligned} \tag{3.9}$$

The goal behind this formulation is to prevent future need for new LPs, while using the minimum number of LPs in the current year.

#### 3.4.2.4 Objective 4: Minimize LP, Maximize Data Rate (*MinLP, MaxDR*)

The last formulation is with a multi-objective model, which as the first objective is to deploy a minimum number of new LPs, and maximize the data rate[5]. This model can be represented as:

$$\begin{aligned} &\text{Minimize} \sum_{c \in C_r} \sum_{p \in KSP_r} x_{c,p}, \\ &\text{Maximize} \sum_{c \in C_r} \sum_{p \in KSP_r} \left(x_{c,p} \times R_c\right), \\ &\text{s.t. Constraints (3.5), (3.6),} \\ &\text{vars. } x_{c,p} \geq 0, \forall c \in C_r, \forall p \in KSP_r. \end{aligned} \tag{3.10}$$

In Section 3.5, we extensively compare the performance of these four different objective functions.

Having the optimization model, we can determine the required number of new LPs to be deployed, their configurations, and respective paths. Before deploying the LPs, we check if the new changes in the network are still compliant with the OSNR requirements of the requests. Thus, if either the OSNR cannot be met, the algorithm considers using the third reconfiguration method, the *LP Reroute*.

### 3.4.3 Reconfiguration Method 3: *LP Reroute*

In this method, the algorithm tries to cope with the requested data rate by rerouting the neighboring LPs of the request in hand. Thus, the released spectrum space can be used to upgrade the existing LPs of the request in hand to a higher data rate (i.e., performing *LP Upgrade*). The first step is to select the currently deployed LPs of the request in hand. Generally, the more LPs are rerouted (removed) from the links with high channel utilization, the more likely it is to release the resources required for LPs of the request in hand to be upgraded. Thus, for each of these LPs, their neighboring LPs are listed and decreasingly ordered based on the channel utilization of the links that they are using. After that, we reroute the neighboring LPs one by one to a k-shortest-path between source and destination of the request in hand, considering the spectrum occupancy and minimum OSNR requirements. After each

[4]For simplicity, we use the term *Max DR, Min LP* to refer to this model.
[5]For simplicity, we use the term *Min LP, Max DR* to refer to this model.

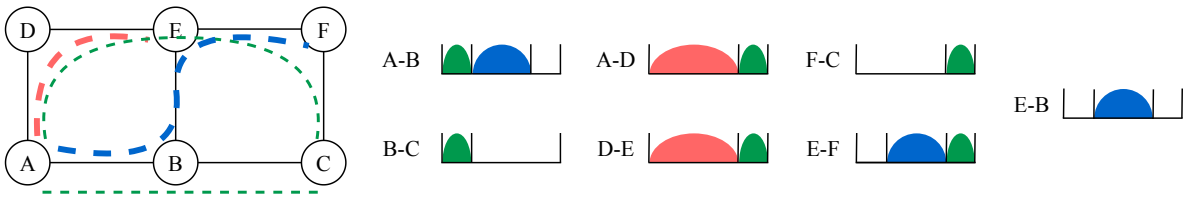



**Figure 3.7:** The *LP Reroute* reconfiguration method for an exemplary scenario.

successful reroute, we try to upgrade the LP of request in hand (whose neighbor has been moved) to a higher data rate. If the upgrade is successful and leads to meeting the requested traffic, the algorithm allocates the resources to the request in hand and stops. Otherwise, if the rerouting of all the neighboring LPs cannot satisfy the increased data rate, the request of the request in hand is blocked. Note that the rerouting method does not have a significant effect on the power consumption since it only enables the possible upgrade of existing LPs to a higher data rate (i.e., an increase in $P_d$).

A high level example of the *LP Reroute* method is presented in Fig. 3.7. In Fig. 3.5 and 3.6, we showed how the *LP Upgrade* and *LP Addition* methods are used to cope with the traffic increase of *red* and *green* requests. Having the *blue* request in hand, the algorithm starts with *LP Upgrade*. However, it does not help, because there is no available free spectrum around the *blue* request in link $A-B$ (see Fig. 3.6). Also, *LP Addition* does not work, assuming adding a new LP will violate the OSNR of some existing LPs. As it is shown in Fig. 3.7, *LP Reroute* can lead to successfully provisioning the *blue* request. To do this, it first reroutes the second LP of the *green* request (right neighbor of the *blue* LP on link $A-B$) from $A \rightarrow B \rightarrow C$ path to $A \rightarrow D \rightarrow E \rightarrow F \rightarrow C$. By doing this, the right side of the spectrum is empty on all the links of the *blue* LP, Thus, it is possible to upgrade the *blue* LP to a higher data rate. This example shows the benefit of the *LP Reroute* reconfiguration method.

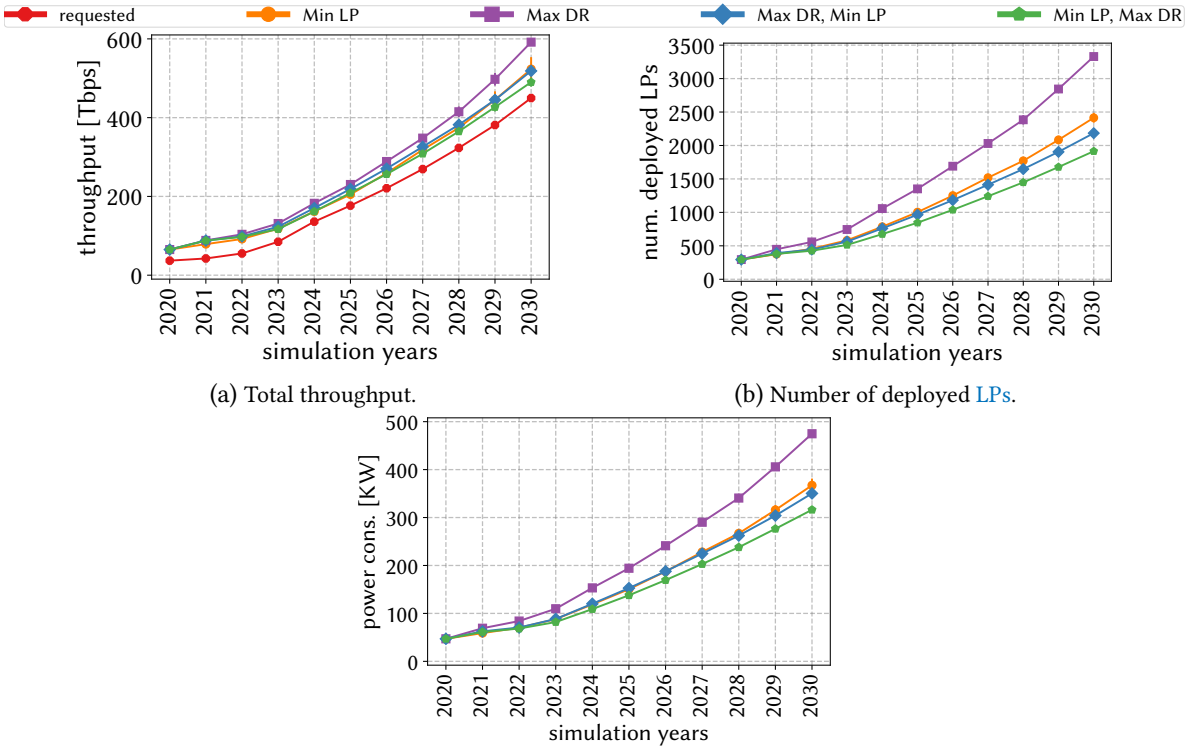
## 3.5 Performance Evaluation

In this section, we evaluate the performance of the introduced RCSA algorithm in different settings. We start by presenting the simulation setup. Thereafter, we discuss the results of the performance evaluation in different metrics.

### 3.5.1 Simulation Setup

We evaluate our planning algorithm with different objective functions for $T = 10$ years. We consider two topologies, Nobel-EU ($|N| = 28$, $|L| = 41$, and $|D| = 378$) and Nobel-Germany ($|N| = 17$, $|L| = 26$, and $|D| = 136$) [168]. Each link consists of a single fiber pair, and the transponders are equipped to handle 26 different configurations for the C-Band [173]. The data rate request model for the first year is taken from our previous work [172]. In addition to different numbers of links, nodes, and requests, the volume of yearly traffic in Nobel-EU is higher than Nobel-Germany, up to 3x [172].

To include the uncertainty of the traffic increase (and generate random traffic requests), we extend [172] by considering a ±15% of random deviation in traffic request onward from the second planning year. The paths for routing the requests can be chosen from the set of k-shortest-paths with $k = 3$. Also, the static and dynamic power consumption values are taken from [223]: $P_s = 120$,




(a) Total throughput.

(b) Number of deployed LPs.

(c) Power consumption.

**Figure 3.8:** Comparison of different objective function of the RCSA algorithm for Nobel-EU topology.

and $P_d = 0.18$ Watts per Gbps. Finally, the value of $\delta$ is considered as 150 Gbps. This value is the minimum possible data rate to be over-provisioned, such that the model can deploy at least one LP (i.e., to make the ILP formulation feasible to be solved).

The planning tool has been implemented in Java, and the optimization has been solved using Gurobi [113]. The evaluations have been performed on a machine equipped with Intel Core i7-6700HQ @2.60 GHz, 16 GB of RAM, running Ubuntu 18.04. Finally, to generate reliable results, each scenario is run for 100 random traffic request cases.

### 3.5.2 Simulations Results

In this part, we explain and compare the evaluation results per topology in terms of total throughput, the total number of deployed LPs, and power efficiency. Afterward, we compare the performance of the RCSA algorithm for different reconfiguration options.

#### 3.5.2.1 Comparing Objective Functions

Let us start with the achieved network throughput in the Nobel-EU topology. As it can be seen in Fig. 3.8a, the *Max DR* approach results in the maximum over-provisioning in the network, while *Min LP, Max DR* achieves the minimum. Similar behavior can be seen for the number of deployed LPs in the network (Fig. 3.8b). The reason for this behavior is *Max DR* greedily increases the data rate, with no constraints on the number of deployed LPs. Meanwhile, in *Min LP, Max DR*, a lower number of LPs is provisioned with a higher data rate configuration.

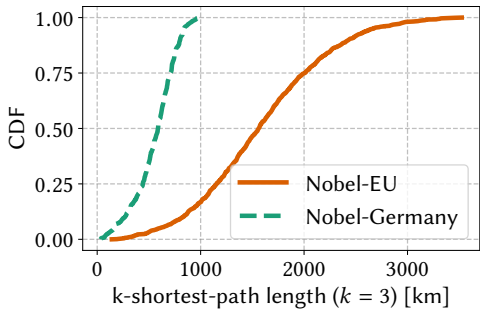
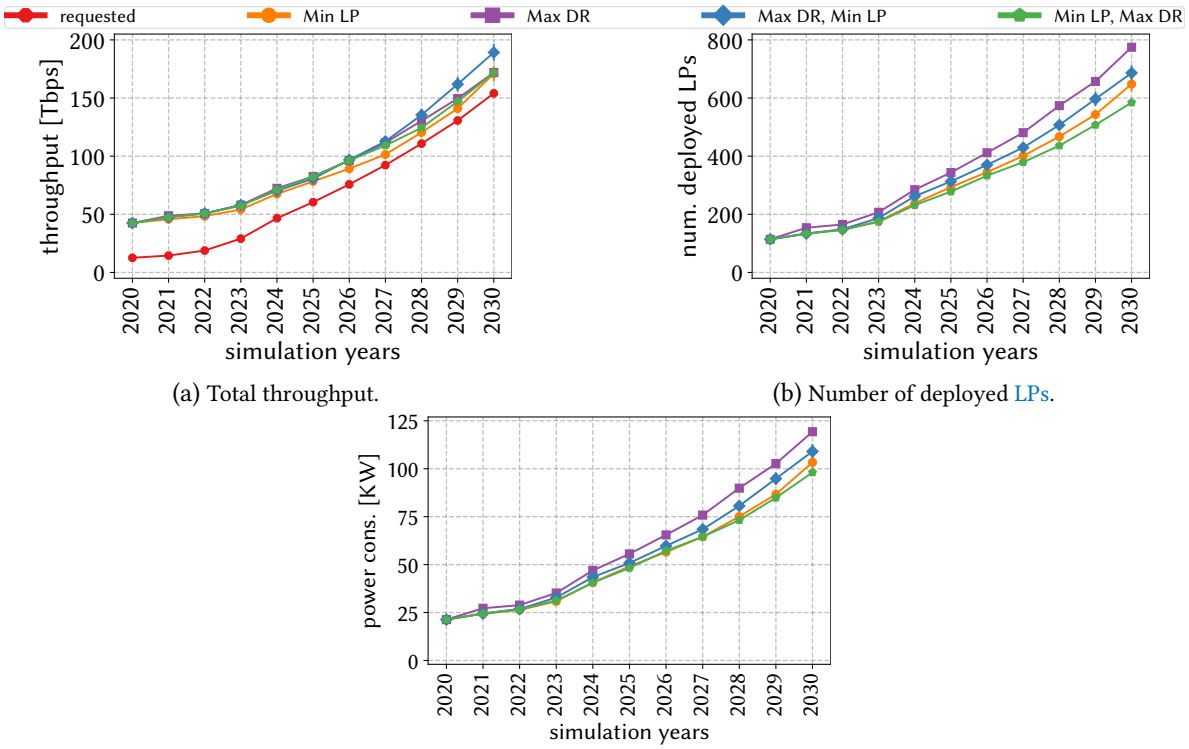



**Figure 3.9:** The comparison of the length of the requests k-shortest-paths ($k = 3$) for Nobel-EU and Nobel-Germany topologies.

Surprisingly, Fig. 3.8b shows that *Min LP* results in deploying more LPs than *Max DR, Min LP*. The main reason is, as shown in Fig. 3.9, the path lengths in the Nobel-EU are up to 3x longer compared to the Nobel-Germany topology. This can prevent LPs from using high data rate configurations due to a decrease in the OSNR in longer distances. Applying the same reasoning, *Max DR* meets the requested traffic by deploying a much low data rate LPs.

Moreover, by deploying a low number of high data rate LPs, *Min LP, Max DR* can support the future requested traffic with a lower number of LPs as compared to the other approaches. Further, Fig. 3.8c shows that the power consumption of the network follows the same trend as the number of deployed LPs. It shows that the static power due to the LP deployment is the dominant factor in the power consumption of the network.

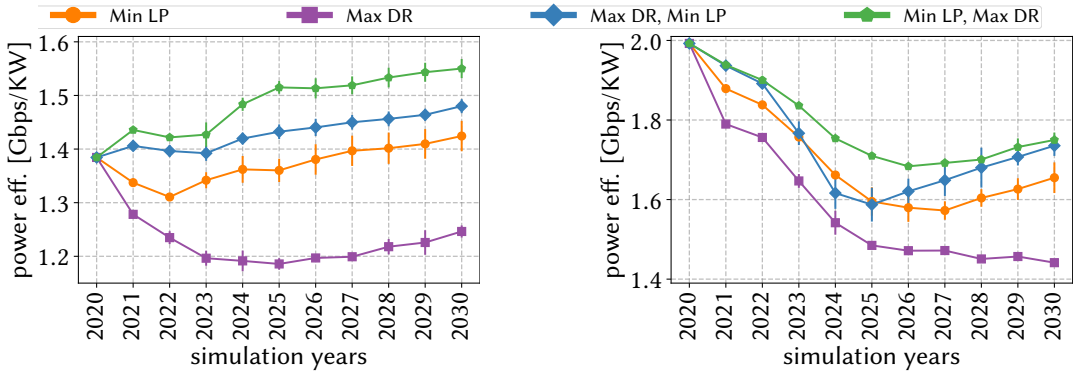



(a) Total throughput.

(b) Number of deployed LPs.

(c) Power consumption.

**Figure 3.10:** Comparison of different objective function of the RCSA algorithm for Nobel-Germany topology.




(a) Nobel-EU network

(b) Nobel-Germany network

**Figure 3.11:** Comparison of power-efficiency.

Moving to the Nobel-Germany topology, Fig. 3.10 shows that the behavior of different objective functions is similar to the Nobel-EU; however, with some differences. These differences are due to the characteristics of the topologies, specifically in terms of path length (see Fig. 3.9). That is, the *Max DR, Min LP* objective leads to the maximum throughput, while in the Nobel-EU topology, *Max DR* results in the maximum. In other words, since the average path length is lower in Nobel-Germany than Nobel-EU, *Max DR* can provision more throughput compared to the other three objective functions. This is because the capability of deploying LPs with a higher data rate is higher in Nobel-Germany. Another difference is shown in Fig. 3.10a, where it can be observed that *Min LP, Max DR* over-provisions the network more than *Min LP* while deploying less LPs (see Fig. 3.10b). Further, as opposed to Nobel-EU, the different path length leads to *Min LP* deploying a lower number of LPs than *Max DR, Min LP*.

Finally, we present the results for the power efficiency, which is defined as the consumed power per Gbps of traffic. Interestingly, by combining the throughput and power consumption, we can see similar behavior in both topologies in terms of power efficiency (see Fig. 3.11). In particular, Fig. 3.11 shows that the single-objective approaches perform poorly compared to multi-objective approaches. For example, it can be seen that during the planning years, deploying with *Min LP* becomes insufficient over time, leading to deploying more LPs in later years (i.e., lower power-efficiency). However, when considering the multi-objective approaches (considering both the static and dynamic power consumption), the power efficiency significantly improves. That is, the *Min LP, Max DR* approach achieves the best power efficiency compared to the other approaches.

#### 3.5.2.2 Comparing Reconfiguration Methods

To evaluate the reconfiguration methods, we consider three approaches which combines different reconfiguration methods:

1. *Approach 1*: It includes only the *LP Upgrade* method [173].
2. *Approach 2*: It considers the *LP Upgrade* and *LP Addition* methods [172].
3. *Approach 3*: This approach combines the three *LP Upgrade*, *LP Addition*, and *LP Reroute* reconfiguration methods [17].

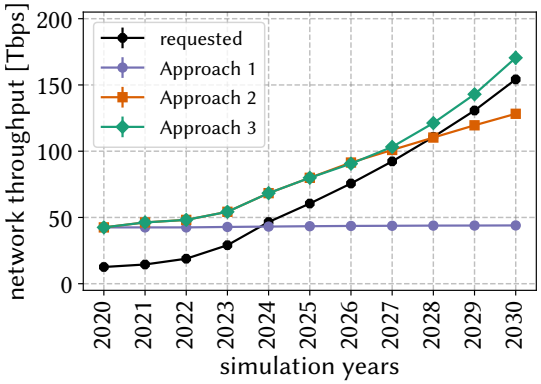



**Figure 3.12:** Network throughput.

We use these approaches to evaluate the impact of each reconfiguration method on the achieved performance. Further, we only consider the Nobel-Germany topology and *Min LP* objective function for the comparison purposes in this part.

We start by comparing the total provisioned throughput by each approach against the requested data rate. According to Fig. 3.12, it can be seen that the *Approach 1* method cannot cope with the requested traffic after 2024, leading to network under-provisioning. The reason is that mostly, the LPs cannot be upgraded to a higher data rate due to violation of minimum OSNR requirement on longer paths.

Further, *Approach 2* increases the throughput of the network as required up to 2028. In this approach, when the LP data rate cannot be upgraded, the set up of new LPs allows offering the required traffic. The reason is that in addition to upgrading the LP data rates, the algorithm can add more LPs if necessary. Also, in Fig. 3.13, it can be seen that *LP Addition*, can actually lead to having also more LP upgrades, both resulting in an increase of the total throughput.

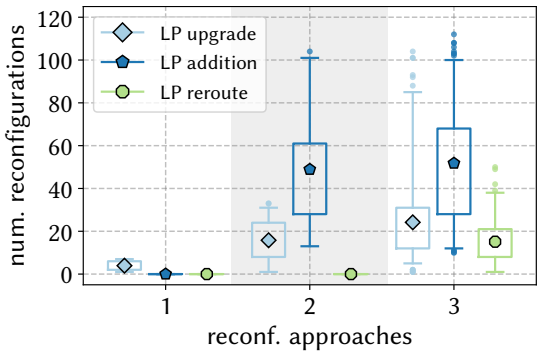



**Figure 3.13:** Number of reconfigurations.

Nevertheless, this combination fails after 2028, since the channels are greedily occupied, which prevents adding/upgrading more LPs in the future. Ultimately, by utilizing the *Approach 3*, it can be observed that the network throughput is increased up to 33% compared to *Approach 2* (52% and 59% in case of Abilene-USA and Germany50 networks, respectively). In more detail, as Fig. 3.13 shows, by rerouting a few LPs, more LPs can be upgraded and provisioned, leading to higher throughput. Interestingly, Fig. 3.14 shows that by benefiting from rerouting, *Approach 3* can cope with traffic

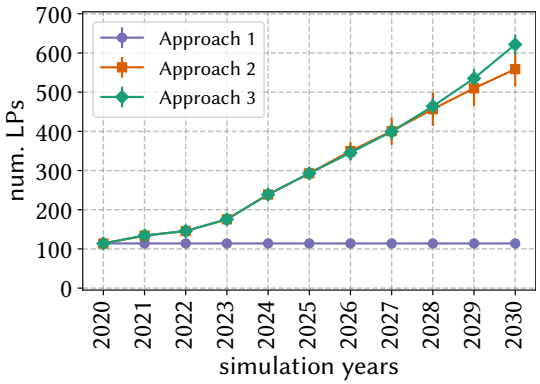



**Figure 3.14:** Number of deployed LPs.

requests by only deploying 3% more LPs compared to *Approach 2*, while using the same physical infrastructure. This fact indicates that *Approach 2* and 3 result in comparable OPEX. For example, the number of active transponders that are a power-hungry resource in the network [223] is only increased by 3% to satisfy the incoming traffic.

Further, Fig. 3.15 shows the reconfigurability of LPs in the network, based on their data rate. It can be seen that generally, the LP configurations with lower data rates are easier to reconfigure. Although *LP Upgrade* seems to be the cheapest reconfiguration method, however, to offer more throughput, the rerouting method shows promise.

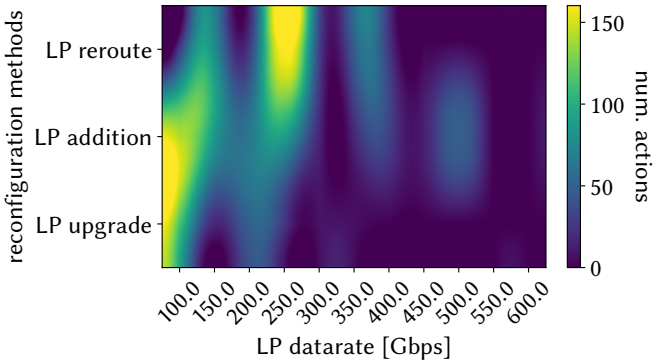



**Figure 3.15:** The distribution of number of reconfigurations of all LPs deployed in Nobel-Germany network based on their data rate.

Finally, we note that the runtime of our heuristic is in order of minutes for a 10-year scenario, which shows that our heuristic can be utilized in online planning tools.

## 3.6 Summary

In this chapter, we have studied a scenario where the network reconfiguration is triggered due to traffic changes in the network. In more detail, we focused on the dynamic network reconfiguration problem in a multi-period resource planning scenario with a traffic increase. We introduced an RCSA algorithm to answer these three main questions: *when*, *what* to reconfigure, and *how* to reconfigure the network to efficiently cope with the increased traffic? To achieve it, we have considered three types of reconfiguration: upgrading, adding, and/or rerouting the requests. Our proposed RCSA

algorithm benefits from all the three reconfiguration methods, leading to a performant solution. We formulated and compared four different variations of our proposed RCSA algorithm, mainly in terms of network throughput, deployed LPs, and power-efficiency. It has been shown that the multi-objective approach, *Min LP, Max DR*, performs the best by achieving the highest power efficiency while reducing the number of required LPs and lower over-provisioned throughput.

Further, the simulation results indicated that combining the three reconfiguration methods can significantly increase the network throughput, using almost the same network resources (e.g., number of deployed LPs). It can lead to substantial cost savings for network providers, deploying their networks using modern flex-grid networks. Additionally, our results showed that the rerouting method performs better on LPs with a low data rate. Therefore, it is evident that greedily upgrading LPs to high data rates can reduce the benefits of this method. Hence, opposed to intuition, the *LP Upgrade* method is not as efficient since it can lead to a lower number of rerouting possibilities and lower overall throughput.

In the next chapter, we study another scenario where users' mobility triggers network reconfigurations instead of traffic changes. Specifically, having a set of mobile users, to keep an acceptable level of Quality of Service (QoS) (e.g., end-to-end delay), the resource allocation decisions need to be adapted. However, these resources and reconfigurations are costly. Thus, we will provide cost-efficient resource allocation and reconfiguration solutions in a scenario with mobile users.

# Chapter 4

# Reconfiguration Due to Mobility: Cost Optimization

## 4.1 Introduction

In the previous chapter, we have introduced a framework to determine the necessary network reconfigurations due to the traffic changes in the network. In addition to traffic changes, the mobility of users can have an impact on the network performance metrics, such as end-to-end delay. In this chapter, we study a resource allocation and reconfiguration problem, where the resources should be allocated to mobile users in a cost-efficient manner. Also, the time and type of necessary network reconfigurations should be determined in order to guarantee a specific service delay for the mobile users. This chapter is based on the presented papers in [14, 15, 19].

With the recent advances in networking technologies, such as 5G/6G, the popularity and development of dynamic and real-time services such as augmented reality and video games are increasing. The deployment of these services is receiving attention, especially in use cases such as smart transportation, autonomous driving, and Internet of Things (IoT). Traditionally, these services were hosted by a centralized Data Centers (DCs), usually as a Virtual Machine (VM). However, with strict requirements of these services, e.g., end-to-end delay, DCs may not be satisfactory anymore. As a remedy, an emerging paradigm called Mobile Edge Computing (MEC) has been proposed where the services are deployed in the proximity of the users. In MEC, instead of a large centralized DC, there are a number of small DCs which are distributed in the service area, e.g., a city. Then, user services are deployed on these distributed DCs and can be used in a shared manner between users of the same area, which can reduce the connection delay between the user and the VM, i.e., end-to-end delay improvement. However, the capacity of these DCs is relatively limited compared to the centralized one. As a deployment strategy, one can deploy all service types in all the DCs (assuming the capacity of DCs are enough). It can make the services accessible to all the users with the minimum delay. However, it comes at great deployment costs, i.e., licensing costs, power consumption, and maintenance. Therefore, it is important yet challenging to determine which service VMs should be deployed on which DCs to optimize the delay and the cost of deployment.

To be able to access the services, users need to use the network links on the route towards the VM hosted in a DC. There are network access points that can be used by the users to be connected

to the core network where the DCs are located. Multiple technologies can be used as network access points. For example, in the case of a MEC solution in a city, the 5G eNodeB towers can be used to connect to the core network. In addition, with the recent advances in space networking, satellites are another candidate to be used as access points, which can offer a high link capacity with low delay [151]. An example is Low-Earth Orbit (LEO) satellites that are being used in many scenarios such as IoT and Internet of Vehicles (IoV) [180, 151]. Nevertheless, these access points solutions have different characteristics which need to be considered, such as maximum link capacity, delay, and cost. For example, satellite networks are usually more expensive, more delay-induced, but they provide global coverage for users. Therefore, in addition to the VM placement problem, routing decisions from the users to the VM instances are yet another crucial problem that needs to be addressed.

However, throughout time, the location of users can be changed due to their mobility. Therefore, the previous VM placement and routing decisions may not be *good* anymore. For example, in the case of autonomous driving, the distance of the car with its respective VM can increase over time, and hence the service delay. To maintain the service requirements (e.g., delay) during the time, the placement of VM should be changed according to the mobility of users. VMs can be migrated between DCs using live migration techniques [227, 211], which are widely available in current virtualization platforms, such as Linux KVM, and Microsoft HyperV. Although VM migration is helpful to improve the delay between user and service endpoint, it comes at a cost, e.g., extra bandwidth usage and service downtime. Thus, in use cases with mobile users, in addition to VM placement and routing problems, the third challenge is to decide when, which, and where to migrate a VM.

### 4.1.1 Key Contributions

This chapter focuses on a scenario with mobile users demanding some services with different requirements, running in a system modeled over a discrete-time horizon. We provide a generic system model and problem formulation to answer the following questions:

1. VM placement: Which VM type and how many of them should be hosted by which DC?
2. User assignment: Which user should use which VM at which DC?
3. Routing: Through which network path should the users access the VMs?
4. VM migration: When, which, and where to migrate a VM?
5. Objective: How to jointly answer these questions to have a minimum deployment (i.e., number of deployed VM instances, and used links) and migration costs while meeting the service requirements?

To answer these questions, we formulate the above problem with two Integer Linear Program (ILP) formulations. The first one is based on static settings, where the decision is made at each specific timeslot. As the second model, we extend the static to the mobility-aware (i.e., dynamic) scenario where the decisions depend on the mobility of users and previous decisions. After that, we show that this problem is NP-Complete, and hence large problem instances cannot be solved promptly. To cope with this limitation, we introduce two heuristic algorithms to solve the problem efficiently.

Finally, we introduce the scenario in which we evaluate the performance of the proposed models and algorithms.

### 4.1.2 Organization

The rest of this chapter is organized as follows: we first present the related work in Section 4.2. Thereafter, we formulate the problem and the two ILP formulations in Section 4.3. Then, we present fast and efficient heuristics in Section 4.6, followed by the evaluation of them in Section 4.7. Finally, before presenting the chapter summary in Section 4.9, we present a Graphical User Interface (GUI) which clarifies the problem and the solutions with a few graphical examples in Section 4.8.

## 4.2 Related Work

We see our work partially embedded in area of mobility-aware resource management for use-cases such as IoV [27, 199, 35, 97, 215, 217] or MEC [153, 102, 200]. On the one hand, in the area of IoV, authors in [27] present an Software-Defined Networking (SDN)-based platform to guarantee the delay requirements for 5G-enabled automotive systems. In their approach, while the vehicles are moving, their connection delay is being continuously monitored. In case of losing certain delay levels, the VM migration is triggered for delay improvement. However, they have not considered other Quality of Service (QoS) parameters such as network bandwidth capacity. Moreover, Yao et. al. [226] assign a VM per vehicle to deliver the required services, such as in-vehicle multimedia entertainment and vehicular social networking. By minimizing the network costs, they determine the necessity and the place to migrate the VMs during vehicle movement. However, they do not determine the routing from vehicles to their respective VM instances. Also, in contrast to us, they solve the problem for the next timeslot instead of the whole time horizon.

On the other hand, in the MEC area, authors in [199, 97] have proposed an approach to dynamically place network functions on MEC servers available in mobile base stations, according to the handover probability of users for the next timeslot. They formulate two optimization models with the objective of minimizing network function migrations and communication cost (i.e., QoS/Quality of Experience (QoE)) between User Equipments (UEs) and network functions. In another work, Bahreini et. al. [35] formulate an optimization model to map application graphs to network graphs according to the user's location. Their objective is to minimize the total operational cost of running and migrating applications and communication costs between users and applications. Finally, a dynamic service placement approach is proposed by Wang et. al. [217]. They minimize the placement, routing, and migration costs based on a specific look-ahead window. However, they do not determine the routing between users and instances. Also, they do not guarantee the end-to-end delay of the delivered services. A comparison of our approach with others is summarized in Table 4.1. Notably, *shared instance* column in Table 4.1 indicates whether a VM is shared among several users at the same time.

This chapter formulates and evaluates the Joint Service Placement, Assignment, Routing, and Migration (JSPARM) problem by minimizing the total operational cost of VM deployment, routing, and migration decisions. Further, it provides a VM migration control to adapt service placement and

**Table 4.1:** Comparison of the related work with our approach, all considering the user mobility.

| Works | Routing | Placement | VM migration | Delay | Shared Instance |
|---|---|---|---|---|---|
| [97, 226] | ✗ | ✓ | ✓ | ✗ | ✗ |
| [199, 215, 27] | ✗ | ✓ | ✓ | ✓ | ✗ |
| [35, 217] | ✗ | ✓ | ✓ | ✗ | ✗ |
| **Our work** | ✓ | ✓ | ✓ | ✓ | ✓ |

routing according to the mobility of users. Finally, our proposed method guarantees the bandwidth and the end-to-end delay parameters, required by the users.

## 4.3 System Model and Problem Formulation

This chapter tackles the problem of JSPARM for mobile users, e.g., cars, airplanes by minimizing the total operational cost. We formulated the JSPARM problem in two cases:

1. Static Joint Service Placement, Assignment, Routing, and Migration (S-JSPARM): it solves the JSPARM for a particular timeslot (i.e., based on the position of users, and network status).
2. Mobility-Aware Joint Service Placement, Assignment, Routing, and Migration (MA-JSPARM) (i.e., dynamic): This approach takes future timeslots into account and hence, the user's mobility to solve the JSPARM problem. We also introduce and model a VM migration control approach to improve the solution in terms of total operational cost.

Before formulating the two JSPARM optimizations, we present the system model, followed by the problem definition. After that, we present the cost models for this problem, and then we begin with the formulation of S-JSPARM and MA-JSPARM optimization problems.

### 4.3.1 System Network

Let us define a finite time horizon comprising $T$ timeslots, each with a duration of $\omega$ (i.e., in minutes). The system network is defined as a bidirectional connected graph $G = (N, L)$, where $N$ and $L$ are the network's set of nodes and links, respectively. In $G$, each link $(u, v) \in L$ between nodes $u, v \in N$ is characterized by a constant cost value $Cost^L_{u,v}$, propagation delay $D_{u,v}$, and capacity $B_{u,v}$. This graph consists of three segments: users, core network, and access points, which are defined below.

### 4.3.2 Users

We denote $F$ as the set of mobile users considered in the problem. Each user $f \in F$ has a different mobility path, and thus, it is characterized by the different locations (in terms of Latitude (Lat) and Longitude (Lon)) and the duration of the activity. We define $\mathcal{T}_f$ as the activity duration (number of timeslots) of user $f \in F$ in the system. Without loss of generality, we assume that users' location is known as a priori. This assumption can be realistic in many use cases, such as users, trains, and ships, that their mobility trajectory is known beforehand. With this assumption, the users locations at any timeslot are considered as nodes in graph $G$, which are denoted as $N_A$ where $\mid N_A \mid = \sum_{i=1}^{\mid F \mid} \mid f_i \mid$.

### 4.3.3 Core Network

In our system model, there is a core network with nodes and links, denoted as $\mathcal{N}_C$ and $\mathcal{L}_C$, respectively. Each node is a network forwarding device that can be programmed, e.g., using OpenFlow [160]. Some of the nodes in the core networks are considered to be DCs, which is denoted as $\mathcal{N}_{DC} \subset \mathcal{N}_C$. These DCs can host VM instances that provide services to the users, with a limited resource capacity of $C_j^{DC}$ (e.g., in terms of CPU cores). Each VM can provide a specific service type $k \in K$, with a resource requirement of $Size_k^{VM}$ (e.g., number of CPU cores). Also, the cost of deploying an instance of VM with service type $k$ is denoted as $Cost_k^{VM}$. This cost can also be dependent on the flavor of the VM.

### 4.3.4 Access Points

Mobile users can communicate to the core network (and hence, DCs) through access points. For example, 4G/5G base stations and satellites are currently the two most popular access point platforms in many usecases [229, 233]. Different access point mediums can have different costs, capacities, and propagation delays. Since our model is intended to be generic, we define $\mathcal{N}_{AP}$ as the set of access point nodes in the network. These access point nodes can be distributed over the network. Each node in $\mathcal{N}_{AP}$ is connected to the closest core network node $\mathcal{N}_C$. Users can connect to an access point node (usually to the closest one). The link between user and access point nodes is denoted as $\mathcal{L}_{AP}$.

Having the users, core network, and access point defined, in summary, considering the network $G$, we have $N = \mathcal{N}_C \cup \mathcal{N}_{AP} \cup N_A$, and $L = \mathcal{L}_C \cup \mathcal{L}_{AP}$.

### 4.3.5 User Requests

The service requests triggered by a user is denoted as $r \in \mathcal{R}$ and is characterized by following 4-tuple:

$$r = (src_r, k_r, \mathcal{B}_r, \mathcal{D}_r),$$

where $src_r \in N_A$ is the user node, $k_r \in K$ is the service type, $\mathcal{B}_r$ is the required bandwidth, and $\mathcal{D}_r$ is the maximum allowed end-to-end delay of the service request $r$. These requests are served by an instance of the respective service, at a specific DC.

In the following, we present the different cost models that can be considered for this problem. Note that the key notations and their definition are summarized in Table 4.2.

**Table 4.2:** Notation definition for the JSPARM problem.

| Parameter | Definition |
|---|---|
| $T$ | Set of timeslots. |
| $F$ | Set of (mobile) users. |
| $K$ | Set of service types. |
| $\mathcal{T}_f$ | The activity duration of user $f \in F$. |
| $G = (N, L)$ | Network graph with node and link sets of $N$ and $L$. |
| $D_{i,j}$ | The delay of link $(i, j) \in L$. |
| $B_{i,j}$ | The capacity of link $(i, j) \in L$. |
| $Cost_L^{i,j}$ | The cost of activating link $(i, j) \in L$. |
| $\mathcal{N}_C/\mathcal{L}_C$ | The set of core network nodes/links. |
| $\mathcal{N}_{AP}/\mathcal{L}_{AP}$ | The set of nodes/links of the access points. |
| $\mathcal{N}_{DC}$ | Set of DC nodes. |
| $C_j^{DC}$ | Capacity of DC node $j$. |
| $C^{VM}$ | Processing Capacity of VMs. |
| $\Delta_k^{VM}$ | Size of VM type $k$. |
| $\mathcal{R}$ | Set of user requests. |
| $\mathcal{R}^t$ | Set of user requests at timeslot $t$. |
| $src_r$ | Source node (user location) of request $r$. |
| $k_r$ | Service type of request $r$. |
| $\mathcal{B}_r$ | Bandwidth requirement of request $r$. |
| $\mathcal{D}_r$ | Delay requirement of request $r$. |
| $Size_k^{VM}/Cost_k^{VM}$ | The size and cost of a VM instance, providing service type $k$. |
| $Cost_k^{mig}$ | Cost of migrating the VM with service type $k$. |
| $\Psi^+(i), \Psi^-(i)$ | Outgoing and incoming nodes from and to node $i$ |
| $\mathcal{M}$ | Migration cost function |
| Decision Variables | |
| $x_{r,j} \in \{0, 1\}$ | =1 if user node (service request) $r$ is assigned to DC $j$. |
| $x_{r,j}^t \in \{0, 1\}$ | =1 if user node (service request) $r$ is assigned to DC $j$ at timeslot $t$. |
| $n_{j,k} \geq 0$ | Shows the number of deployed VMs of type $k$ in DC node $j$. |
| $n_{j,k}^t \geq 0$ | Shows the number of deployed VMs of type $k$ in DC node $j$ at timeslot $t$. |
| $l_{u,v}^{r,j} \in \{0, 1\}$ | =1 if the link $(u, v)$ is on the path of request $r$ traffic to DC $j$. |
| $l_{u,v,t}^{r,j} \in \{0, 1\}$ | =1 if the link $(u, v)$ is on the path of request $r$ traffic to DC $j$ at timeslot $t$. |
| $y_{u,v} \in \{0, 1\}$ | =1 if link $(u, v)$ is active. |
| $y_{u,v}^t \in \{0, 1\}$ | =1 if link $(u, v)$ is active at timeslot $t$. |
| $m_k^t \geq 0$ | Number of VM migrations for service type $k$ at time $t$. |

### 4.3.6 Cost Models

As we mentioned before, this optimization framework considers three types of cost: The cost of VM deployment, routing, and VM migration. In below, we explain each cost in detail:

#### 4.3.6.1 VM Deployment Cost

Without loss of generality, two approaches can be introduced to calculate the VM deployment cost:

**1. Owning a VM:** An option for the service providers is to build their own infrastructure that can offer services based on deployed VMs. In this case, the costs can be calculated based on computational and non-computational costs. According to [45], the computational costs include physical servers (i.e., memory and processor) and physical storage. On the other hand, non-computational costs consist of the cost of hypervisor licenses, operating system, management software, network connectivity, and software/hardware maintenance [45].

**2. Renting a VM:** A more popular option among companies is to rent VMs in distributed public clouds [230]. It is the method that we use in this chapter. Using this approach, clients (service providers in our case) can schedule and rent VMs with different processing capacities. For instance, Amazon EC2 offers widespread VM types with different renting duration, processing capacities, and prices [29]. In our scenario, we used this approach to model the VM deployment costs, which is calculated as below:

$$\textit{Total VM Deployment Cost} = \sum_{j \in \mathcal{N}_{DC}} \sum_{k \in \mathbb{K}} n^{t}_{j,k} \times Cost^{VM}_{k} \tag{4.1}$$

where $n^{t}_{j,k}$ is the number of VMs for service type $k$ in DC $j \in \mathcal{N}_{DC}$. Also, $Cost^{VM}_{k}$ is the price associated to deploying one VM instance providing service type $k$.

As it can be seen, we consider VM costs based on the service type only. However, an improvement to this model can be to consider different flavors for VMs. For instance, the flavor of each VM for service type $k$ can be either *small*, *medium*, *large*, or *xlarge*. In this case, the VM processing capacity and price is increasing from *small* to *xlarge* flavors. Therefore, by using different flavors, the provided solution can be more cost-efficient, since the flexibility of solution would be increased.

#### 4.3.6.2 Routing Cost

In our considered network, there are two types of network links: *i)* core, and *ii)* access points links. These links can be leased by the client (i.e., service provider company) in two manners:

**1. Pay-Per-Bandwidth Usage:** In this case, several number of links can be used, but the only parameter that contributes to the routing cost is bandwidth utilization. In this case, the routing cost can be calculated as below:

$$\textit{Total Routing Cost} = \sum_{r \in \mathcal{R}^{t}} \sum_{(u,v) \in L} \sum_{j \in \mathcal{N}_{DC}} \frac{(\mathcal{B}_{r} \times l^{r,j}_{u,v,t})}{B_{u,v}} \times Cost^{L}_{u,v} \tag{4.2}$$

where total routing cost is calculated as cost (e.g., \$) per bandwidth unit, $\mathcal{B}_{r} \times l^{r,j}_{u,v,t}$ is the amount of traffic for demand $r \in \mathcal{R}^{t}$ that is being sent over link $(u,v)$ at timeslot $t \in T$. Also, $B_{u,v}$ is

the bandwidth capacity of link $(u, v)$. This approach is suitable for scenarios where the network bandwidth is shared between different tenants (e.g., companies). In this way, each tenant can lease the network and pay the cost based on its usage.

**2. Pay-Per-Link:** In this approach, the client leases a link, no matter how much bandwidth would be utilized. Therefore, the focus here would be to reduce the number of used links as much as possible. In our scenario, we used this approach to model the total routing cost. The total routing cost by using the pay-per-link approach can be modeled as:

$$\textit{Total Routing Cost} = \sum_{t \in T} \sum_{(u,v) \in L} y^t_{u,v} \times Cost^L_{u,v}, \tag{4.3}$$

where $y^t_{u,v}$ is the indicator variable to show if the network link $(u, v)$ is active (i.e., used) at timeslot $t \in T$. Also, $Cost^L_{u,v}$ is the cost of using/activating the network link $(u, v)$.

We now introduce the S-JSPARM optimization formulation.

## 4.4 Static Joint Service Placement, Assignment, Routing, and Migration (S-JSPARM) Formulation

The static case is defined as follows: given a network $G$ and a set of mobile users with service requests of $A$, place the required VMs in the DCs and find the route to the users, such that the network and VM deployment costs are minimized and all the service requirements (bandwidth and maximum delay) are guaranteed. Inspired by [170], we formulate S-JSPARM by integrating the classical multi-commodity flow and capacitated facility location-routing problems into an ILP optimization. Since we focus on the static case, the problem is solved for a single timeslot. Nevertheless, in Section 4.5, we introduce the multi-period case of the formulation.

### 4.4.1 Objective Function

The objective of the S-JSPARM optimization model is to minimize the total VM deployment and routing cost. Based on the models introduced in Section 4.3.6, in this chapter, we use *renting a VM* and *pay-per-link* cost models for calculating the VM and routing costs, respectively. Thus, we can define the objective function $P$ as below.

$$\mathbf{P} = \sum_{j \in \mathcal{N}_{DC}} \sum_{k \in K} Cost^{VM}_k n_{j,k} + \sum_{(u,v) \in L} Cost^L_{u,v} y_{u,v}, \tag{4.4}$$

where $Cost^{VM}_k$ is the deployment cost of a single VM for service type $k$. $Cost^L_{u,v}$ is the cost of using link $(u, v) \in L$. Also, $n_{j,k}$ is an integer decision variable that shows the number of deployed VMs of service type $k$ at DC node $j$. Moreover, $y_{u,v}$ is a binary decision variable that indicates if the link $(u, v) \in L$ is active.

### 4.4.2 User-to-DC Assignment

Each user request $r$ must be assigned to a VM which is already hosted by a DC node. Thus, we have:

$$\sum_{j \in \mathcal{N}_{DC}} x_{r,j} = 1, \forall r \in \mathcal{R}, \tag{4.5}$$

where $x_{r,j}$ is a binary decision variable that is equal to 1, if the user request $r$ is assigned to a DC node $j$. Thus, the above constraints ensure that each service request is assigned to one and only one DC.

### 4.4.3 VM Existence

The user request $r$ with service type $k_r$ can be assigned to a DC node $j$, if at least a VM of service type $k_r$ is deployed and exists on DC node $j$. We can write this constraint as:

$$x_{r,j} \leq n_{j,k_r}, \forall r \in \mathcal{R}, \forall j \in \mathcal{N}_{DC}, \tag{4.6}$$

### 4.4.4 DC Capacity

The available resources at each DC node are limited to $C^{DC}$. Therefore, a limited number of VMs can be deployed on each DC. This can be formulated as:

$$\sum_{k \in K} \Delta_k^{VM} n_{j,k} \leq C_j^{DC}, \forall j \in \mathcal{N}_{DC}, \tag{4.7}$$

where $\Delta_k^{VM}$ denotes the size of a VM of type $k$.

### 4.4.5 VM Capacity

Similar to DCs, VMs are also resource-constrained. In other words, each VM can serve a limited amount of users and traffic at the same time. This can be written as:

$$\sum_{r \in \mathcal{R}} \mathcal{B}_r x_{r,j} \leq C^{VM} n_{j,k_r}, \forall j \in \mathcal{N}_{DC}, \tag{4.8}$$

where $C^{VM}$ is the processing capacity of VM and $\mathcal{B}_r$ is the traffic volume of user request $r$.

### 4.4.6 Network Capacity

The capacity of the network link $(u, v) \in L$ is limited to $B_{u,v}$. Therefore, on each link, the sum of user traffic that is traversed through it should not be more than the capacity of that link. This constraint is modeled as:

$$\sum_{r \in \mathcal{R}} \sum_{j \in \mathcal{N}_{DC}} l_{u,v}^{r,j} \mathcal{B}_r \leq B_{u,v}, \forall (u, v) \in L, , \tag{4.9}$$

where $l_{u,v}^{r,j}$ is a binary decision variable that is equal to 1, if the traffic from user request $r$ is routed using link $(u, v)$ toward DC node $j$.

### 4.4.7 Delay Requirement

The required delay $\mathcal{D}_r$ from user request $r$ needs to be guaranteed. To achieve this, the sum of the network links' propagation delay must be equal to or lower than the required delay $\mathcal{D}_r$ by the user. This can be formulated as:

$$\sum_{(u,v)\in L} \sum_{j\in\mathcal{N}_{DC}} l_{u,v}^{r,j} D_{u,v} \leq \mathcal{D}_r, \forall r \in \mathcal{R},, \tag{4.10}$$

where $D_{u,v}$ is the propagation delay of link $(u,v)$.

### 4.4.8 User Traffic Generation

The traffic needs to be generated from the source node, where the user is. To do this, the traffic needs to be generated and leave the source node from one of its outgoing links. We can achieve this by:

$$\sum_{v\in\Psi^+(u)} \sum_{j\in\mathcal{N}_{DC}} l_{src_r,v}^{r,j} = 1, \forall r \in \mathcal{R},, \tag{4.11}$$

where $\Psi^+(u)$ is the set of outgoing links from node $u$. As it can be seen, for each user request $r$, the summation of variable $l_{src_r,v}^{r,j}$ should be equal to 1, which the source node is $src_r$.

### 4.4.9 Flow Conservation Law

After the traffic is generated, it needs to traverse the network towards the assigned DC. Each node of the network should receive this traffic and forwards it from one of its links, except in the case that the node is a DC. This is named flow conservation law, which can be presented as below:

$$\sum_{v\in\Psi^+(u)} \sum_{j\in\mathcal{N}_{DC}} l_{u,v}^{r,j} - \sum_{v\in\Psi^-(u)} \sum_{j\in\mathcal{N}_{DC}} l_{u,v}^{r,j} = \begin{cases} 0 & ,\forall r \in \mathcal{R}, \forall u \in \{N \setminus \mathcal{N}_{DC}\}, u \neq src_r \\ x_{r,u} & ,\forall r \in \mathcal{R}, \forall u \in \mathcal{N}_{DC} \end{cases}, \tag{4.12}$$

where $\Psi^-(u)$ is the set of incoming links to node $u$.

### 4.4.10 Routing/Assignment Relation

There is a relation between routing and assignment decisions that must be met. This relation is, the routing must be performed towards the assigned DC. This relation can be written as:

$$l_{u,v}^{a,j} \leq x_{r,j}, \forall (u,v) \in L, \forall r \in \mathcal{R}, \forall j \in \mathcal{N}_{DC}. \tag{4.13}$$

In other words, the above constraints state that traffic routing must be passed through for the service requests that are assigned to a DC $j$.

### 4.4.11 Network Link Status

The final set of constraints are indicators for the status of network links. It is controlled by the binary variable $y_{u,v}$, which is equal to 1, if at least a user traffic is passing through the link $(u,v) \in L$.

Therefore, we have:

$$y_{u,v} \leq \sum_{r \in \mathcal{R}} \sum_{j \in \mathcal{N}_{DC}} l_{u,v}^{r,j}, \forall (u,v) \in L, \tag{4.14}$$

$$\rho y_{u,v} \geq \sum_{r \in \mathcal{R}} \sum_{j \in \mathcal{N}_{DC}} l_{u,v}^{r,j}, \forall (u,v) \in L \tag{4.15}$$

where $\rho$ is a large positive number ($\rho \gg 0$).

Having the objective function and the constraints, we can present the S-JSPARM optimization formulation as follows:

$$\begin{aligned} &\text{Minimize} \quad \mathbf{P}, \\ &\text{s.t. constraints } (4.5) - (4.15), \\ &\text{vars: } x_{r,j} \in \{0,1\}, \forall r \in \mathcal{R}, \forall j \in \mathcal{N}_{DC}, \\ &n_{j,k} \geq 0, \forall j \in\in \mathcal{N}_{DC}, \forall k \in K, \\ &l_{u,v}^{r,j} \in \{0,1\}, \forall r \in \mathcal{R}, \forall j \in \mathcal{N}_{DC}, \forall (u,v) \in L. \end{aligned} \tag{4.16}$$

In the next section, we extend S-JSPARM, and present the MA-JSPARM optimization model.

## 4.5 Mobility-Aware Joint Service Placement, Assignment, Routing, and Migration (MA-JSPARM) Formulation

MA-JSPARM adds a new dimension to S-JSPARM: *time*. In this case, MA-JSPARM addresses these questions by minimizing the total operational cost over the whole time horizon. In MA-JSPARM, in addition to VM deployment and routing costs, there are costs associated to the VM migration (referred as migration cost $Cost^{mig}$). Before formulating the MA-JSPARM, we first present the VM migration cost modeling.

### 4.5.1 VM Migration Cost

As mentioned before, VMs can be migrated between DCs if necessary. VM migration is not only a beneficial operation. It can induce several types of overheads and costs to both operators and clients [227, 211], such as power consumption overheads, performance degradation, and/or increase of network bandwidth usage [195, 149]. Therefore, VM migration must be carefully planned and operated. Accordingly, in this work, we model the total VM migration cost as below:

$$\textit{Total VM Migration Cost} = \sum_{t \in T} \sum_{k \in K} Cost_k^{mig} \times m_k^t \tag{4.17}$$

where $Cost_k^{mig}$ is the cost of migrating a VM for service type $k$. This cost is calculated as $\mathcal{M}$, the function of VM cost (i.e., proportional to the VM size) and the core ground network link cost:

$$Cost_k^{mig} = \delta \times \mathcal{M}(Cost_{u,v}^{L}, Cost_k^{VM}), (u,v) \in \mathcal{L}_C, k \in K, \tag{4.18}$$

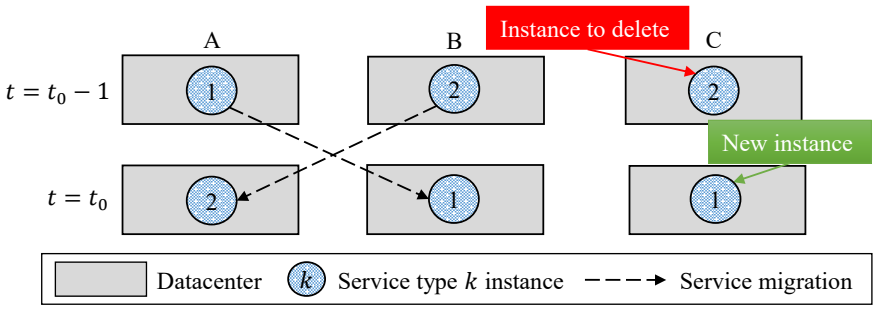



**Figure 4.1:** An example of the VM migration computation for three DCs A, B, and C, when two VM migrations, one service deletion, and one creation is performing.

where $0 \leq \delta \leq 1$ is the weighting factor used to tune the migration cost. $\delta$ is useful to mimic different migration cost based on the specific application and network. Nevertheless, more factors such as server CPU utilization and application sensitivity against migration can be considered to formulate the migration cost function [221]. We note that presenting an accurate and service-specific migration cost model is out of scope of the thesis.

Additionally, $m_k^t$ is the number of VM migrations that need to be performed at timeslot $t \in T$ for service type $k$. In order to compute the number of migrations of service type $k$ at the timeslot $t \in T$, we propose Eq. 4.19:

$$m_k^t = \sum_{j \in \mathcal{N}_{DC}} [n_{j,k}^t - n_{j,k}^{t-1}]^+ - \Big[ \sum_{j \in \mathcal{N}_{DC}} n_{j,k}^t - \sum_{j \in \mathcal{N}_{DC}} n_{j,k}^{t-1} \Big]^+, \forall k \in K, \forall t \in T^+, \tag{4.19}$$

where $[a]^+$ is defined as $max\{0, a\}$. Let us note that $m_k^0 = 0$, since VM migration cannot happen at $t = 0$. Consider the example in Fig. 4.1 with two VM instances $K = \{1, 2\}$ and three DCs $A$, $B$, and $C$. Suppose at timeslot $t_0 - 1$, one VM instance of $k = 1$ is required at DC $A$, and two VM instances of $k = 2$ are required at DC $B$, and $C$. However, at next timeslot ($t_0$), due to the mobility of users and increase in their distance with the current VM, two VM instances of $k = 1$ are required at B and C, whereas only one VM instance for $k = 2$ is required at DC $A$. This new placement of services requires a VM migration for $k = 1$ from DC A to B, a second VM migration for service $k = 2$ from DC B to A, and a VM instance creation and deletion at DC C for service types $k = 2$ and $k = 1$, respectively. Applying Eq. 4.19 to calculate the number of VM migrations for each service type $k$ ($m_t^k$), we obtain:

$$(k = 1) : m_1^{t_0} = [(0 - 1)^+ + (1 - 0)^+ + (1 - 0)^+] - [(2 - 1)^+] = 1,$$
$$(k = 2) : m_2^{t_0} = [(1 - 0)^+ + (0 - 1)^+ + (0 - 1)^+] - [(1 - 2)^+] = 1.$$

Thus, the total number of VM migrations for all $k \in K$ is given by $m_1^{t_0} + m_2^{t_0} = 2$.

Having the VM migration cost modeled, we define $\mathcal{R}^t$ as the set of service requests generated by users at timeslot $t$. We extend the variables defined in S-JSPARM to include a per timeslot index $t \in T$. Therefore, by using the same definition as in S-JSPARM model, we convert the variables of S-JSPARM to multi-period variables as $x_{i,j} \rightarrow x_{i,j}^t$, $l_{u,v}^{r,j} \rightarrow l_{r,j}^{u,v,t}$, $n_{j,k} \rightarrow n_{j,k}^t$, and $y_{u,v} \rightarrow y_{u,v}^t$. Additionally, we define a new variable $m_k^t \geq 0$ to include the VM migration decisions. $m_k^t$ represents the number of VM migrations for service type $k \in K$ at the timeslot $t \in T$.

Considering the above explanations, we can present the formulation of MA-JSPARM. Once again, we start by forming the objective function **Q**. As mentioned before, the objective function of MA-JSPARM is defined as the sum of VM deployment, routing, and VM migration costs over time horizon $T$. It can be written as:

$$\mathbf{Q} = \sum_{t\in T}\sum_{j\in\mathcal{N}_{DC}}\sum_{k\in K} Cost_k^{VM} n_{jk}^t + \sum_{t\in T}\sum_{(u,v)\in L} Cost_{u,v}^{L} y_{u,v}^t + \sum_{t\in T}\sum_{k\in K} Cost_k^{mig} m_k^t. \tag{4.20}$$

Therefore, the MA-JSPARM model can be presented as:

$$\text{Minimize} \quad \mathbf{Q}, \tag{4.21}$$

$$\text{s.t.} \quad \sum_{j\in\mathcal{N}_{DC}} x_{r,j}^t = 1, \forall t \in T, \forall r \in \mathcal{R}^t, \tag{4.22}$$

$$x_{r,j}^t \le n_{j,k_r}^t, \forall r \in \mathcal{R}^t, \forall t \in T, \forall j \in \mathcal{N}_{DC}, \tag{4.23}$$

$$\sum_{k\in K} \Delta_k^{VM} n_{j,k}^t \le C_j^{DC}, \forall t \in T, \forall j \in \mathcal{N}_{DC}, \tag{4.24}$$

$$\sum_{r\in\mathcal{R}^t} \mathcal{B}_r x_{r,j}^t \le C^{VM} n_{j,k_r}^t, \forall t \in T, \forall j \in \mathcal{N}_{DC}, \tag{4.25}$$

$$\sum_{r\in\mathcal{R}^t}\sum_{j\in\mathcal{N}_{DC}} l_{u,v,t}^{r,j} \mathcal{B}_r \le B_{u,v}, \forall t \in T, \forall (u,v) \in L, \tag{4.26}$$

$$\sum_{(u,v)\in L}\sum_{j\in\mathcal{N}_{DC}} l_{u,v,t}^{r,j} D_{u,v} \le \mathcal{D}_r, \forall t \in T, \forall r \in \mathcal{R}^t, \tag{4.27}$$

$$\sum_{v\in\Psi^+(u)}\sum_{j\in\mathcal{N}_{DC}} l_{src_r,v,t}^{r,j} = 1, \forall t \in T, \forall r \in \mathcal{R}^t, \tag{4.28}$$

$$\sum_{v\in\Psi^+(u)}\sum_{j\in\mathcal{N}_{DC}} l_{u,v,t}^{r,j} - \sum_{v\in\Psi^-(u)}\sum_{j\in\mathcal{N}_{DC}} l_{u,v,t}^{r,j} = \begin{cases} 0 & ,\forall t \in T, \forall r \in \mathcal{R}^t, \forall u \in \{N \setminus \mathcal{N}_{DC}\}, u \ne src_r \\ x_{r,u}^t & ,\forall t \in T, \forall r \in \mathcal{R}^t, \forall u \in \mathcal{N}_{DC} \end{cases}, \tag{4.29}$$

$$y_{u,v}^t \le \sum_{r\in\mathcal{R}^t}\sum_{j\in\mathcal{N}_{DC}} l_{u,v,t}^{r,j}, \forall t \in T, \forall (u,v) \in L, \tag{4.30}$$

$$l_{u,v,t}^{r,j} \le x_{r,j}^t, \forall (u,v) \in L, \forall t \in T, \forall r \in \mathcal{R}^t, \forall j \in \mathcal{N}_{DC}, \tag{4.31}$$

$$\rho y_{u,v}^t \ge \sum_{r\in\mathcal{R}^t}\sum_{j\in\mathcal{N}_{DC}} l_{u,v,t}^{r,j}, \forall t \in T, \forall (u,v) \in L, \tag{4.32}$$

$$\begin{aligned} \textit{vars:} \quad & x_{r,j}^t \in \{0,1\}, \forall t \in T, \forall r \in \mathcal{R}^t, \forall j \in \mathcal{N}_{DC}, \\ & n_{j,k}^t \ge 0, \forall t \in T, \forall j \in\in \mathcal{N}_{DC}, \forall k \in K, \\ & l_{u,v,t}^{r,j} \in \{0,1\}, \forall t \in T, \forall r \in \mathcal{R}^t, \forall j \in \mathcal{N}_{DC}, \forall (u,v) \in L. \end{aligned}$$

The constraints of MA-JSPARM model 4.22-4.32 are the extension of the S-JSPARM constraints 4.5-4.15 by time dimension.

### 4.5.2 Hardness

The formulated MA-JSPARM is a combination of the Multi-Period Capacitated Facility Location (MPCFL) and Multi-Commodity Flow (MCF) problems. The Capacitated Facility Location (CFL)

problem is known to be NP-Hard [73, 122, 165]. Therefore, the MPCFL, with an additional dimension (i.e., time) is also NP-Hard, since CFL is a special case of it. Additionally, the MCF with deadlines is proved to be NP-Hard [140]. As a result, the combination of these two problems is even harder to solve.

## 4.6 Proposed Cost-Efficient Heuristics

Considering the problem complexity, to be able to solve it for practical-size instances, we introduce two heuristic solutions, named User-by-User (UbU) and Timeslot-by-Timeslot (TbT). As shown in Fig. 4.2, in both of these heuristics, we decompose the problem into smaller sub-problems in different dimensions: one based on the users and the other based on timeslots. Considering their dependencies, we then combine these sub-problems to achieve a solution for the main problem.

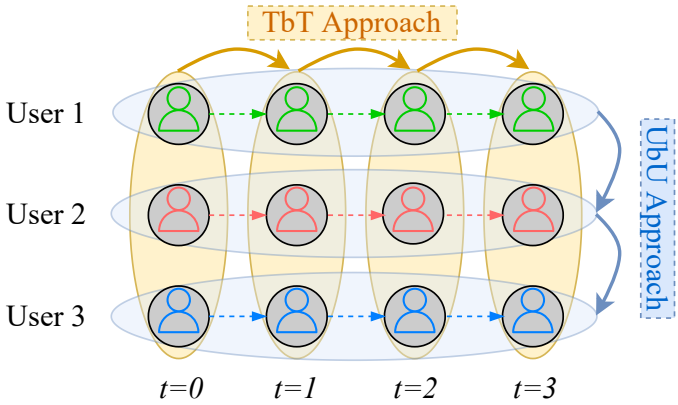



**Figure 4.2:** An example of UbU and TbT approaches for three mobile users and $T = 4$ timeslots (user positions at the beginning of each $t \in T$.)

### 4.6.1 User-by-User (UbU) Algorithm

In short, UbU algorithm solves the problem for each user sequentially over $T$ timeslots, taking into account the VM placement and the routing decisions for the previous users over $T$ (see Fig. 4.2). Moving from $t$ to $t + 1$, VM migrations are performed towards meeting delay requirements and/or total operational cost minimization.

The UbU heuristic consists of two parts: the first part (shown as Alg. 1) selects DC and places VM only for the first user in the input data set and the second part (shown as Alg. 2) does this for the remaining users in the input data set.

#### 4.6.1.1 Algorithm 1: First User

This algorithm is depicted in Alg. 1. For the first user, there is no pre-selected DC or placed VM available, therefore at each timeslot, only two options are possible. The first option is to share the VM with previous timeslots. In this case, the total cost will be sum of routing and VM deployment cost. The second to create a new VM at another DC and migrate the VM from the DC from the previous timeslot to the newly selected DC. In this case, the total cost will include the VM migration cost as well. $f_{\text{sm}}$ is a binary flag indicating whether the user of the current timeslot cannot re-use the

**Algorithm 1:** UbU Algorithm (First User)

**Data:** topology $G$, service type $k$, timeslot range $T$, migration cost $C_k^{mig}$
**Result:** VM placement, routing, and migration decisions at each timeslot, total operational cost

**for** *each timeslot* $t \in T$ **do**
  $f_{\text{sm}} = False$;
  /* Option 1: share one VM */
  Get user locations from $i_{last}$ to $i$;
  Find the DC ($sel_{dc}$) with the least total operational cost $C_1$ to all the locations, and calculate all latencies $L_{\text{list}}$;
  /* Option 2: migrate VM to a new DC */
  Get the best DC for the previous timeslots and return total operational cost $C_2$;
  Find the best DC for the current timeslot and add the routing cost to $C_2$;
  $C_2 = C_2 + C_k^{mig}$; // Add migration cost
  /* Select the option with lower total operational cost */
  **if** $C_1 < C_2$ **then**
    **for** $l$ *in* $L_{list}$ **do**
      **if** $l$ *violates service* $s$ *requirement* **then**
        $f_{\text{sm}} = True$;
        Break;
  **else**
    $f_{\text{sm}} = True$;
  **if** $f_{sm}$ **then**
    Add service $s$ to a new VM in the new DC;
  **else**
    Share one VM with the previous timeslots;
Update link remaining capacity of the topology for future routing decisions;

same DC with previous timeslots (i.e., VM migration). Also, $i_{last}$ is the timeslot when the user cannot share the VM with the previous timeslots. In order to make a decision, we compare the overall costs induced by the two options. After the decision, the remaining capacity of each DC and each link in the topology is updated accordingly. Note that the delay constraint is enforced while selecting the candidate DC nodes.

#### 4.6.1.2 Algorithm 2: Remaining Users

This algorithm is presented in Alg. 2. There are three options at each timeslot for the remaining users, and we compare the additional costs result by different options. The first option is to check all instantiated VMs of previous users that are available at the current timeslot and find the one with the minimum routing cost from the current location of the user. Here, the only incurred cost is the routing cost. For the second option, we assign the user to the VM (if there is any) which the user was assigned to it in the last timeslot. The additional cost in this case, is the sum of VM cost and routing cost. For the last option, we migrate the VM for the current user from the DC it used to be at

**Algorithm 2:** UbU Algorithm (Remaining Users)

**Data:** topology $G$, service type $k$, timeslot range $T$, migration cost $C_k^{mig}$
**Result:** VM placement, routing, and migration decisions at each timeslot, total operational cost

```
/* starting from the second user                                    */
for each user f ∈ {2..F} do
    for each timeslot t ∈ T do
        /* Option 1: share the VM used by previous users            */
        Get all VMs at this timeslot with enough remaining capacity;
        Find the VM which incurs minimum routing cost;
        Assign routing cost to C_1;
        /* Option 2: use the same VM from the previous timeslots    */
        Find the best DC location with a VM from the previous timeslot;
        Assign the sum of routing cost and VM cost to C_2;
        /* Option 3: migrate the VM to a new DC                     */
        Find the best DC location for this timeslot;
        Assign the sum of routing cost and VM migration cost to C_3;
        Select the option with the minimum cost from C_1, C_2, or C_3;
        Add service to the corresponding VM;
Update link remaining capacity of the topology for future routing decisions;
```

the last timeslot to a new DC, which induces additional routing, VM and migration cost. The three options are compared against each other in terms of the additional cost. Besides all locations of the fixed entities, e.g., DCs, routers, links, access point; the network input $G$ contains the user locations in the timeslot range $T$.

### 4.6.2 Timeslot-by-Timeslot (TbT) Algorithm

This heuristic solves the problem per timeslot, that is, based on the position of all users at a given timeslot $t \in T$. Solving the problem for timeslot $t$, TbT considers the network status (VM-to-DC mapping, and routing) from timeslot $t-1$ (see Fig. 4.2). However, unlike UbU, TbT does not change the decisions that are made for the previous timeslots, since it does not consider the whole time horizon for the decision making.

At each timeslot, TbT takes a user from the input data, and it finds the DC which has the lowest routing cost from the user's location at the respective timeslot. It starts with the routing cost because usually, this is the dominant share from the total operational cost, compared to VM and migration costs. Before selecting the DC, it checks the link bandwidth (constraint 4.26), and end-to-end delay constraints (constraint 4.27). If one of them is not met, it selects the next DC with the lowest routing cost. Thereafter, it re-uses the existing VM for the requested service type $k$ at the DC, if the VM capacity constraint (constraint 4.25) are not violated. If this VM is not available, it will launch a new instance of the VM in the respective DC. However, it validates the DC capacity constraint (constraint 4.24) for possible violations. The algorithm selects the DC based on the total routing and VM deployment costs. It repeats the same for all the users for the current timeslot and then moves to the next until the limit of timeslots $T$ is reached. Finally, the migration cost is calculated

**Algorithm 3:** TbT Algorithm

**Data:** topology $G$, timeslot range $T$
**Result:** VM placement, routing, and migration decisions at each timeslot, total operational cost

$C_{total} = 0$;
**for** *each timeslot* $t \in T$ **do**
  **for** *each user* $f \in F$ **do**
    selectedDC = null;
    $C_{min} = 0$;
    **for** *each DC* $j \in \mathcal{N}_{DC}$ **do**
      $C_1$ = calculate the routing cost to DC $j$ from user's $f$ location at timeslot $t$;
      /* Check the delay requirements */
      **if** *delay constraint is* ***not*** *met* **then**
        continue;
      $C_2$ = calculate the VM deployment cost at DC $j$;
      /* Check the DC and VM capacity constraints */
      **if** *VM or DC capacity constraints are* ***not*** *met* **then**
        continue;
      **if** $C_1 + C_2 < C_{min}$ **then**
        selectedDC = $j$;
      **else**
        continue;
    $C_{total} += C_{min}$;
    Add service to the corresponding VM;
    Update link remaining capacity of the topology for future routing decisions;
  /* Calculate the VM migration cost */
  $C_3$ = calculate the migration cost based on the model in Eq. 4.19. $C_{total} += C_3$

based on the migrations that are decided between the consequent timeslots. Of course, it migrates a VM if all the user connections are moved at the same time from the respective VM. The description of the TbT algorithm is summarized in Alg. 3.

## 4.7 Performance Evaluation

In this section, we firstly introduce the use case scenario and the input parameters. Then, we compare the performance of S-JSPARM and MA-JSPARM models, in terms of total operational cost, number of VM migrations, average end-to-end delay, and runtime. After that, we evaluate the performance of the heuristic algorithms and compare them with the optimal solution (i.e., MA-JSPARM).

### 4.7.1 Usecase: In-Flight Service Provisioning (SAGIN)

Today, with a significant increase in worldwide user traffic volume, providing in-flight services such as entertainment to onboard passengers has become an essential goal for airline companies [32]. Recently, there have been advances in providing high-capacity Air-to-Ground (A2G) communications

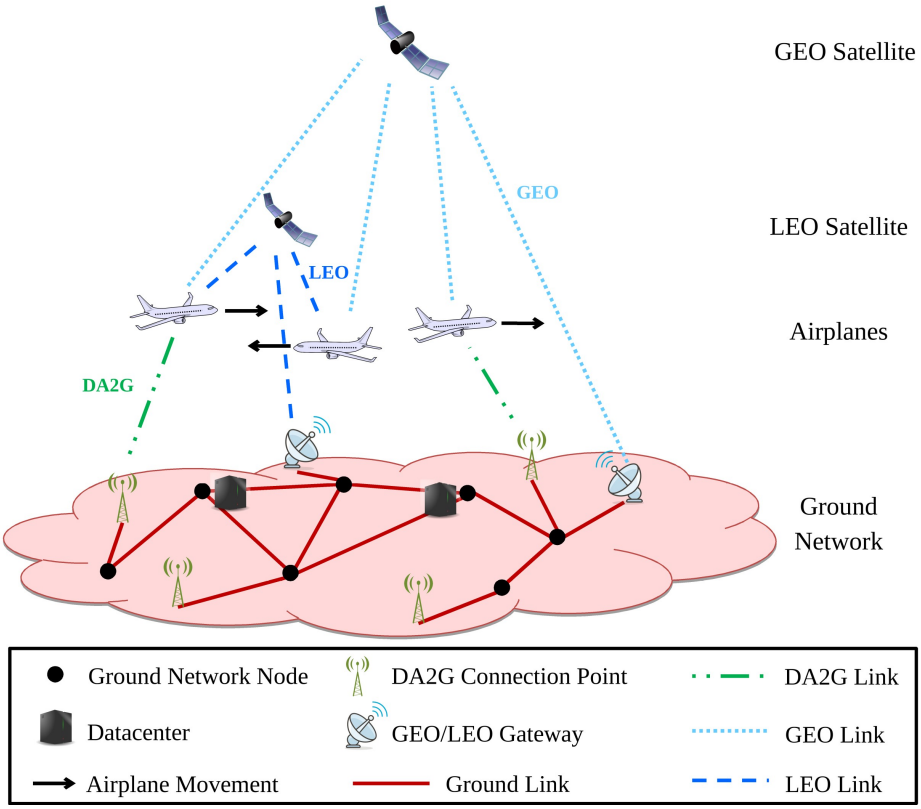



**Figure 4.3:** A high-level view of the Space-Air-Ground Integrated Network (SAGIN).

to airplanes [107, 90]. In particular, LEO satellites can relay the airplane traffic towards a gateway on the ground with a high link capacity [180]. Also, Direct-Air-to-Ground (DA2G) connections enable airplanes to communicate directly to the base stations located on the ground network [83][214]. These developments have made new opportunities for airlines to deploy new *online* and interactive services, such as voice/video calls and gaming, to improve passenger experience and grow the revenues [32]. To realize these services, airline companies can move the servers providing the in-user services from the airplane to the distributed DCs located on the ground. This can also reduce the weight of the airplane, hence the fuel consumption.

Fig. 4.3 shows a high-level view of a SAGIN. Firstly, it can be seen that some airplanes are flying over the ground network. In this case, the airplanes are the users, and the ground network is the core network of our system model. Note that the trajectory of the airplanes is known before the flight begins, so it can be used for solving the JSPARM problem. Further, it can be seen that two different A2G alternatives (i.e., access points in our model) exist, which the airplanes can use to connect to the ground network and then, a particular DC: *i)* satellites, and *ii)* DA2G. The satellites are connected to their gateways, located in the ground network. They can relay the traffic from the airplane towards the gateway. Geostationary Earth Orbit (GEO) satellites are designed to have an orbital period equal to the Earth's rotation period. However, due to the high orbit altitude of these satellites, their Round Trip Time (RTT) is over 600 ms [180], being too large for some services. Therefore, in our evaluation, we do not consider GEO satellites. On the other hand, OneWeb, Telesat, and SpaceX have been developing LEO satellite constellations, which are located at lower altitudes

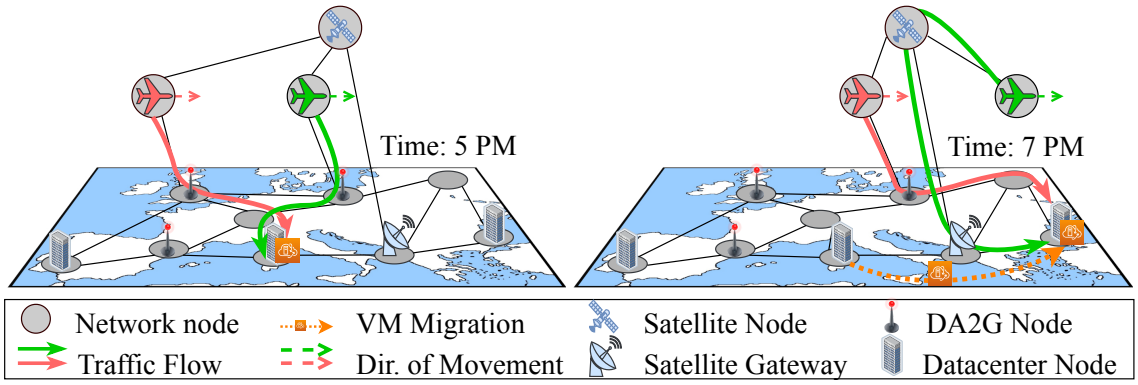



**Figure 4.4:** Two snapshots of a SAGIN, showing two flights, and VM placement, routing, and migration decisions over time.

(less than 2000 km) [180]. Thus, these satellites can offer network bandwidth with low latency (the average RTT is 50 ms [14]), which is more preferable in our use case.

The second option to connect an airplane to the ground network is DA2G, which establishes a direct connection from a flight to a base station on the ground if it is within its range. The connection to DA2G base stations changes due to the mobility of flights over time. A DA2G link produces around 10 ms of RTT [82].

It is evident that different A2G connection options differ in terms of provided bandwidth, delay, and cost. Thus, it is crucial to determine the DC to connect to as well as the underlying routing decisions to provide in-flight services to the airplane with the minimum cost.

Moreover, due to the mobility of airplanes over time, a DC may not remain the efficient one over time (i.e., the distance of the airplane to DC and hence the routing delay can increase during flight). Therefore, to maintain a continuous low-delay service delivery to the airplanes, the VM placement decisions can be adapted over time.

An example of our scenario is presented in Fig. 4.4. This figure shows two flights flying over a core network, requesting the same service type $k$. In this core network, there are three DCs, three DA2Gs, and one satellite node that relays the traffic to the ground gateway node. The left sub-figure shows the network status at 5 PM where both airplanes are connected through the available DA2G nodes to the central DC node. Within this DC, a single VM is deployed to serve both flights with minimum cost. After two hours, at 7 PM, the network status reaches the right sub-graph with the mobility of flights. In this sub-graph, it can be seen that the traffic routings are changed considering the available A2G connections. Moreover, to keep the delay below the requirement of the services, a VM migration has been triggered from the middle DC to the right one.

Below, we introduce the input parameters for the simulation scenario, and then we present the analysis of the results.

### 4.7.2 Simulation Setup

Our study focuses on a realistic European-based SAGIN. For the ground network topology, we use the PAN network topology [105]. We set the bandwidth for ground network links to 2 Gbps, and their delays were determined according to the length of the optical fiber transmissions. Further, the

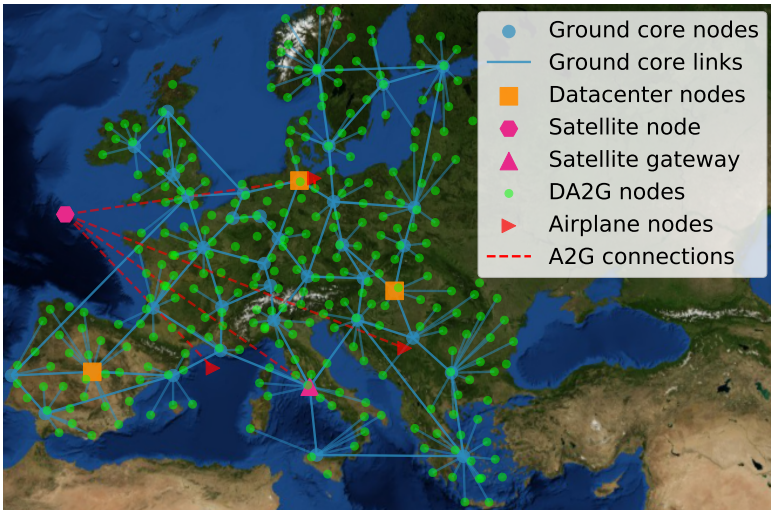



**Figure 4.5:** An example of the modelled European-based SAGIN with three flights.

locations for the DC nodes have been selected based on [76]: Athens (Greece), Helsinki (Finland), Liverpool (England), Strasbourg (France), Madrid (Spain), and Lviv (Ukraine). The locations of the DA2G base stations are chosen based on the actual deployment of 295 base stations in Europe [159] by Deutsche Telekom. Also, the bandwidth and delay of the DA2G access point links are considered as 75 Mbps and 10 ms [82], respectively.

According to the results presented in [169], it is observed that the number of LEO satellites moving over Europe is relatively low (5 out of the 72 LEO satellites in the Iridium constellation). Compared to GEO, utilizing LEO satellites introduces lower RTT, which is more suitable for our use case. Therefore, we only consider the LEO constellation for the satellite access point option. For simplification purposes, we apply an abstraction model, where the group of LEO satellites occupying Europe is represented as a single satellite node. Also, the set of links that connect each user to LEO node and from LEO to the satellite gateway is considered to have 50 Mbps capacity [202]. Moreover, to avoid unrealistic assumptions, we set the latency of the satellite links to the worst-case achievable latency of 50 ms, according to LEO latency calculations provided in [161]. Also, a satellite gateway has been considered in Rome, Italy [202]. The set of considered flights has been exported from FlightRadar24 live air traffic for 24 hours on 9.11.2017. In our experiments, assuming $\omega = 30$ minutes, we consider two set of flights with different duration (denoted by $\tau$): *i) Short* flights $F_s$, such that $\tau_s = 3, \forall f \in F_s$; and *ii) Long* flights $F_l$, where $\tau_l = 7, \forall F \in F_l$. Due to the time complexity issues, the size of each set of flights is considered as 20. Without loss of generality, we set the visibility distance from each user to the closest DA2G base station as 350 km as the absolute geometrical maximum [123], although in reality, it can vary based on, e.g., the antenna type and weather conditions. Fig. 4.5 shows an exemplary graph $G_t$ with three flights at a specific timeslot.

In order to reduce the time complexity of the optimization, two aggregated service requests per user per timeslot have been considered. These two service types are defined as:

1. Video streaming with bandwidth and end-to-end delay requirement of 1.5 Mbps [166] and 300 ms [89], respectively.

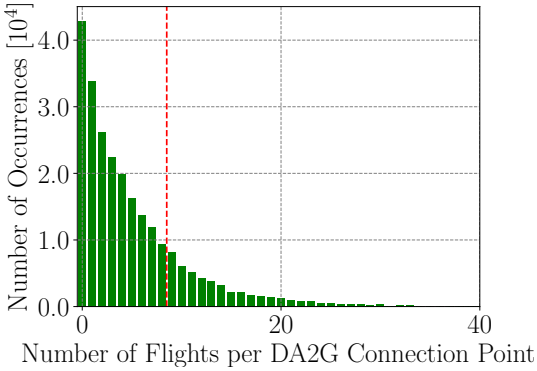



**Figure 4.6:** Number of flights in a DA2G base station coverage range, based on the real data of flights for 24 hours over Europe.

2. Voice-over-IP (VoIP) with 64 Kbps bandwidth and 100 ms [13] of delay requirements.

We assumed the airplane models as Airbus A320 with 150 passengers [26], where 20% of the passengers use the aforementioned network services [28]. We assume these requests are divided between our two service types equally.

To make the scenario even more realistic, we create a DA2G congestion probability model, which disables the DA2G link at a specific location, based on our flight data. Fig. 4.6 shows the histogram summarizing all the flights over Europe in 24 hours. According to our flight data, the DA2G link capacity, and the services required by a flight, every DA2G base station can serve simultaneously only around nine flights (see the vertical line in Fig. 4.6). Based on this study, the probability of DA2G congestion is 19.714%. This information is used to determine the congestion of DA2G base stations at each timeslot as input to the simulation.

An essential parameter in this study is cost, which has been considered in USD. According to Amazon Web Services [37], Compute-Optimized instance types is suggested for high performance purposes, e.g., video streaming. Therefore, we assume all $k \in K$ use the same *c4.2xlarge* instance type, which costs $229 per month [29]. On the other hand, the link costs are associated with the bandwidth required by our services ($\sim$ 20 Mbps), which are 60 and 130 USD for ground and satellite link use, respectively [52, 127]. Regarding the cost of using a DA2G link, we assumed it to be 83 USD. For the VM migration cost, we use the model presented in Section 4.5.1. We have modelled the SAGIN network $G$ by Python Networkx 2.1. The proposed S-JSPARM and MA-JSPARM optimization models have been implemented using Gurobi Optimizer 8.0.1 with 5% solution gap for scalability purposes. Also, the heuristics are developed in a simulation tool based on Python language. The simulations were executed on a desktop computer, equipped with Intel Core i7-6700 @3.40 GHz CPU, 16 GB of RAM, running Ubuntu 18.04 x64 OS. Each scenario with random flights from the data set is chosen and repeated 30 times for statistically reliable results.

### 4.7.3 Simulation Results

In this part, we first present the comparison of S-JSPARM and MA-JSPARM models in the scenario above in terms of different costs. After that, we present the analysis of results for comparing the performance of the MA-JSPARM model, with the UbU, TbT, and the baseline heuristics.

#### 4.7.3.1 S-JSPARM and MA-JSPARM Comparison

In order to evaluate the impact of different VM migration costs with respect the network and VM deployment costs, we vary the migration cost value by increasing $\delta$ from 0.1 and 1, with step size of 0.1. In our first study, we compare the total operational cost (including VM migration, network, and VM deployment costs) when applying MA-JSPARM and S-JSPARM for the two different types of users: short flights with $\tau_s = 3$ and long flights with $\tau_l = 7$. Also, the impact of having other users flying in the same area has been considered by the DA2G link congestion probability.

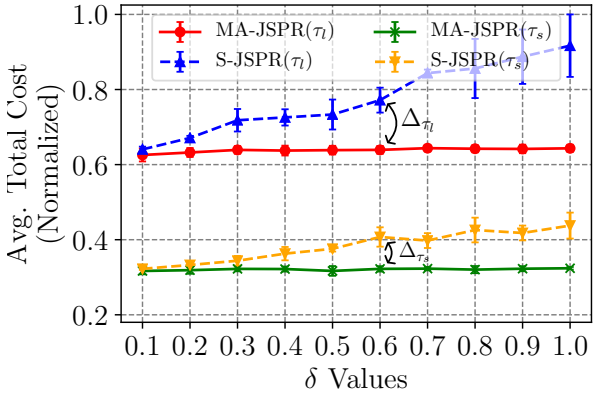



**Figure 4.7:** MA-JSPARM and S-JSPARM comparison: average normalized total operational cost

The total operational cost of the models is compared in Fig. 4.7. It can the observed that the cost of MA-JSPARM is always lower than S-JSPARM and the difference increases with the increase of $\delta$. Also, it can be seen that the cost of flights with longer duration ($\tau_l$) is higher than the short ones ($\tau_s$) since the number of required resources is higher for the long user. Also, the number of VM migrations is expected to be higher for the longer flights, which contributes to the total operational cost. Moreover, it can be seen that the cost difference for the case with long flights is higher than the short ones ($\Delta_{\tau_l} > \Delta_{\tau_s}$). Hence, it can be concluded that the amount of cost savings is higher when more information about the future is taken into account. It has to be mentioned that in our use case, the future positions of the flights are known, which makes the problem more straightforward to solve. In other use cases, such as autonomous vehicles, the vehicle mobility pattern and its future positions rely on models, which are not easy to determine.

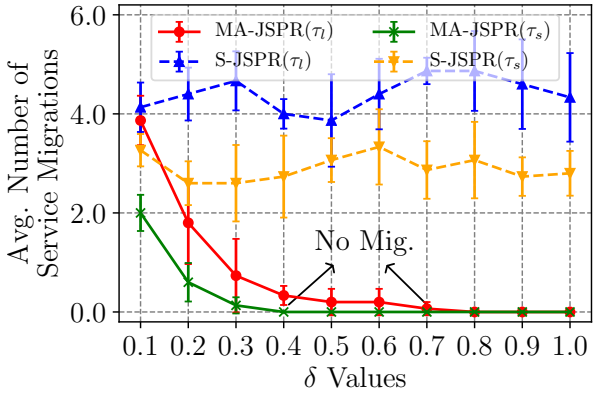



**Figure 4.8:** MA-JSPARM and S-JSPARM comparison: average number of VM migrations.

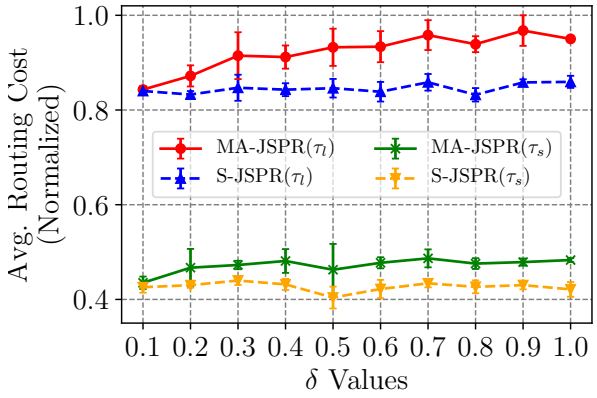



**Figure 4.9:** MA-JSPARM and S-JSPARM comparison: average normalized routing cost

Secondly, Fig. 4.8 shows the comparison of the number of VM migrations for MA-JSPARM and S-JSPARM. It can be observed that the average number of VM migrations in S-JSPARM is significantly higher than MA-JSPARM. This observation is due to the fact that S-JSPARM does not take the future flight locations into account. Moreover, it can be seen that the number of VM migrations for the long flights case $\tau_l$ is always higher than $\tau_s$, since the flights are active for a longer period, and hence, the probability and need for migration raises. The figure also shows that the number of VM migrations decreases with the increase of $\delta$ value.

However, this can form a trade-off between the routing cost and $\delta$ (or the number of VM migrations). For example, if the value of $\delta$ is large and the flight is moving, in future positions, more routing cost needs to be paid to reach DC that hosts the particular VM. This trade-off can be seen in Fig. 4.9. As is reported, for the $\tau_s$ and $\tau_l$, MA-JSPARM model avoids VM migrations at $\delta = 0.4$ and $\delta = 0.7$, respectively (see Fig. 4.8). Accordingly, as Fig. 4.9 presents, the routing cost in these cases increases proportionally with $\delta$. After $\delta = 0.4$ and $\delta = 0.7$, the routing cost does not increase and shows fluctuations, since no VM migration is triggered anymore.

Additionally, Fig. 4.9 shows that S-JSPARM model produces lower routing costs compared to MA-JSPARM. The reason is S-JSPARM neglects the future positions of the flight and hence, it places the VM at the closest DC (i.e., low routing cost). We note that MA-JSPARM can control the number of VM migrations while meeting the end-to-end required delay of each service request.

Besides, the achieved average end-to-end delay of both services for S-JSPARM and MA-JSPARM is compared in Fig. 4.10. It can be observed that the service requirements are generally met. Also, we see that the delay is mostly below 50 ms, which means the DA2G is utilized most of the time to route the services to the selected DC. Moreover, Fig. 4.10 shows that in both $\tau_l$ and $\tau_s$ cases, S-JSPARM can achieve lower average end-to-end delay than MA-JSPARM. Again, this is because S-JSPARM does not consider the future flight positions to solve the problem. In our scenario, there is a local DC node close to the satellite gateway node in the ground. Consequently, it can be seen that the average delay for MA-JSPARM($\tau_s$) is higher than MA-JSPARM($\tau_l$). It is because the short user $\tau_s$ is passing by fewer DCs during the flight period compared to the longer $\tau_l$ one. Therefore, MA-JSPARM($\tau_s$) would prefer to use the local DC close to the satellite gateway to avoid the routing cost to reach farther DCs through the DA2G links. Additionally, it can be observed that the maximum achieved end-to-end delay in MA-JSPARM($\tau_l$) is higher than MA-JSPARM($\tau_s$). The reason is that in $\tau_l$ case, the

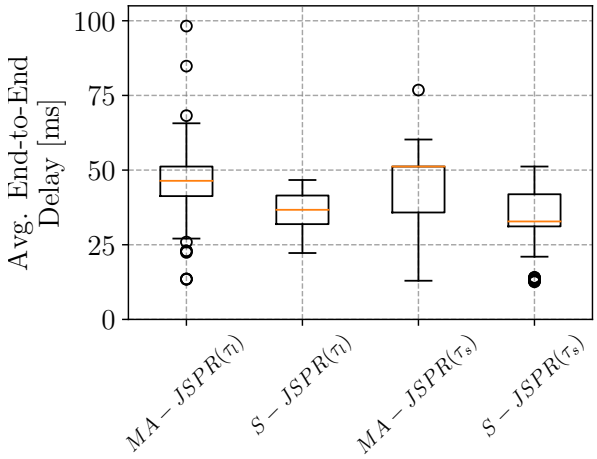



**Figure 4.10:** MA-JSPARM and S-JSPARM comparison: average end-to-end service delay.

paths would be longer than $\tau_s$ case to improve the cost-efficiency (see Fig. 4.9). Notably, the average end-to-end delay can be affected significantly by the DA2G link congestion.

Finally, we present the runtime comparison of MA-JSPARM and S-JSPARM in Table 4.3. It can be seen that the runtime for long flights is higher than for the short ones. Moreover, Table 4.3 indicates that the time complexity of S-JSPARM is significantly lower than MA-JSPARM, since it does not consider the future flight positions.

**Table 4.3:** MA-JSPARM and S-JSPARM comparison: execution time [s].

| $\delta$ Values | 0.1 | 0.3 | 0.5 | 0.7 | 0.9 |
|---|---|---|---|---|---|
| **MA-JSPARM($\tau_l$)** | 1997.45 | 1936.02 | 3310.87 | 2496.06 | 1697.35 |
| **S-JSPARM($\tau_l$)** | 108.67 | 98.61 | 111.30 | 101.65 | 105.58 |
| **MA-JSPARM($\tau_s$)** | 203.74 | 259.45 | 261.15 | 214.67 | 238.62 |
| **S-JSPARM($\tau_s$)** | 52.35 | 52.32 | 52.41 | 53.35 | 48.51 |

#### 4.7.3.2 Heuristics Comparison

In this part, we compare the performance of the proposed UbU, TbT, and baseline heuristics with respect to the optimal MA-JSPARM.

Let us introduce the baseline algorithm first:

### 4.7.4 Baseline Algorithm

It works based on the UbU, but statically and greedily, without considering the VM placement and routing from previous users. Its description is presented in Alg. 4. Moreover, it refrains itself from VM migration and creates a new VM instance when using the existing VM(s) leads to delay violation.

To evaluate the approaches, we only consider the set of long flights, with the size of 5, 10, 20, and 30. Note that due to the scalability issue of MA-JSPARM, we could not run it for larger than 20 flights. Moreover, the DC locations are set as *Madrid*, *Hamburg*, and *Budapest*. Also, the VM migration costs

**Algorithm 4:** Baseline Heuristic

**Data:** topology $G$, service type $k$, timeslot range $T$, migration cost $C_k^{mig}$
**Result:** VM placement, routing, and migration decisions at each timeslot, total operational cost

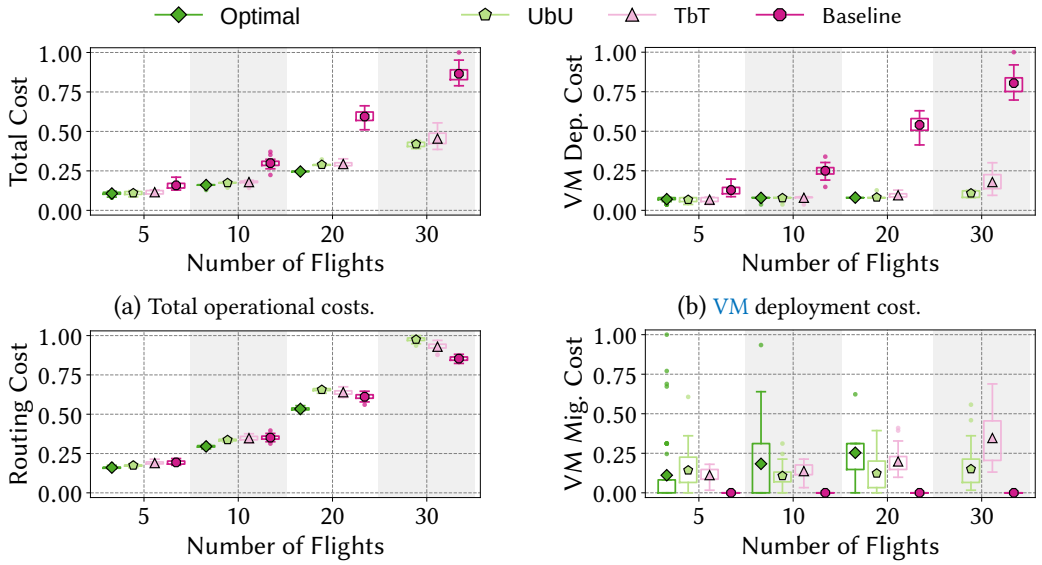
```
for each timeslot t ∈ T do
    Get user location at t;
    if No VM from the last timeslot available then
        Find the DC with the least routing cost;
        Update total operational cost C_total with routing cost and VM cost;
    else
        Get routing latency l to the previous VM;
        if l violates service k requirement then
            Find the DC with the least routing cost;
            Update total operational cost C_total with routing cost and C_k^mig;
        else
            Update total operational cost C_total with routing cost;
```

are fixed based on the service types, and the services have been randomly assigned to each user for every $t \in T$.

The results are summarized in Fig. 4.11. While both UbU and TbT find a solution around 2000x faster than the MA-JSPARM model, as it is illustrated in Fig. 4.11a, they are able to achieve near-optimal results. The total operational cost of TbT increases for a higher number of users, more than the UbU approach. This is mainly due to the VM cost, since TbT does not consider the whole $T$; hence, compared to UbU, the possibility of VM sharing among users is lower, leading to deploying more VMs (Fig. 4.11b). However, since TbT considers all the flights on a single timeslot, the average routing cost is lower than UbU approach since it does not stretch the routing paths over $T$. The difference in routing cost increases by increasing the number of flights in the input.

As depicted in Fig. 4.11c, the routing cost of the baseline approach is lower than the UbU and TbT, because it always tries to place VMs on DCs with the minimum routing delay (which can be translated to minimum #hops, hence, routing cost). Moreover, refusing to reuse the VMs assigned to other flights leads to having a higher VM cost for the baseline, increasing the probability of finding a closer VM (i.e., lower routing cost).

Finally, Fig. 4.11d indicates that the optimal case triggers more VM migrations since it considers the users during the whole $T$ and minimizes the total operational cost. It can be seen that TbT has lower VM migrations compared to UbU for the lower number of flights. Moreover, it increases with the number of flights. In fact, UbU improves the VM sharing for the flights, leading to lower VM migration possibility.




(a) Total operational costs.

(b) VM deployment cost.

(c) Routing cost

(d) VM migration cost.

**Figure 4.11:** Comparison of the UbU, TbT, MA-JSPARM, and the baseline approaches in terms of average total, VM deployment, routing, and VM migration costs (normalized).

## 4.8 Graphical User Interface (GUI)

As an extension, in this section, we present a GUI of our proposed model introduced in this chapter. This tool provides a simple and clear way to demonstrate the optimal routing, VM placement, and migration decisions during the flight in a realistic SAGIN for different scenarios. Fig. 4.12 shows a screenshot of the GUI[1] running on an Apache web server, includes three main parts:

### 4.8.1 GUI Network and Users

The central part depicts a European-based SAGIN map, which contains several components: a European fixed network interconnecting ground core nodes (Cost266 topology [168]), some of them containing DCs [76], DA2G nodes distributed over Europe [159], a representative satellite node and its gateway in Rome, Italy [202]. In this SAGIN, while three airplanes fly over Europe, the optimal routing, VM placement and migration decisions at each $t \in T$ are depicted on the map.

### 4.8.2 Scenarios

Three exemplary scenarios are listed The upper left side of Fig. 4.12. These scenarios are different in three components

1. Number of flights over $T$.
2. Number and location of DCs.
3. Cost values, in particular, VM deployment, and migration costs.

In below, we present the summary of an exemplary output of this GUI.

[1]Video of the demo: https://www.youtube.com/watch?v=gQPpc1q7OxE

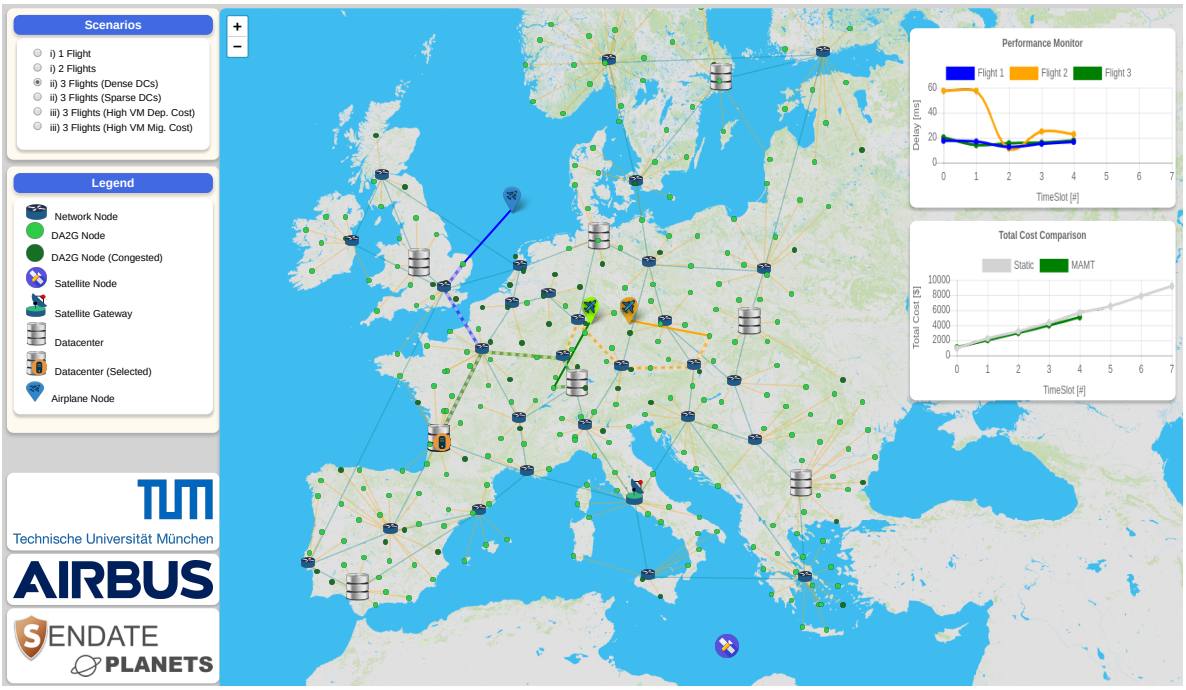



**Figure 4.12:** A screenshot of the proposed GUI.

### 4.8.3 Output

While different airplanes (3 in Fig. 4.12) are flying over the Europe-based SAGIN, the optimal VM placement, migration, and routing decisions are visualized on the GUI. It can be seen how these decisions are dynamically updating over time, according to the mobility of airplanes. Further, a delay monitor is placed on the upper right part of Fig. 4.12. It shows the end-to-end delay for all the flights. Moreover, the total operational cost achieved by the proposed MA-JSPARM model is compared to the static case, i.e., S-JSPARM, in the lower chart in Fig. 4.12.

We believe our proposed optimization models and algorithms introduced in this chapter, along with the GUI tool, can help airline companies to investigate different resource management approaches. They can evaluate their approaches based on the number and path of the flights they offer, their budget, and the service types they would like to provide to the passengers.

## 4.9 Summary

In this chapter, we studied the minimization of the operational costs for the use cases with mobile users. More specifically, considering different access point mediums, we have focused on minimizing the cost of resources such as VMs, network links, and also the cost of reconfigurations such as VM migrations. We have presented a generic system model and formulated two ILPs models for the JSPARM problem: *i)* The static case, S-JSPARM, and *ii)* the mobility-aware/dynamic one, MA-JSPARM. The former considers one single timeslot, while the latter one considers a set of future user positions for solving the JSPARM problem and minimizing the total operational costs.

Further, we have used a realistic case study for the flying airplanes for a European-based SAGIN, which includes the satellite network, the DA2G network, the ground network, as well as the

flights and the distributed DC locations. We demonstrated MA-JSPARM is able to utilize DA2G and satellite connections to satisfy the passenger service requests in a cost-effective manner, compared to S-JSPARM. In addition, we showed a trade-off between routing and migration costs. Also, we showed that a VM migration model could avoid unnecessary migrations and improve the long-term cost-efficiency of MA-JSPARM. Additionally, our evaluation results showed that S-JSPARM achieves higher delay levels and lower runtime compared to MA-JSPARM.

The main issue with the presented optimization models is their scalability. Due to the NP-hardness of the problem, S-JSPARM and MA-JSPARM cannot solve the large problem instances in a tractable time. Therefore, we came up with two heuristic algorithms, which break the problem into smaller sub-problems. On the one hand, we introduced the UbU algorithm to solve the JSPARM problem for each user sequentially over the whole time horizon. On the other hand, we presented the TbT algorithm, which solves the problem at each timeslot (similar to S-JSPARM). The results indicate that while reducing the runtime from hours to seconds, the heuristics can achieve near-optimal solutions.

We believe that the models provided in this chapter can help airlines improve the network planning to support the services of their passengers. Nevertheless, they can tune the inputs of the proposed models (e.g., cost values and service requirements) to fit the models to their particular use case. Also, this chapter can help them to compare the impact of different service providers (e.g., network/satellite providers) on their long-term total operational cost. We note that the proposed resource management algorithms can be applied to other scenarios with mobile users, e.g., MEC, IoT, and IoV (mobility prediction methods might be needed in these cases).

In the next chapter, we focus on delay minimization of users, considering their mobility over a time horizon. Conceptually, it is similar to this chapter; however, it does not consider the cost of resources. Instead, it focuses on minimizing the end-to-end delay for real-time and interactive services to mobile users.

# Chapter 5

# Reconfiguration Due to Mobility: Delay Optimization

## 5.1 Introduction

In the previous chapter, we have focused on minimizing the total deployment cost of a network with mobile users during an infinite number of timeslots (time horizon). The idea was to provide delay-guaranteed services to the mobile users while determining the service placement, user assignment, routing, and reconfigurations with minimum total operational cost. In this chapter, we focus on the same scenario, but with a different goal. We aim to provide services to mobile users with the minimum delay in a best-effort manner. This approach is beneficial for dynamic scenarios with various real-time and interactive services, where guaranteeing a certain delay is not necessary, but minimizing it is the goal. Further, we consider an online/incremental scenario where the users enter the network, and the resources are allocated to them on runtime for their whole activity period. In addition to an online algorithm, we present a second solution based on Machine Learning (ML) to perform the dynamic resource allocation and reconfiguration for the mobile users in an online manner. This chapter is based on the works published in [3] and [18].

Operational cost is an undeniably vital metric to consider when designing a communication system, which was the previous chapter's focus. Given a network, a set of mobile users, and their required services, we needed to make resource allocation decisions. First, the number of Virtual Machines (VMs) of a specific service type to be deployed on a Data Center (DC), given limited DC capacity. Second, assignment of users to VMs, subject VM processing limitation. Third, determination of the routing path from mobile users to the assigned VM considering network capacity. Fourth, when, from which, and to which DC a VM migration must be scheduled. These four decisions were made such that the deployment costs were minimized. The deployment costs were calculated as the sum of the deployed computation resources (i.e., VMs) on distributed DCs, used network links, and migration cost of the VMs to provide delay-guaranteed services to users. In this section, we focus on similar scenario and problem, with a few key differences. Firstly, in this chapter, we relax the placement decision in the problem. Thus, we assume that the services are already deployed on all the DC nodes. In this case, the task of management of the number of deployed VMs can be delegated to auto-scaling mechanisms in cloud orchestration tools, such as OpenStack [174, 142] or Kubernetes [188].

The next difference is the objective function, wherein in this chapter, the aim is to minimize the end-to-end delay between the users and service endpoints (i.e., DCs). The final difference is that the problem is looked at from an incremental/online point of view. In this problem, the mobile users are joining the network in time, and the resources are reserved for them based on the designed algorithm. While in the previous chapter, the users were being solved as a batch, specifically by the Timeslot-by-Timeslot (TbT) algorithm (see section 4.6.2).

### 5.1.1 Key Contributions

This chapter focuses on best-effort service delay minimization in a scenario with mobile users in a time-slotted system. Given a network with distributed DCs, a set of users and their future positions, and the resource limitations, we define the Dynamic Joint Service Assignment, Routing, and Reconfiguration (D-JSARR) problem, which answers the following questions:

1. Assignment: Which users should be assigned to which DC during their activity period?
2. Routing: How the users should reach the assigned DC?
3. Reconfiguration: When, and from/to which DC a user connection should be reconfigured?
4. Objective: How to jointly solve the above questions to minimize the total routing and reconfiguration delays for all the users?

To solve the above problem, first, we formulate it as an Integer Linear Program (ILP) optimization model and prove its NP-hardness. This complexity makes the ILP model not usable in large realistic scenarios. Therefore, we need to develop some fast and efficient solutions, which can solve the problem in runtime. Having that in mind, we propose two heuristic solutions. Firstly, we propose *Homa*, an efficient online algorithm that addresses the in-user service provisioning problem in polynomial time. In *Homa*, we consider a user-by-user approach for solving the problem in online settings. In particular, given a sequence of graphs as input, we reformulate the problem into a dynamic bipartite matching problem with special properties and constraints. Having a sequence of bipartite matchings to be solved, we transform the matching and reconfiguration decisions in an auxiliary graph and use a shortest-path algorithm (e.g., Dijkstra) to solve the problem in a user-by-user fashion. We also show that *Homa* finds the optimal solution in specific scenarios.

Secondly, we propose *Figo*, which is a framework based on ML that solves the above problem in a timely fashion. *Figo* works based on Q-learning method, that after training it with enough data, can solve the D-JSARR problem in an online manner. Further, we evaluate the performance of *Homa* and *Figo* frameworks using simulations with realistic settings. The simulation results show that our approaches can significantly reduce the runtime, performs very closely to the optimal solution obtained from an ILP solver.

### 5.1.2 Organization

The rest of this chapter is organized as follows. State-of-the-art works are discussed in Section 5.2. Then, the problem formulation and optimization modeling of the D-JSARR problem are presented

in Section 5.3 and Section 5.4, respectively. Then, we present the first heuristic framework, *Homa*, and its evaluation in Section 5.5 and Section 5.6. As another solution for the D-JSARR problem, in Section 5.7, we present an ML-based framework, *Figo*. This framework is evaluated in Section 5.8. Finally, the chapter is summarized in Section 5.9.

## 5.2 Related Work

The related work can be discussed in different categories. There are some works that have focused on the mobility-aware service provisioning in Mobile Edge Computing (MEC) environment [200, 218, 215, 153, 97, 102, 95, 199, 35]. In particular, authors in [97, 199] have studied the dynamical placement of network functions on MEC servers according to the future handover probability of users. Similarly, authors in [200] have proposed a VM migration method to deal with user mobility. Further, to be able to support the delay-sensitive applications, Wang et. al. [218] has focused on minimizing the service placement, routing, and migration costs proactively, based on a specific look-ahead window. Also, a closely-related domain to our work is terrestrial networks. [91, 150].

In this area, there are studies that have focused on resource allocation [14, 57, 124, 204, 15, 18], satellite-based vehicular networks [229], reliability [58], and energy-efficiency [191]. For example, Varasteh et al. [14, 15] has presented an offline mobility-aware optimization formulation for the joint service placement, routing, and migration for flying users. Further, in [18], they have extended their work to use reinforcement learning for decision making. Moreover, authors in [124, 204] have focused on the onboard access point node communication analysis. In particular, in [124], the authors have performed a throughput analysis of Direct-Air-to-Ground (DA2G) node communication during its user. Also, [204] has presented a Quality of Service (QoS)-aware approach to satisfy the passengers' traffic flows using different access point node alternatives during the user service duration.

On the other hand, some works have focused on the resource allocation and management of MEC environments [53, 141, 143, 148, 177]. In [53], authors have proposed a graph-based algorithm to determine the number of servers, their size, and the operation area. Considering a maximum resource capacity for the MEC servers, their goal is to serve the maximum number of users at the edge area. Similar to them, the authors in [141] have tackled the problem of edge server placement. They have proposed a multi-objective formulation for balancing the workloads on the MEC servers and reducing the delay between the clients (industrial control centers in their case) and these servers.

A closer area to our work is where the works have considered the resource allocation reconfiguration problem with mobile users [153, 102, 200, 220, 224, 232, 218, 215, 97, 199, 35, 228, 95]. For instance, authors in [232] have proposed a mobility-aware service placement approach, considering the mobility of users and their locations, to minimize the service access delay and deployment costs. Further, some works [97, 199] have studied the dynamical placement of network functions on distributed MEC servers according to the future handover probability of users. Specifically, [97] proposes two service migration solution for MEC-based applications: reactive, and proactive. There are other works with a focus on proactive service provisioning using a look-ahead window and pre-allocation techniques [177, 218]. Also, authors in [200] have proposed a VM migration method, called Follow-Me Cloud, to deal with user mobility. In more detail, the aim of them is always to connect the

mobile users to the optimal gateway, while the service instance (i.e., the VM) follows the users, using the migration technologies. They have proposed an approach based on the Markov decision process to determine the appropriate migration decisions. However, they do not consider the network and servers capacity limits.

In the Internet of Vehicles (IoV) area, similar works exist [27, 226, 228, 34] that provide service continuity to users (vehicles), considering their mobility. For instance, Zhang et al. in [228] have focused on the placement of a constant number of edge servers to improve the QoS (i.e., reducing the average waiting time of services in the IoV), and load balancing. Also, the authors in [27] present a 5G-enabled framework that migrates the services based on the mobility of the vehicles along the road. They keep monitoring the delay between the vehicle and the MEC server. In case that the delay goes higher than a threshold, they migrate the service to another MEC server in the direction of the vehicle movement. However, these works overlook the end-to-end routing, reconfiguration overheads, and resource capacities. However, some works [199, 200] assume that the access points are always available to the users to connect to. Also, some of them overlooked the end-to-end routing decisions [153, 102, 95, 27], DC capacity [27, 102, 153].

Finally, from an algorithmic viewpoint, this chapter is related to the matching problem in graph theory. The matching problem has a long history of research, going back to decades ago [42, 30]. Due to the enormous applications of matching in practice [22], there is a huge body of research for online (bipartite) *static* matching and its variants [157, 79, 46, 133, 162, 137, 51, 134, 139, 134].

However, due to the dynamic nature of many matching problems in practice, recently, dynamic matching has been receiving attentions [39, 192, 48, 66, 119]. Despite the similarity in names, our problem in this paper is different from others. In our problem, there is a sequence of graphs that change dynamically due to the mobility of users. Therefore, the goal is to maintain a minimum-weight bipartite matching, considering the mobility of the users.

## 5.3 Problem Formulation

We present the mathematical formulation of the D-JSARR problem, using the notations summarized in Table 5.1. Let us start the modeling by defining dynamic graphs. Informally speaking, a dynamic graph is a graph that changes over time, either nodes and/or links. The formal definition is given by Definition 1.

**Definition 1.** *A discrete and finite number of timeslots is defined as $T$. We define a dynamic graph as a sequence of $T$ static graphs $G_1 = (N_1, L_1), G_2 = (N_2, L_2), ..., G_T = (N_T, L_T)$.*

The dynamic graph consists of three components: users, core network, and access points. These components are introduced and modeled below.

### 5.3.1 Users

The set of users is defined as $F = \{f_1, f_2, ..., f_{|F|}\}$. At each timeslot, each user $f$ has a particular position over the service area (latitude and longitude), denoted by $p_{f,t}$. Also, each user $f$ is active for duration of $\mathcal{T}_f$ timeslots. Each user location $p_{f,t}$ is a network node in $G_t$. Therefore, we define the set

**Table 5.1:** Notation definition for the D-JSARR problem.

| Parameter | Definition |
|---|---|
| $T$ | Set of timeslots |
| $G_t = (N_t, L_t)$ | Dynamic graph at timeslot $t \in T$, $N$ and $L$ are the sets of nodes and links |
| $\mathcal{N}_{DC}$ | Set of data center nodes |
| $\mathcal{N}_t^F$ | Set of airplane nodes at time $t$ |
| $C_j^{DC}$ | Capacity of DC $j$ |
| $\mathcal{N}_C / \mathcal{L}_C$ | Set of core network nodes and links |
| $\mathcal{N}_{AP} / \mathcal{L}_{AP}$ | The set of nodes/links of the access points. |
| $F$ | Set of mobile users |
| $\mathcal{T}_f$ | The duration of user $f \in F$ |
| $p_{f,t}$ | The position of user $f$ at timeslot $t$ |
| $b_{f,t}$ | The traffic volume of user $f$ at timeslot $t$ |
| $D_{i,j}$ | The delay of link $(i, j) \in L$ |
| $B_{i,j}$ | The bandwidth capacity of link $(i, j) \in L$ |
| $R_{i,j}$ | Reconfiguration delay of assigning an airplane from DC node $i$ to $j$ |
| $\delta^+(i), \delta^-(i)$ | Outgoing and incoming links from and to node $i$ |
| $\mathcal{D}_{routing}^T$ | Total routing delay |
| $\mathcal{D}_{reconf}^T$ | Total reconfiguration delay |
| $\mathcal{S}$ | Matching solution for a dynamic bipartite graph |
| $C(\mathcal{S})$ | Total routing delay of solution $\mathcal{S}$ |
| $\mathcal{R}(\mathcal{S})$ | Total reconfiguration delay of solution $\mathcal{S}$ |
| $d(j)$ | Degree of node $j$ |
| Decision Variables | |
| $x_j^{p_{f,t}} \in \{0, 1\}$ | =1 if user node $p_{f,t}$ is assigned to DC $j$ at timeslot $t$ |
| $l_{u,v,j}^{p_{f,t}} \in \{0, 1\}$ | =1 if link $(u, v)$ is used to route user node $p_{f,t}$ to DC $j$ at timeslot $t$ |
| $r_{i,j}^{p_{f,t}} \in \{0, 1\}$ | =1 if user node $p_{f,t}$ is reassigned from DC $i$ to $j$ at timeslot $t$ |

of user nodes as $\mathcal{N}_t^F = \{p_{f,t} : f \in F, t \in T\}$ and $\mathcal{N}_t^F \subset N_t$. Similar to the previous chapter, a feature of our problem is the prior knowledge of user positions since, in some use cases like airplanes, trains, the user path is pre-determined and typically does not change. Moreover, each user generates an aggregated service traffic with a data rate of $b_{f,t}$ (e.g., in Mbps). These services are considered to be interactive/real-time, such as air-to-ground voice/video calls, gaming, etc., with delay requirements of around 100 ms [89].

### 5.3.2 Core Network

We define $\mathcal{N}_C \subset N_t$ and $\mathcal{L}_C \subset L_t$ as the core network's set of nodes and links, respectively. The user services are hosted by DCs located on this network, and defined as $\mathcal{N}_{DC} \subset \mathcal{N}_C$. A limited amount of resources are rented at each DC. Thus, each DC $j$ can process a maximum traffic volume of $C_j^{DC}$.

### 5.3.3 Access Points

We denote the set of access point nodes as $\mathcal{N}_{AP} \subset N_t$, distributed/located on the service area. These nodes can directly communicate to the users around them and connect them to the core network. Usually, these access points can be a mobile base station or even a satellite connection that relays the traffic to its gateway node on the core network. These access point nodes can be distributed over the network. Each node in $\mathcal{N}_{AP}$ is connected to the closest core network node $\mathcal{N}_C$. Users can connect to an access point node (usually to the closest one). The links between users and access point nodes are denoted as $\mathcal{L}_{AP}$. Each link in $\mathcal{L}_{AP}$ has its constraints, such as communication range, delay, and link capacity.

Finally, we associate a constant propagation delay and a bandwidth capacity to each link $(u, v) \in L_t$, denoted by $D_{u,v}$ and $C^L_{u,v}$, respectively. The dynamicity of $G_t$ comes from the mobility of users and the availability of their access point connections. As shown in Fig. 5.1, the availability of mobile base stations around the user position can change due to a congestion model [14], hence the available links of the graph $G_t$ in a particular timeslot.

In the following, we introduce the formulation of the ILP optimization problem. Then, we prove its NP-Hardness, and hence, we propose a heuristic framework to solve the problem.

## 5.4 Integer Linear Program (ILP) Formulation

In this section, we present the offline optimization formulation as an ILP model. Let us start with two delay functions. We use these functions later to form the objective function of the model.

### 5.4.1 Routing Delay

The first function is the routing delay. This delay comes from the path that the traffic from a user takes to reach the assigned DC for receiving its services. Therefore, $\mathcal{D}^T_{routing}$ is the defined sum of the propagation delay of the links that the user traverses to reach the assigned DC in all timeslots:

$$\mathcal{D}^T_{routing} = \sum_{t \in T} \sum_{f \in F} \sum_{j \in \mathcal{N}_{DC}} \sum_{(u,v) \in L_t} l^{p_{f,t}}_{u,v,j} D_{u,v} \tag{5.1}$$

where $l^{p_{f,t}}_{u,v,j} \in \{0, 1\}$ is a decision variable that is equal to 1 if the traffic of user node $p_{f,t}$ is routed towards DC $j$ using the link $(u, v)$ at timeslot $t$.

### 5.4.2 Reconfiguration Delay

As mentioned before, the users are mobile in our scenario. When they move, their current DC selection might not remain the best in the future. Therefore, to keep the delay as low as possible, we need to reconfigure the user from the previous DC to a closer one. With this decision, the user will have a lower routing delay to the new assigned DC, which improves the QoS. The reassignment[1] of a user connection from a DC to another one is realized by synchronization of the user state and data migration between two DCs. Due to the traffic overhead and effort, user reconfigurations come with

[1] In this chapter, reconfiguring/reconfiguration and reassigning/reassignment terms are used interchangeably

a delay penalty. Therefore, the total reconfiguration delay for all users over the whole time horizon $T$ is denoted by $\mathcal{D}^{T}_{reconf}$, and can be calculated as follows:

$$\mathcal{D}^{T}_{reconf} = \sum_{t \in T} \sum_{f \in F} \sum_{(i,j) \in \mathcal{N}_{DC} \times \mathcal{N}_{DC}} r^{p_{f,t}}_{i,j} R_{i,j} \tag{5.2}$$

where $r^{p_{f,t}}_{i,j} \in \{0,1\}$ is a decision variable that is equal to 1 if the connection of user node $p_{f,t}$ is reassigned from DC node $i$ to $j$ at timeslot $t$. Also, $R_{i,j}$ represents the delay of reconfiguring a user connection from DC $i$ to $j$.

### 5.4.3 Constraints

Let us now define the constraints that need to be met by the model. Note that these constraints are related to the ones presented in previous chapter, Sections 4.4 and 4.5.

#### 5.4.3.1 User-to-DC Assignment

Assuming that flows are not splittable, so each user must be assigned to one and only one DC at each timeslot $t$:

$$\sum_{j \in \mathcal{N}_{DC}} x^{p_{f,t}}_{j} = 1, \forall t \in T, \forall f \in F, \tag{5.3}$$

where $x^{p_{f,t}}_{j} \in \{0,1\}$ is a binary decision variable which is equal to 1, if the user node $p_{f,t}$ is assigned to DC $j$ at timeslot $t$.

#### 5.4.3.2 Traffic Generation

The traffic should be generated from the user nodes towards a neighbor node in the graph, i.e., access point node(s):

$$\sum_{v \in \delta^{+}(p_{f,t})} \sum_{j \in \mathcal{N}_{DC}} l^{p_{f,t}}_{p_{f,t},v,j} = 1, \forall t \in T, \forall f \in F. \tag{5.4}$$

where $l^{p_{f,t}}_{p_{f,t},v,j}$ is a binary decision variable which is equal to 1, if the user position at timeslot $t$ (i.e., $p_{f,t}$) uses link $p_{f,t}, u$ to connect to DC $j$. Also, $\delta^{+}(i)/\delta^{-}(i)$ is a function that returns the outgoing/incoming links from/to node $i$.

#### 5.4.3.3 Flow Conservation

These constraints route the traffic flow through the network towards the assigned DC:

$$\sum_{v \in \delta^{+}(u)} \sum_{j \in \mathcal{N}_{DC}} l^{p_{f,t}}_{p_{f,t},v,j} - \sum_{v \in \delta^{-}(u)} \sum_{j \in \mathcal{N}_{DC}} l^{p_{f,t}}_{p_{f,t},v,j} = \begin{cases} 0, \forall t \in T, \forall f \in F, \forall u \in N_t \backslash \mathcal{N}_{DC}, u \neq p_{f,t} \\ x^{p_{f,t}}_{u}, \forall t \in T, \forall f \in F, \forall u \in \mathcal{N}_{DC} \end{cases}. \tag{5.5}$$

In the above set of constraints, the network node that receives the traffic will forward it out, unless it is a DC. This is the standard flow conservation rule that we apply in our model.

#### 5.4.3.4 Routing/Assignment Relation

We need to ensure that the traffic flow of each user node at a particular timeslot is forwarded towards the assigned DC:

$$l_{u,v,j}^{p_{f,t}} \leq x_j^{p_{f,t}}, \forall (u,v) \in L_t, \forall t \in T, \forall f \in F, \forall j \in \mathcal{N}_{DC}, \tag{5.6}$$

In the above constraints, the traffic of the user node $p_{f,t}$ can be routed towards DC $j$, if and only if the user node is assigned to it at timeslot $t$ (i.e., $x_j^{p_{f,t}} = 1$). Thus, the above constraint can prevent the traffic from being forwarded to incorrect DC nodes.

#### 5.4.3.5 DC Capacity

As mentioned before, the amount of user traffic that is being processed by each DC is limited. Therefore, we limit each DC to be able to process a specific amount of traffic:

$$\sum_{f \in F} x_j^{p_{f,t}} b_{f,t} \leq C_j^{DC}, \forall t \in T, \forall j \in \mathcal{N}_{DC}, \tag{5.7}$$

In above, it is ensured that the sum of the assigned user traffic to each DC stays below the DC capacity limit $C_j^{DC}$.

#### 5.4.3.6 Network Link Capacity

The amount of user traffic volume delivered on a network link is limited to the link's capacity $B_{u,v}$. To express this, we define the following set of constraints:

$$\sum_{f \in F} \sum_{j \in \mathcal{N}_{DC}} l_{u,v,j}^{p_{f,t}} b_{f,t} \leq B_{u,v}, \forall (u,v) \in L_t, \forall t \in T. \tag{5.8}$$

#### 5.4.3.7 Reconfigurations

The final set of constraints belong to the reconfigurations that have to be performed:

$$x_j^{p_{f,t}} = \sum_{i \in \mathcal{N}_{DC}} r_{i,j}^{p_{f,t}}, \forall t \in T \backslash \{0\}, \forall f \in F, \forall j \in \mathcal{N}_{DC}, \tag{5.9}$$

$$x_j^{p_{f,t}} = \sum_{i \in \mathcal{N}_{DC}} r_{j,i}^{p_{f,t+1}}, \forall t \in T (t \neq T), \forall f \in F, \forall j \in \mathcal{N}_{DC}. \tag{5.10}$$

Constraints (5.9) and (5.10) link $x$ and $r$ variables in a flow conservation way. If $x_j^{p_{f,t}} = 0$, the user node $p_{f,t}$ is not assigned to DC $j$ at timeslot $t$, meaning no reconfiguration is happened for the user. $x_j^{p_{f,t}} = 1$ implies that a reconfiguration assigns the user node $p_{f,t}$ to DC $j$ at timeslot $t$ and reassigns it at $t+1$. This reassignment can be from the DC $j$ to $j$, which incurs zero reconfiguration delay: $\forall i, j \in (J \times J)$, if $i = j$, $R_{i,j} = 0$. We note that reconfiguration is not possible at the first timeslot.

Finally, the ILP formulation can be presented as follows. The objective function aims at minimizing the sum of routing and reconfiguration delays for all the users over the whole time horizon $T$.

$$\begin{aligned} \text{Minimize } & \left(\mathcal{D}^{T}_{routing} + \mathcal{D}^{T}_{reconf}\right), \\ \text{s.t. Constraints } & (5.3) - (5.10), \\ \text{vars: } & x_j^{p_f,t} \in \{0,1\}, \forall t \in T, \forall F \in F, \forall j \in \mathcal{N}_{DC}, \\ & l^{p_f,t}_{p_f,t,v,j} \in \{0,1\}, \forall t \in T, \forall f \in F, \forall j \in \mathcal{N}_{DC}, \forall (u,v) \in L_t, \\ & r^{p_f,t}_{i,j} \in \{0,1\}, \forall t \in T, \forall f \in F, \forall i,j \in \mathcal{N}_{DC}. \end{aligned} \tag{5.11}$$

**Theorem 1.** *The optimization model (5.11) is NP-hard.*

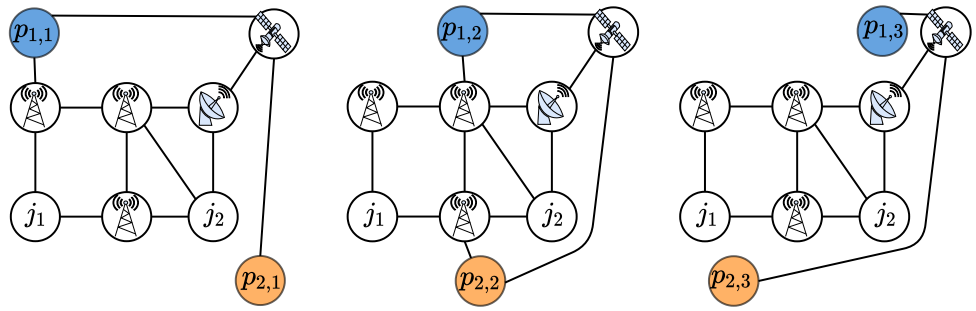
*Proof.* Let us begin by defining the Generalized Assignment Problem (GAP). According to [60, 100], GAP determines the minimum-cost assignment of $n$ jobs to $m$ agents, such that considering the capacity restrictions on the agents, each job is assigned to exactly one agent. Therefore, we are able to reduce from GAP to an instance of model (5.11) with $T = 1$:

$$\begin{aligned} \text{Minimize } & \sum_{f \in F} \sum_{j \in \mathcal{N}_{DC}} c_{p_f,j} x_j^{p_f}, \\ \text{s.t. } & \sum_{j \in \mathcal{N}_{DC}} x_j^{p_f} = 1, \forall f \in F, \\ & \sum_{f \in F} x_j^{p_f} b_f \leq C_j^{DC}, \forall j \in \mathcal{N}_{DC}, \\ \text{vars: } & x_j^{p_f} \in \{0,1\}, \forall f \in F, \forall j \in \mathcal{N}_{DC}, \end{aligned} \tag{5.12}$$

where $c_{p_f,j}$ is the assignment cost of user node (job) $p_f$ to DC (agent) $j$ (e.g., the shortest-path delay value). This transformation can be clearly done in polynomial time. If there exists an algorithm that solves the model (5.11), it solves the corresponding GAP as well. Given the NP-hardness of GAP [60], the model (5.11) must be NP-hard too. □

## 5.5 *Homa* Heuristic Framework

In this section, we present *Homa*, our first heuristic solution for the D-JSARR problem. *Homa* is a deterministic online approach which solves the D-JSARR problem in polynomial time. The proposed heuristic is formulated as Dynamic Constrained Minimum-Weight Bipartite Matching (DC-MWBM) problem, which is explained in the following subsection. In this context, a bipartite matching should be maintained (considering the reconfiguration cost) for a sequence of graphs that may change over time dynamically. In the next step, we solve the DC-MWBM in a user-by-user manner where a set of user requests for the service provisioning during their user time. To solve the problem for a user, we once again transform the problem into a shortest-path routing problem, where using an auxiliary graph, the user-to-DC assignments, routing, and reconfiguration decisions are taken. These steps are explained in the following.




(a) $G_1$ ($t = 1$) (b) $G_2$ ($t = 2$) (c) $G_3$ ($t = 3$)

**Figure 5.1:** An example of the problem input as a dynamic graph with two users, two DC nodes, and three timeslots in a scenario with two exemplary access point types: base stations and satellite. With the change of user positions during the time, the available connection links might change. For example, user 2 at $t = 1$, (i.e., node $p_{2,1}$) is connected only to the satellite, while at $t = 2$ (i.e., node $p_{2,2}$), it gets into a base station range, thus, both access point options are accessible.

### 5.5.1 Dynamic Constrained Minimum-Weight Bipartite Matching (DC-MWBM)

Given the sequence of graphs $G_t$ (see Fig. 5.1), we transform each $G_t$ to a bipartite graph $\mathcal{G}_t$. We define bipartite graph $\mathcal{G}_t = (\mathcal{N}_t^F, \mathcal{N}_{DC}, L_t)$. The set of links is defined as

$$L_t = \{(f, j, n) : f \in \mathcal{N}_t^F, j \in \mathcal{N}_{DC}, n \in \delta^+(p_{f,t})\}$$

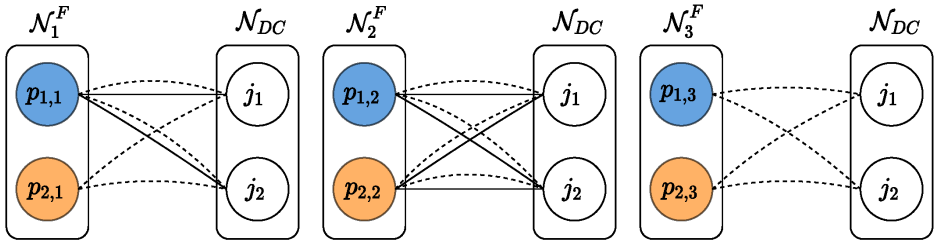
which shows a matching from user node $p_{f,t}$ to DC $j$ through available access point node $n$. Each $(f, j, n) \in L_t$ is associated with a weight function $w(f, j, n) : L_t \rightarrow R^+$ which is defined as the routing delay between $f \in \mathcal{N}_t^F$ and $j \in \mathcal{N}_{DC}$ through the access point node $n$ (routing is performed in $G_t$). Since the position of users changes per timeslot, the link weights from the $w$ function also change, i.e., due to the users' mobility, the routing delay between the user nodes, and DCs changes. Considering that, the total number of links for each $\mathcal{G}_t$ can be at most $2 \cdot \mid \mathcal{N}_t^F \mid \cdot \mid \mathcal{N}_{DC} \mid$. For clarification, Fig. 5.2 shows the bipartite graphs created for the scenario in Fig. 5.1. In this example, it is assumed that $\forall f \in F, \mathcal{T}_f = T = 3$.

Having the sequence of bipartite graphs $\mathcal{G}_t$, we define the offline DC-MWBM problem. In this problem, we need to select a set of edges $\mathcal{M}_t \subset L_t$ such that each DC node $j$ have a node degree at most equal to its capacity $C_j^{DC}$. For simplification purposes, we consider the DC capacity as the maximum number of users that can use the DC resources at the same time.

$$d(j) \leq C_j^{DC}, \forall j \in \mathcal{N}_{DC}, \tag{5.13}$$

where $d(j)$ indicates the node degree of node $j$. Also, since we focus on the best-effort delay minimization, we relax the bandwidth constraints for now. However, we later show that in the online solution, the bandwidth constraint can be addressed by removing the link with not enough capacity before solving a problem instance.

**Remark 1.** *Considering Hall's theorem [114], $\mathcal{G}_t$ can have a matching with cardinality $\mid \mathcal{N}_t^F \mid$. Although $\mid \mathcal{N}_t^F \mid > \mid \mathcal{N}_{DC} \mid$ in $\mathcal{G}_t$; however, by making $C_j^{DC}$ copies of nodes $j \in \mathcal{N}_{DC}$ in $\mathcal{G}_t$, we have $\forall U \subseteq \mathcal{N}_t^F, \mid U \mid \leq \mid \delta^+(U) \mid$ (where $\delta^+(U)$ is the set of DC nodes, i.e., copied nodes, that users can connect to them). This condition is true by having the lower bound value for the DC capacity as $\forall j \in \mathcal{N}_{DC}, C_j^{DC} = \left\lceil \frac{|F|}{|\mathcal{N}_{DC}|} \right\rceil$ (and thus a feasible solution).*




(a) $\mathcal{G}_1$ (b) $\mathcal{G}_2$ (c) $\mathcal{G}_3$

**Figure 5.2:** The sequence of bipartite graphs $\mathcal{G}_t$ based on the example in Fig. 5.1. At each timeslot $t$, user nodes $p_{f,t}$ can be matched to a DC node $j$ through an access point node if available: satellite (dashed lines) or DA2G node (solid lines). For example, at the $t = 1$, user 1 has both base station access point node and satellite connection, whereas user 2 has only access to the satellite. At $t = 2$, both users have access to both access point nodes, while at $t = 3$, only have satellite connection available. Also, at each timeslot $t$, due to the mobility of users, the link weight (user-to-DC routing delay) changes dynamically.

Hence, we ensure that all the user nodes in $\mathcal{G}_t$ are matched (connected) to a DC node, i.e.,

$$d(p_{f,t}) = 1, \forall p_{f,t} \in \mathcal{N}_t^F, t \in T. \tag{5.14}$$

Having a sequence of dynamic bipartite graphs $\mathcal{G}_t$, the solution of the offline DC-MWBM problem is a sequence of matchings $\mathcal{S} = \mathcal{M}_1, \mathcal{M}_2, ..., \mathcal{M}_T$, where $\mathcal{M}_t$ is the matching of bipartite graph $\mathcal{G}_t$. Similar to the objective function in ILP model (5.11), the cost of solution $\mathcal{S}$ can be defined as follows:

#### 5.5.1.1 Matching Cost

The total matching cost of a solution $\mathcal{S}$ reflects the routing delay value and is the sum of matchings (selected links in bipartite graphs) in all timeslots:

$$C(\mathcal{S}) = \sum_{t \in T} \sum_{l \in \mathcal{M}_t} w(l), \tag{5.15}$$

where $l = (f, j, n)$ in $\mathcal{G}_t$. We remind that $C(\mathcal{S})$ is the sum for the routing delay of all the users to the matched DC node in all timeslots.

#### 5.5.1.2 Reconfiguration Cost

We define a per-link reconfiguration delay to model the cost of matching changes (i.e., reconfiguration delay). We denote the total reconfiguration cost of solution $\mathcal{S}$ as $\mathcal{R}(\mathcal{S})$, defined as:

$$\mathcal{R}(\mathcal{S}) = \sum_{t \in T \setminus 0} \sum_{(f,j) \in \mathcal{M}_t} \sum_{(f',j') \in \mathcal{M}_{t-1}, (f=f', j \neq j')} R_{j',j}, \tag{5.16}$$

where $R_{j',j}$ is the reconfiguration delay of reassigning a user connection from DC node $j'$ to $j$ in two consecutive timeslots. We note that reconfigurations are not possible at first timeslot.

Having the cost functions in hand, our goal is to calculate $\mathcal{S}$ for the sequence of $\mathcal{M}_1, \mathcal{M}_2, ..., \mathcal{M}_T$, minimizing the total matching and reassignment delays:

$$\text{Minimize } \big(C(\mathcal{S}) + \mathcal{R}(\mathcal{S})\big). \tag{5.17}$$

Similar to Theorem 1, the DC-MWBM can be proved to be NP-hard, which is omitted due to the lack of space.

### 5.5.2 Solving DC-MWBM

In this subsection, we develop a deterministic online approach to solve the DC-MWBM problem. At this stage, a request of service provisioning for a set of users is given as an instance of DC-MWBM. To solve this matching problem, *Homa* is designed to work in a user-by-user manner (similar to the approach presented in Section 4.6.1). *Homa* uses the user location information (i.e., the user locations during its active time) to schedule the DC connections and reconfigurations for the whole user period. Therefore, given a user $f$ with duration $\mathcal{T}_f$, we first build an auxiliary graph $\mathcal{G}_A$. We then introduce and solve a shortest-path routing problem in $\mathcal{G}_A$, which gives the solution of the online DC-MWBM. The description of the proposed algorithm is aligned in Alg. 5. The input of the algorithm is a single user $f$ with the activity duration $\mathcal{T}_f$, for which we find the user-to-DC assignment and the necessary reconfigurations at each timeslot $t \in \mathcal{T}_f$. Also, to keep track of DC capacities, a dictionary with key-value pairs $\left(j, (t, C_j^{DC})\right), j \in \mathcal{N}_{DC}, t \in T$ is calculated and given using the network status from previously served users. In line 1, we initialize the sequence of bipartite graphs $\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_{\mathcal{T}_f}$ for the given user $f$. In line 2, we create the auxiliary graph $\mathcal{G}_A$ as a multigraph. We create a dictionary in line 3 to keep the DC nodes with enough capacity at each timeslot (line 5). After that, we start to build $\mathcal{G}_A$. Note that in this example (see Fig. 5.1 and Fig. 5.2), two access points are considered, base stations and satellites, which can be used to connect the users to the core network. Fig. 5.3 is the follow up for the example in Fig. 5.2 and it shows how Alg. 5 converts DC-MWBM into a shortest-path problem and solves it.

#### 5.5.2.1 Case 1, $t = 1, t \in \mathcal{T}_f$

This is the case for the first timeslot of the user $f$. In this case, we first add the user node $p_{f,t}$ to $\mathcal{G}_A$ as the first node (line 8). Then, we create a node for each available DC node (line 10), and connect the user node $p_{f,t}$ node at $t = 1$ to it (lines 11-15). These links represent an assignment decision, for instance link $(p_{f,1}, j_1)$ means user node $f$ is assigned to DC $j_1$ at timeslot $t = 1$. As mentioned before, each user node can connect to a DC using one of the access point node connections (i.e., satellite or base station), which is determined in line 6. Therefore, for each available access point node (line 11), and add a link for each of them (line 15). For example, in Fig. 5.1a, it can be seen that at $t = 1$, user $f = 2$ has only satellite connection. Therefore, in Fig. 5.3b at $t = 1$, we connect the user node $p_{2,1}$ to the DC nodes only using the satellite connection (dashed line). We define a weight for each of these links, which will be used later to calculate the shortest path. These weights are defined as the delay of the shortest path between the user and DC nodes (in original graph $G_t$) through the corresponding access point node connection. Note that the links with lower than $b_{f,t}$ available capacity are removed from the network before calculating the shortest path. Therefore, in lines 12 and 13, we calculate the shortest-path delay from the user node $p_{f,t}$ to the neighbor node $n$ (i.e., either satellite or base station access points), and from $n$ to the DC node, respectively. We then sum up these two delay values (line 14) and finally create a link between the user and DC node with the total delay value as its weight (line 15).

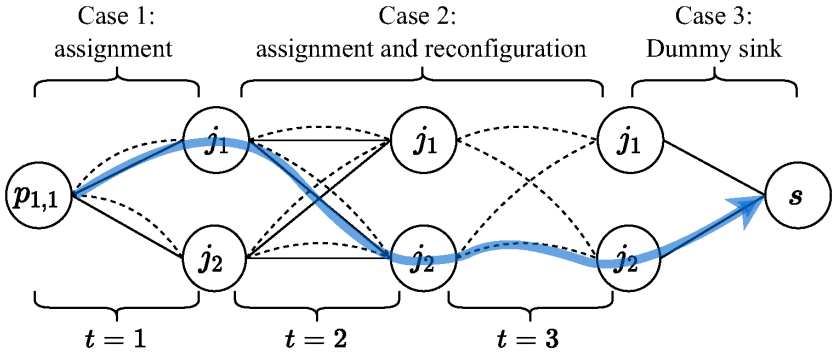
**Algorithm 5:** *Homa* Algorithm

**Input :** user $f$, $\mathcal{T}_f$, $t \in T$, dc_capacity_dic
**Output:** Solution $\mathcal{S}$

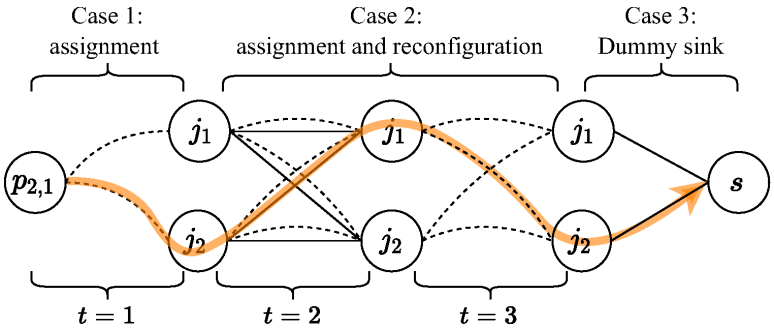
1 initialize $\mathcal{G}_t, \forall t \in \mathcal{T}_f$ ;
2 $\mathcal{G}_A$ = new MultiGraph() ;
3 available_dcs = {} ;
4 **foreach** $t \in \mathcal{T}_f$ **do**
5  available_dcs[t] = get_available_dcs(dc_capacity_dic[t]);
6  user_access_point_nodes = $G_t$.get_node_neighbors($p_{f,t}$);
  /* Case 1 */
7  **if** $t == 1$ **then**
8  $\mathcal{G}_A$.add_node($p_{f,t}$) ;
9  **foreach** $j_t \in available\_dcs[t]$ **do**
10  $\mathcal{G}_A$.add_node($j_t$);
11  **foreach** $n \in user\_access\_point\_nodes$ **do**
12  delay1 = $G_t$.dijkstra_path_length($p_{f,t}$, $n$) ;
13  delay2 = $G_t$.dijkstra_path_length($n$, $j_t$);
14  sum_delay = delay1 + delay2 ;
15  $\mathcal{G}_A$.add_link($p_{f,t}$, $j_t$, $sum\_delay$);
16  **else**
  /* Case 2 */
17  **foreach** $j'_{t-1} \in available\_dcs[t-1]$ **do**
18  **foreach** $j_t \in available\_dcs[t]$ **do**
19  $\mathcal{G}_A$.add_node($j_t$);
20  **foreach** $n \in user\_access\_point\_nodes$ **do**
21  delay1 = $G_t$.dijkstra_path_length($p_{f,t}$, $n$);
22  delay2 = $G_t$.dijkstra_path_length($n$, $j_t$);
23  sum_delay = delay1 + delay2;
24  **if** $j'_{t-1} \neq j_t$ **then**
25  sum_delay += $R_{j'_{t-1},j_t}$;
26  $\mathcal{G}_A$.add_link($j'_{t-1}$, $j_t$, $sum\_delay$);
  /* Case 3 */
27 $\mathcal{G}_A$.add_node($s$);
28 **foreach** $j_t \in available\_dcs[\mathcal{T}_f - 1]$ **do**
29  $\mathcal{G}_A$.add_link($j_t$, $s$, 0);
30 $\mathcal{S}$ = $\mathcal{G}_A$.dijkstra_path($p_{f,0}$, $s$);
31 update_capacity(dc_capacity_dic, $\mathcal{S}$));
32 **return** $\mathcal{S}$;

#### 5.5.2.2 Case 2, $t > 1, t \in \mathcal{T}_f$

In this case, we continue building $\mathcal{G}_A$ by connecting DC nodes in two consecutive timeslots. We create a link between the DC nodes in $t-1$ and $t, \forall 1 < t \leq \mathcal{T}_f$ (lines 17-26). The link $(i_{t-1}, j_t)$ which connected DCs $i$ and $j$ in $\mathcal{G}_A$, indicates that in two consecutive timeslots $t-1$ and $t$, the user nodes $p_{f,t-1}$ and $p_{f,t}$ are assigned to DC nodes $i$ and $j$, respectively. Compared to the first case and the assignment decisions, the possibility of reconfigurations is enabled here. When $i \neq j$, the link represents a reconfiguration from DC $i$ to $j$ at timeslot $t$. We loop through the previous DC nodes




(a) $\mathcal{G}_A$ for the exemplary user $f = 1$.




(b) $\mathcal{G}_A$ for the exemplary user $f = 2$.

**Figure 5.3:** Building the auxiliary graph $\mathcal{G}_A$, and solving the DC-MWBM problem for the scenario in Figs. 5.1 and 5.2 in a user-by-user manner. (a) The auxiliary graph $\mathcal{G}_A$ and solution for $f = 1$ is built with the possible assignment and reconfiguration decisions through the user time. It can be seen that at $t = 3$, only satellite connection is available (dashed line). The calculated shortest-path from $p_1^1$ to $s$ is $p_1^1 \dashrightarrow j_1 \rightarrow j_2 \dashrightarrow j_2 \rightarrow s$. This path means at $t = 1$, the user node $p_1^1$ is connected to DC $j_1$ through DA2G (the one in range), at $t = 2$ the user node $p_2^1$ is connected to $j_2$ using base station node (one reconfiguration from $j_1$ to $j_2$), and at $t = 3$, the user node $p_3^1$ continues connecting to DC $j_2$ through satellite. (b) The calculated shortest-path here is $p_1^2 \dashrightarrow j_2 \rightarrow j_1 \dashrightarrow j_2 \rightarrow s$, which uses satellite, base station node, and satellite to connect to DCs $j_2$, $j_1$, and $j_2$ at timeslots $t = 1$, $t = 2$, and $t = 3$, respectively. For this user, two reconfiguration at $t = 2$ and $t = 3$ between DC $j_2 - j_1$, and $j_1 - j_2$ are determined, respectively. We note that the reconfiguration between base station access point and satellite connection inside the user can be done with an almost instant routing table update using e.g., OpenFlow [11].

at $t - 1$ (line 17) and also at $t$ (line 18). We create a node in $\mathcal{G}_A$ for all the available DC nodes $j_t$ (line 19). Similar to the first case, a maximum of two links can exist between nodes (per access point node connection type). We calculate the delay of the shortest-path between $p_{f,t}$ and DC $t$ through the neighbor node (i.e., access point node connections) $n$ in $G_t$. In case that two DC nodes at $t - 1$ and $t$ are not the same node in $G_t$, it indicates a reconfiguration decision. Therefore, we add the reconfiguration delay between two DC nodes to the link weight (line 25). Finally, in line 26, we add a link to $\mathcal{G}_A$ from node $j_{t-1}$ to $j_t$ with the summed routing delay as the weight.

#### 5.5.2.3 Case 3, Dummy sink

This is the case where we add a dummy sink node $s$ to $\mathcal{G}_A$ in line 27. Thereafter, we connect the DC nodes in timeslot $t = \mathcal{T}_f$ (last user timeslot) to $s$ with weight 0 (line 29).

Having the auxiliary graph $\mathcal{G}_A$ built, we can calculate the solution for the user $f$. To do this, we need to calculate the shortest path from $p_{f,1}$ and the dummy sink node $s$ nodes using conventional algorithms, e.g., Dijkstra (line 30). The returned path gives us the assignment, routing, and reconfiguration decisions with minimum total delay. A comprehensive example is presented in Fig. 5.3. Finally, the DC and network link capacities are updated in line 31 to be used for future users.

**Remark 2.** *Algorithm 5 can solve the problem for users with any duration. In particular, we only need to keep track of DC capacities per timeslot. That is, the $\mathcal{G}_A$ can be formed for a user with any duration. Clearly, the number of nodes and links for $\mathcal{G}_A$ would depend on the user duration.*

**Remark 3.** *For a given user $f$ with duration $\mathcal{T}_f$ timeslots, graph $\mathcal{G}_A$ can have $O(\mathcal{T}_f \cdot \mid \mathcal{N}_{DC} \mid)$ nodes and $O(\mathcal{T}_f \cdot \mid \mathcal{N}_{DC} \mid^2)$ links.*

**Lemma 2.** *For any given user $f$ with duration $\mathcal{T}_f$, Homa is complete[2] and optimal.*

*Proof.* For a given user $f$, we build the auxiliary graph $\mathcal{G}_A$ with the available DC nodes. Also, we add all the possible routing options (i.e., access points) with the weight defined as the shortest-path delay between the user and DC. Also, we consider all possible assignments and reconfiguration decisions in $\mathcal{G}_A$. Therefore, since Dijkstra is complete and optimal [81], for the given user $f$, *Homa* is also complete and optimal. □

**Theorem 3.** *For any given set of users $F$, if $\forall j \in \mathcal{N}_{DC}, C_j^{DC} \geq \mid F \mid$, Homa is complete and optimal.*

*Proof.* users in $F$ depend on each other through the DC capacity. If $\forall j \in \mathcal{N}_{DC}, C_j^{DC} \geq \mid F \mid$, it simply means at each timeslot, a single DC node can host all the users. Thus, the DC capacity constraints can be relaxed in this case. Therefore, according to the Lemma 2, *Homa* is complete and optimal for all the set of user $F$. □

**Theorem 4.** *For any given user $f$, the complexity of Homa for finding the optimal solution is $O(\mathcal{T}_f \cdot |N_t|^2)$.*

*Proof.* We calculate the runtime based on the number of times the Dijkstra algorithm needs to be performed to find the solution for user $f$. The Dijkstra algorithm needs to be executed for each link in the auxiliary graph $\mathcal{G}_A$, except the links for the dummy destination node $s$. Therefore, considering Remark 3, we need $2 \cdot (\mathcal{T}_f - 1) \cdot \mid \mathcal{N}_{DC} \mid^2 + \mid \mathcal{N}_{DC} \mid$ links (i.e., Dijkstra runs) to build the $\mathcal{G}_A$. Each Dijkstra run is executed for $G_t = (N_t, L_t)$, with runtime $O(\mid N_t \mid^2)$. Since $\mid \mathcal{N}_{DC} \mid \ll \mid N_t \mid$, the overall algorithm runtime for any user $f$ can be expressed as $O(\mathcal{T}_f \cdot |N_t|^2)$. We note that in use cases that the duration of the timeslots is minimal (close to 0), the algorithm complexity is pseudo-polynomial (based on $\mathcal{T}_f$). □

## 5.6 Performance Evaluation

This part presents the performance evaluation of *Homa* in various scenarios and settings. We first introduce the two baseline algorithms, which are two greedy variations of *Homa*. After that, we explain

[2] An algorithm is complete if it always finds a solution, if one exists. The completeness does not imply optimality.

the simulation setup and scenarios. Finally, we present the results of the performance evaluation of *Homa* against the optimal offline solution (Section 5.4) and the two baseline algorithms.

### 5.6.1 Algorithms to Compare

For comparison purposes, we develop two variations of *Homa* as the baseline algorithms. For comparison purposes, we re-implemented the algorithm presented in the state-of-the-art work [27]. We name this algorithm as Threshold-Based Reconfiguration (TBR). Further, we develop two variations of *Homa* (Minimize Delay (MD), and Maintain Connection (MC)) as the baseline algorithms. In below, we explain these approaches in short.

#### 5.6.1.1 Threshold-Based Reconfiguration (TBR)

The first algorithm to compare with *Homa* is taken from the state-of-the-art work [27]. In [27], the authors propose a resource management framework for service placement and migration for the automotive use-case within the MEC scenario. Similar to us, they have focused on providing low-delay services to the mobile users (cars in their case) while determining the appropriate MEC server to host the services and the necessary reconfigurations according to the mobility of the vehicles. In their approach, they set a threshold for which, if the current routing delay value is higher than a threshold value, the service is migrated to another MEC server. This MEC server (destination of the migration) is chosen as the farthest one to lower the number of future migrations. For comparison purposes, we have chosen two threshold values, 20 and 40 ms. Therefore, we show these two algorithm versions as TBR-20 and TBR-40, respectively.

However, in contrast to us, the approach presented in [27] lacks considering a few constraints. Firstly, they do not consider the DC and link capacities into the account. Secondly, they ignore the service migration (reconfiguration) penalty in their algorithm. Nevertheless, for a fair comparison, we have added these considerations into the implementation of their algorithm.

#### 5.6.1.2 Minimize Delay (MD)

The next algorithm is a baseline that focuses on minimizing the routing delay between the user nodes and the assigned DC. MD reconfigures the user connections whenever it leads to a lower routing delay. At $t = 1$, it chooses the DC with the lowest routing delay to connect the user to it. For $t > 1$, it reconfigures the user connection to a DC which keeps the routing delay at that specific timeslot minimized. Regarding the implementation, it is enough to set $\forall i, j \in \mathcal{N}_{DC}, R_{i,j} = 0$, i.e., reconfiguration becomes a decision with no cost. Thus, in the procedure of building $\mathcal{G}_A$, in the case 2 where $t > 1$, we update the weight of link $(i, j), i, j \in \mathcal{N}_{DC}$ with zero reconfiguration cost, i.e., changing line 25 to $sum\_delay$ += 0 in Algorithm 5.

#### 5.6.1.3 Maintain Connection (MC)

The last baseline algorithm is called MC. As it appears from its name, this algorithm always maintains its connection to the first selected DC. Developing this algorithm aims to highlight the importance of providing flexibility in the network (i.e., reconfigurations). At the first timeslot of the flight, MC

connects the user to the DC with minimum routing delay and keeps using it for the rest of the flight duration. The implementation is done with a slight tweak in Algorithm 5, in the second case 2, where $t > 1$. In particular, while building the $\mathcal{G}_A$, in two consecutive timeslots $t-1$ and $t$, MC adds links only between DC nodes $i_{t-1}$ and $j_t$, if $i = j, i, j \in \mathcal{N}_{DC}$. In this way, the reconfiguration of user connections would not be possible for the Algorithm (since there are no links between different DCs in $\mathcal{G}_A$).

### 5.6.2 Simulation Setup

In this section, we use the similar usecase from the previous chapter 4.7.1, which is the in-flight service provisioning in a realistic European-based Space-Air-Ground Integrated Network (SAGIN). Similarly, two access points are considered for each flight, namely DA2G and satellite. In below, we introduce the input parameters which are used explicitly in this chapter.

The set of considered flights has been exported from userRadar24 live air traffic for 24 hours on 9.11.2017. In our experiments, we select a set of flights $F$, such that $\mid F \mid = \{20, 50, 100, 300, 500\}$. For simplification, we assume that all the users are transmitting the same amount of traffic $b_{f,t}$ during all the timeslots. As the ILP model takes hours to solve a problem instance with 100 flights, the number of timeslots has been limited to eight so that we can have problem instances with a higher number of flights. Thus, we choose $F$ with the same duration, equal to seven timeslots (i.e., 3.5 hours of flight). Considering the speed of the flight (800-900 Kmph) and the DA2G node connection range, the duration of each timeslot is considered as 30 minutes. Note that this value can be changed based on different scenarios. Indeed, the number of timeslots depends on the user speed and the range of the DA2G node and satellite coverage, which can lead to an even different number of timeslots per flight. However, to simplify the evaluation scenario, we consider the same number of timeslots and duration for all the flights.

The simulation settings are based on a realistic European-based user space (see Fig. 4.5). We use the European Cost266 topology [78] for the ground core network. The delay of links in $\mathcal{L}_C$ is determined according to the length of the optical fiber transmissions. Regarding the DC nodes, we choose two sets of DC nodes, with 3 and 6 nodes, distributed over Europe: a small set $\mathcal{N}_{DC}$={Hamburg, Madrid, Budapest}, and a larger set $\mathcal{N}_{DC}$={Strasbourg, Stockholm, Madrid, Athens, Glasgow, Krakow}. We consider three cases for the DC capacity:

1. *Low*: $\forall j \in \mathcal{N}_{DC}, C_j^{DC} = \left\lceil \frac{|F|}{|\mathcal{N}_{DC}|} \right\rceil$ (note that lower DC capacity makes the problem infeasible, see Remark 1).
2. *High*: $\forall j \in \mathcal{N}_{DC}, C_j^{DC} = \mid F \mid$
3. *Medium*: the average of *low* and *high* values.

Since the focus of *Homa* is to provide (best-effort) low-delay in-flight services, we consider $\forall f \in F, t \in \mathcal{T}_f, b_t^f = 1$ with a relaxation on the bandwidth constraints for the access point node and ground network links. There are 295 DA2G node base stations (already deployed in Europe), which their location (in terms of latitude and longitude) is taken from [98]. The propagation delay from user to the DA2G node base station is set to 10 ms [82]. Further, we consider the DA2G node

connectivity range as 150 km [15], although in practice, it can vary based on, e.g., the antenna employed technology and weather conditions. Without loss of generality, we connect the user nodes $p_{f,t}$ to the closest DA2G node base station in range. Further, to make our scenario more realistic, we consider a DA2G node congestion model from the previous chapter 4.7.1.

Regarding the reconfiguration delay, we use the distance between source and destination DC nodes:

$$R_{i,j} = \begin{cases} M * dijkstra\_path\_length(i,j), i,j \in \mathcal{N}_{DC}\ (i \neq j), \\ 0, i,j \in \mathcal{N}_{DC}\ (i = j), \end{cases}$$

$M$ is a random number between 0 and 2. We note that the modeling of the service migration is out of the scope of this paper. However, different $R_{i,j}$ models can be plugged into the *Homa*, depending on the specific application.

To design different experiments, we use the input parameters and their values in Table 5.2 wherein in each experiment, three parameters are fixed, and the fourth one is varied. We implement and solve the ILP model using Gurobi [113] optimization solver. Also, *Homa*, *MC*, and *MD* approaches are implemented in Python. We perform the evaluations for 30 random sets of scenarios on a machine equipped with Intel Core i7-6700 CPU3.40 GHz, 16 GB RAM, and running Arch Linux with kernel 5.5.11-arch1-1.

**Table 5.2:** Input parameters and their values used for the performance evaluation. The default value is in bold text.

| Input parameter | Values |
|---|---|
| Number of flights ($\mid F \mid$) | 20, **50**, 100, 300, 500 |
| Flight duration ($\mathcal{T}_f, \forall f \in F$) | 4, **8** |
| Number of DCs ($\mid \mathcal{N}_{DC} \mid$) | 3, **6** |
| DCs capacity ($C_j^{DC}, \forall j \in \mathcal{N}_{DC}$) | *Low*, ***Medium***, *High* |

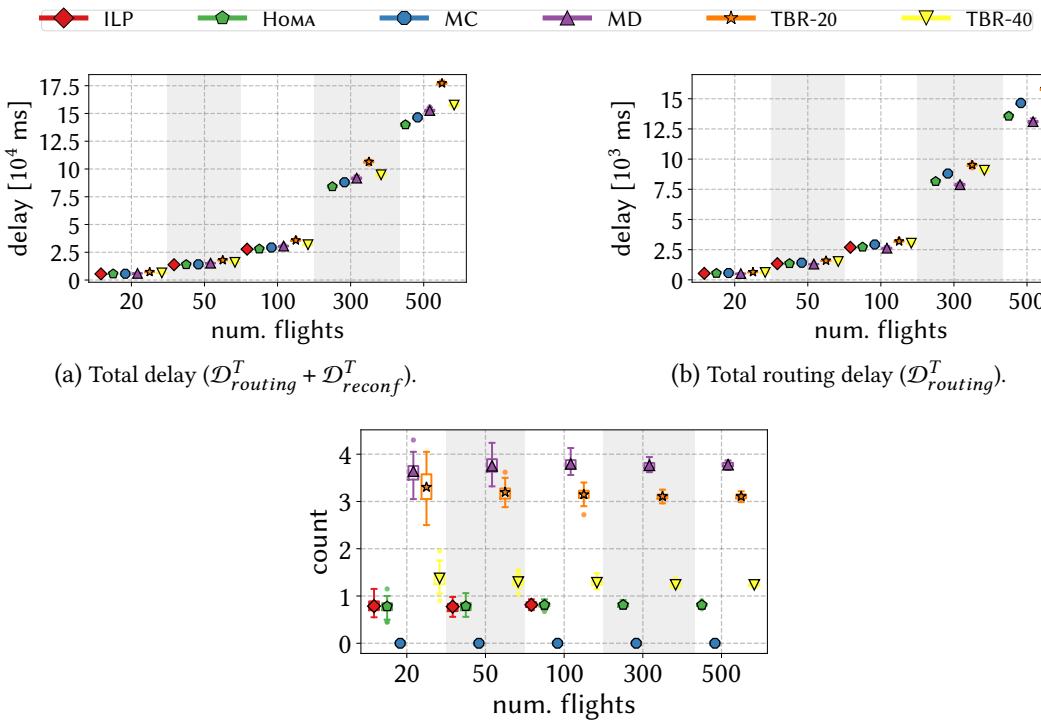
### 5.6.3 Simulation Results

In this part, we present the simulation results for different parameters and experiments.

#### 5.6.3.1 Impact of Number of Flights

In the first experiment, we first consider the default values for the flight duration $\mathcal{T}_f$, number of DCs $\mid \mathcal{N}_{DC} \mid$, and DC capacity $C_j^{DC}$ according to Table 5.2. The effect of number of flights on total delay (i.e., the objective function value), routing delay, and number of reconfigurations can be seen in Fig. 5.4a-5.4c. In Fig. 5.4a, it can be seen that *Homa* achieves a near-optimal objective function value with around 1% of distance from the optimal.

Also, *Homa* outperforms the baseline and state-of-the-art algorithms for up to 15% on average regarding the objective function value. The MD algorithm achieves a lower routing delay compared to *Homa* (see Fig. 5.4b) since it greedily reconfigures the airplane connection to the closest DC to




(a) Total delay ($\mathcal{D}^T_{routing} + \mathcal{D}^T_{reconf}$).

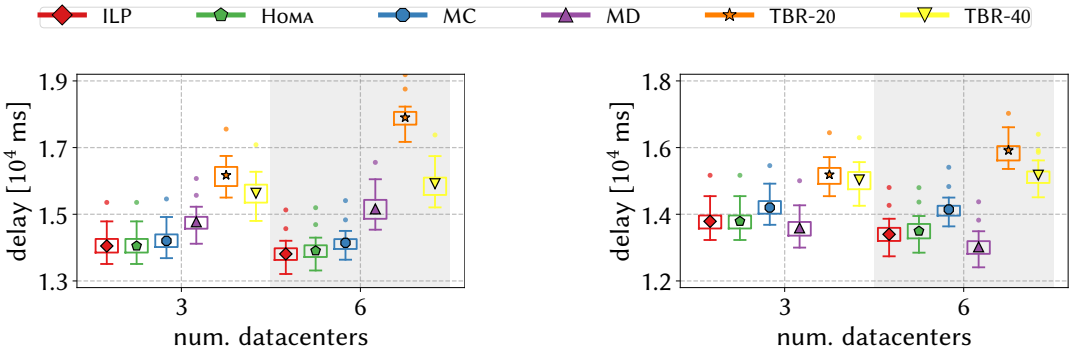
(b) Total routing delay ($\mathcal{D}^T_{routing}$).

(c) Number of reconfigurations.

**Figure 5.4:** Comparison of the simulation results for different number of flights.

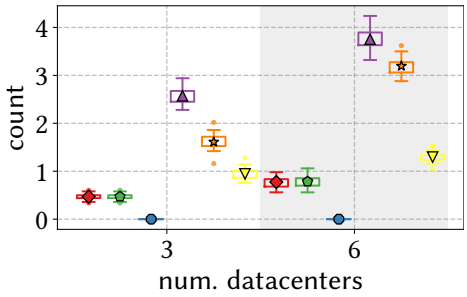
minimize the routing delay. However, it requires a significant number of reconfigurations to be able to minimize the airplane to DC delay (see Fig. 5.4c). Further, MC leads to a larger routing delay compared to *Homa*, since it keeps using the first (closest) selected DC for the whole flight period and does not reconfigure the connection. Therefore, during the flight, when the airplane gets far from the DC, the routing delay also increases. Also, it can be seen that the TBR-20 and TBR-40 lead to a higher objective value function. The reason is in TBR, determining the threshold value for triggering the reconfiguration is challenging, especially when the goal is to minimize the overall delay. Lower threshold values (i.e., TBR-20) can lead to higher total delays and more reconfigurations are triggered for TBR (see Fig. 5.4c). According to its design, TBR reconfigures the service to the farthest DC, which increases the routing delay. In addition, since TBR does not take the reconfiguration overhead into the account, the balance between the routing delay and reconfiguration delay is missed, hence, leading to sub-optimal decisions. Fig. 5.4c shows the number of per-flight reconfigurations for the flight duration, in which, the proposed *Homa* approach can make near-optimal reconfiguration decisions for the flights. Also, MC does not perform any reconfigurations.

Overall, it is evident that *Homa* can accurately balance the flight to DC routing delay and reconfigurations, demonstrating a near-optimal behavior while outperforming the baseline and state-of-the-art algorithms. We note that the average round-trip delay per flight per timeslot can be driven from Fig. 5.4a. This value is around 68 ms in the worst case for 500 flights using the proposed HOMA approach (requirements being around 100 ms [89]), which can support a wide range of interactive services such as air-to-ground voice calls, video conferencing, and gaming.




(a) Total delay ($\mathcal{D}^T_{routing} + \mathcal{D}^T_{reconf}$).

(b) Total routing delay ($\mathcal{D}^T_{routing}$).




(c) Number of reconfigurations.

**Figure 5.5:** Comparison of the simulation results for different number of DCs.

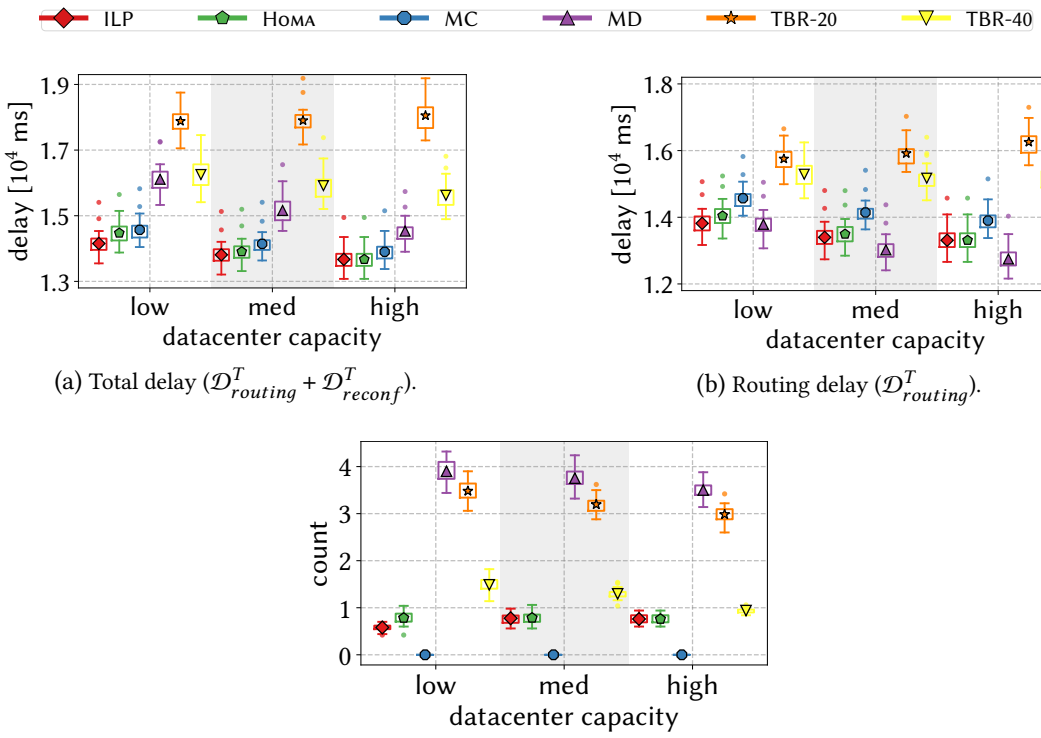
#### 5.6.3.2 Impact of Number of DC Nodes

Fig. 5.5a-5.5c show the evaluation results of the algorithms for varied number of DCs, while keeping the default values for other parameters. The goal here is to compare the behavior of different approaches when the number of available DCs changes. This can assist the operators in more accurate capacity planning.

Regarding the total delay, Fig. 5.5a indicates that a lower number of DCs leads to a higher delay value. That is because, with a higher number of DCs, flight nodes get in a closer distance to a DC node on average during the flight. It can also be seen in Fig. 5.5b, where the routing delay is generally lower in the case with six DC nodes. However, the TBR algorithm does not follow this trend, since by each reconfiguration, the routing delay increases (the reconfiguration is done towards the farthest DC).

Also, it can be seen that the difference in the difference of total delay achieved by the algorithms is generally lower for three DCs compared to six. It is because the role of DC capacity (which is the main cause of sub-optimality) is more critical with a higher number of DCs (i.e., there is more room to be sub-optimal).

However, regarding the number of reconfigurations, Fig. 5.5c shows that the case with six DCs has around 20% more reconfigurations compared to the case with three, since it brings more flexibility (and more opportunities for reconfiguration, especially for the TBR algorithm). Indeed, this value gets lower with services with higher reconfiguration costs. However, still, the case with three DC makes more sense since the total delay still leads to *reasonable* solutions, while the number of




(a) Total delay ($\mathcal{D}^T_{routing} + \mathcal{D}^T_{reconf}$).

(b) Routing delay ($\mathcal{D}^T_{routing}$).

(c) Number of reconfigurations.

**Figure 5.6:** Comparison of the simulation results for different DC capacities.

reconfigurations is kept low. The results show that using *Homa* can bring the operators substantial cost savings by renting/building 50% fewer DCs while sacrificing a low portion of solution quality.

#### 5.6.3.3 Impact of DC Capacity

In this analysis, we use the default values for the number of flight, flight duration, and the number of DC nodes and vary the DC capacity. The goal of this part is to find out what is the impact of different DC capacities on the performance of the algorithms. Fig. 5.6a shows the comparison of the achieved total delay by the algorithms for *low*, *medium*, and *high* DC capacities. Since our approach is online (user-by-user), the DC capacity can play a big role in having the available resources in a *good* DC location when needed. It can be seen that the total delay decreases with higher DC capacity.

It can be seen that generally, the behavior of the algorithms is similar for the cases with different DC capacities. However, the total delay decreases with higher DC capacity. Also, the same trend is there for the routing delay as in Fig. 5.6b. The reason behind this is there are more DCs available to serve the flights (i.e., average distance of airplanes and DCs is decreased). Moreover, as proved in Theorem 3, it is evident that *Homa* is able to take optimal decisions for DC selection and reconfiguration in case of *high* DC capacity. Finally, Fig. 5.6c shows that the DC capacity does not change the number of reconfigurations much for the optimal and *Homa*, i.e., they stay around one reconfiguration per flight for all the DC capacities. These evaluations show that it is not necessary to rent plenty of resources at each DC. In fact, *Medium* or even *low* amount of resources could provide a *good* solution. It indicates that in contrast to other algorithms, *Homa* can lead to cost savings for the operators.

#### 5.6.3.4 Runtime

The runtime comparison of the optimal offline solution (i.e., ILP), *Homa*, MD, MC, and TBR algorithms is presented in Table 5.3. Firstly, this table shows that the ILP formulation is significantly slower than the other heuristics. Also, it can be seen that *Homa* can solve the problem for 500 flights in around six seconds per flight. Also, MC has the lowest runtime since it does not consider the reconfiguration decisions. Thus the number of Dijkstra runs is considerably lower, compared to the *Homa*, and MD approaches. Finally, the TBR algorithms are the fastest ones since they are less sophisticated.

**Table 5.3:** The comparison of the mean runtime of the different approaches (in seconds) for the default input parameters.

| | $\|F\| = 20$ | $\|F\| = 50$ | $\|F\| = 100$ | $\|F\| = 300$ | $\|F\| = 500$ |
|---|---|---|---|---|---|
| ILP | 187.1 | 569.2 | 12412.4 | N/A | N/A |
| *Homa* | 15.3 | 52.2 | 166.8 | 1175.4 | 3216.5 |
| MD | 16.8 | 44.5 | 151.9 | 1014.7 | 3140.1 |
| MC | 4.4 | 11.2 | 57.5 | 123.5 | 163.6 |
| TBR-20 | 0.8 | 3.3 | 11.2 | 94.63 | 278.54 |
| TBR-40 | 0.5 | 2.4 | 8.7 | 80.8 | 228.1 |

#### 5.6.3.5 Discussion

Let us summarize some remarks related to *Homa*. Firstly, we choose to adopt the discrete-time model mainly due to its simplicity. Furthermore, this model can be arbitrarily close to the continuous-time model by making the timeslot duration very small. However, it is not the case for our scenario. Considering the flight speed and the access point node coverage range, the decision-making can be done in longer periods (on a scale of a few minutes).

Secondly, we consider the user paths a priori knowledge once planes are ready to take off. In practice, the paths can be changed in some emergencies (e.g., bad weather conditions). In this case, *Homa* can be rerun to adapt the previously-made decisions using the updated network status (e.g., network link availability/congestion status) according to the emergency.

Third, we model the Low-Earth Orbit (LEO) satellite constellation with a single representative node with the worst-case delay value. The integration of the dynamic satellite network considering the cost/delay of the satellite handover could be added to our approach as follows: the handover cost between two consecutive timeslots could be added to the cost of satellite connection links (i.e., dashed lines in Fig. 5.2).

Finally, we point out that *Homa* does not guarantee the service-specific delay requirements but follows a best-effort approach towards delivering low-delay services to the users. Nevertheless, the routing decision-making part of *Homa* can be enhanced with deterministic networking theories such as Network Calculus [208, 12], to provide a delay guarantee for the critical traffic flows (e.g., flight status).

## 5.7 *Figo* Online Algorithm

In this section, we present *Figo*, the second approach to solve the D-JSARR problem. *Figo* utilizes ML techniques and is deployed on a controller node which can learn the efficient policy to determine the assignment, routing, and reconfiguration decisions. ML is an application of Artificial Intelligence (AI) which can improve an algorithm automatically by experiencing and using the data. In other words, ML algorithms use sample data (i.e., training data) to model and make predictions and decisions on a specific problem, such as Email filtering, pattern recognition, and computer vision [31]. Reinforcement Learning (RL) is currently one of promising branches in ML research. It has shown extraordinary problem-solving abilities in different fields such as medicine, robotics, and even could tackle the previously unsolvable games such as Go [167]. RL is based on a trial and error method, in which an agent interacts with an environment and receives feedback for the performed action. The ultimate goal is to learn a good policy for sequential decision problems by maximizing a cumulative reward [167]. The RL agent collects the system state for each episode and calculate the expected reward at each timeslot. Then, it selects an action according to a predefined strategy, and the system transfers to a new state in the next timeslot. Similarly, the agent calculates the reward and chooses new action. We define the state space, action set, and reward function.

### 5.7.1 State-Space

We define state-space $\mathbb{S}$, containing states $S(t)$ for the user $f$ at timeslot $t$. Each state consists of three elements:

$$\mathbb{S} = \{S(t) = (p_{f,t}, p_{f,\mathcal{T}_f}, j_t) \mid \forall f \in F, \forall j \in \mathcal{N}_{DC}\}, \tag{5.18}$$

where $p_{f,t}$ and $p_{f,\mathcal{T}_f}$ correspond to the current and final position of the user $f$, respectively. It conveys both a sense of mobility and enables the agent to plan for a long-term solution. In order to diminish our state-space, we divide the network (graph $G$) into areas on a $100 \times 100$ grid, from which the positions $p_{f,t}$ are taken. This enables an improved exploration during the training phase and also an increase in performance for unknown users. The last component of the state is $j_t \in \mathcal{N}_{DC}$ which is the current DC that the user is assigned to.

### 5.7.2 Action Set

The actions are the set of user-to-DC mappings that can change at each timeslot (i.e., reconfiguration operation). The routing decisions are pre-calculated based on the shortest-path algorithm. Therefore, we define the action set $\mathbb{X}$ as follows:

$$\mathbb{X} = \{\chi_j \mid j \in \mathcal{N}_{DC}\}, \tag{5.19}$$

where $\chi_j$ indicates if a given user is assigned to DC $j$. Note that in *Figo*, for simplification reasons, the link capacity constraints (Eq. 5.8) are relaxed.

### 5.7.3 Reward Function

At each timeslot, the agent gets a reward $\mathbb{R}(S, \chi)$ in a specific state $S$ after executing action $\chi$ from $\mathbb{X}$. We define the reward function $\mathbb{R}$ as below:

$$\mathbb{R}(S, \chi) = \mathcal{D}(S, j) - \mathcal{D}^t_{reconf} - \mathcal{D}^t_{routing}, \forall j \in J, \forall t \in T, \tag{5.20}$$

where $\mathcal{D}(S, j)$ is the delay of Dijkstra's shortest-path from the current user position to the closest DC $j$. Also, $\mathcal{D}^t_{reconf}$ and $\mathcal{D}^t_{routing}$ are the reconfiguration and routing delays at the timeslot $t$, respectively. The comparison with the minimum routing delay for each DC allows all timeslots to have a common baseline. Also, it prevents large reward variance at each timeslot, thus avoiding an inconsistent training, which would lead to slow convergence.

### 5.7.4 The Machine Learning (ML) Framework

Figo uses Q-learning to solve the D-JSARR problem. Q-learning belongs to RL algorithms, in particular, to off-policy and model-free category. Off-policy RL methods do not actively follow the policies that they learn. In this case, the agent tries to learn from its actions. Even sometimes, it takes random actions (greedy policy) to explore the scenario fully. Model-free algorithms learn the optimal policy by estimating the optimal Q-values for each state without previous knowledge of the environment. Q-value $Q(S, \chi)$ is a metric to evaluate how *good* action $\chi$ taken at a particular state $S$ is. These Q-values are stored in the Q-table. It is in contrast to model-based algorithms that learn the optimal policy by searching the so-called policy space. Further, in contrast to Model-based approaches, the model-free Deep Q-Learning (DQN) cannot make predictions for the next state and will only choose the optimal action for every single state. Notably, a model-based approach would allow an agent to look and choose an action considering future actions. The downside is that it learns a model purely from experience, making the model-based approaches very time-consuming, while model-free methods are more efficient, easier to implement, and tune.

In Q-learning, values are updated according to the greedy policy, where the actions with the highest Q-value are chosen. These values are updated according to the Bellman Equation. Then, at a given state, Q-learning chooses an action by looking up the highest value in the table or random (epsilon-greedy). After each interaction, the reward is measured, and the Q-table is updated. Thus, the Q-table can be defined and $M \times N$ matrix, with $M$ actions and $N$ states:

$$Q_{t+1}(S(t), \chi(t)) = (1 - \alpha)Q_t(S(t), \chi(t)) + \alpha[\mathbb{R}(t) + \gamma \cdot \max_{\mathbb{X}} Q_{t+1}(S(t+1), \chi(t+1))], \tag{5.21}$$

where $\alpha$ is the learning rate, and $\gamma$ is the discount factor, which allows us to balance the concentration on short or long-term rewards. This procedure is not practical to be used in the case of a large state-space[106]. In particular, it is much more difficult for an agent to visit each state and store all these Q-values in a large state-space. To tackle this issue, an alternative is DQN, in which a Deep Neural Network (DNN) is deployed. In particular, it is used to approximate the optimal action-value function, enabling it to predict the Q-values for each state accurately. As shown in [163], using the DQN allows learning of *good* policies within a large and complex state-space usecase. However, a problem with DQN is that the same value is used to both evaluate and choose the actions performed

by the RL agent, which may result in over-optimistic results. By separating these values, overestimation can be prevented. Thus, we propose to use Double Deep Q-Learning (DDQN)[209] which tackles this problem by using two Neural Networks (NNs), one to evaluate the Q-values, while the other one chooses the action to take. Thus, DDQN would help *Figo* to produce more robust and accurate policy learning.

### 5.7.5 Training

*Figo* learns by using the continuously received user data in a simulated environment. Long-term and continuous training enables *Figo* to improve the returned solutions for unknown scenarios. Further, we use Prioritized Experience Replay (PER) [189] which stores the previous experiences in replay buffers, instead of discarding them. Later, we sample from these experiences to train on data that is not as correlated and correct, therefore avoiding depreciating policies. For efficient training, two parameters need to be contemplated:

1. Environment: The environment is what our RL agent interacts with during the training. It provides our agent with the observation and reward of the performed action. The environment is deterministic as well as fully observable, in which the data for the user location and the DC assignment are available for the whole time horizon $T$ (discrete). We base our environment structure on the openAI Gym [55]. To further improve performance, in addition to the current agent state, the routing delay to each DC is added as an input to the RL agent observation. It helps the agent to be able to distinguish possible *good* actions.

2. Hyperparameters: The DNN is updated during training by using a batch of randomly picked data from past observations [189]. We define a decreasing $\epsilon$ value for the training algorithm to trade-off the exploration against the exploitation [33]. The last parameter to be considered is the discount factor $\gamma$, which balances the weight of long and short-term rewards. When the value of $\gamma$ is close to one, it indicates higher importance for long-term rewards. After performing a grid-search, we conclude to use the training hyperparameters listed in Table 5.4.

**Table 5.4:** Training hyperparameters.

| Parameter | Value |
|---|---|
| Learning rate $\alpha$ | $5 \times 10^{-4}$ |
| Discount factor $\gamma$ | 0.9 |
| Final exploration factor $\epsilon$ | 0.02 |
| Exploration fraction | 30% |
| Batch size $n$ | 32 |
| NN layers (including input and output) | 4 |
| Training length | $10^5$ timeslots |

#### 5.7.5.1 Neural Network (NN) Structure

We use a NN with two fully-connected hidden layers, using a rectifier linearity activation function. The input layer is composed of 18 input neurons, the size of our observation. The output layer is fully-connected and consists of one output neuron for each possible action. Each output neuron displays a probability value between 0 and 1. The action neuron with the highest value is then chosen for each timeslot. To optimize the agents NN performance, all the data features are normalized, preventing the input features from influencing the DNN weights differently.

## 5.8 Performance Evaluation

In this section, we introduce our simulation scenario and parameters for evaluation. Thereafter, we compare the proposed *Figo* framework against the optimal ILP solution.

### 5.8.1 Simulation Setup

The simulation scenario and input parameters are the same European-based SAGIN which was used for the previous Chapter 4.7.1. However, we only mention the additional settings that we consider for the current evaluation. For a detailed evaluation of *Figo*, we consider two different DA2G node connectivity ranges $\Psi_{DA2G}$ as 70 and 150 km. The set of flights has been exported from userRadar24 live air traffic for 24 hours on 9.11.2017. Due to the high computation complexity of the ILP, we choose a set of only 50 random flights from our flight database. However, *Figo* can solve the problem for a higher number of flights. Let us consider the duration of each timeslot as 30 minutes and flights with two different durations: *i) short* flights with the duration of 2 hours, i.e., $\mathcal{T}_f = 4$, and *ii) long* 4-hour flights i.e., $\mathcal{T}_f = 8$. In our scenario, based on the reconfiguration cost model presented in Section 5.6.2 we consider two services with low and high reconfiguration delays, $\mathfrak{R} = 5$, and $\mathfrak{R} = 25$ ms, respectively.

The ILP model is implemented and solved with Gurobi [113] in Python. Moreover, *Figo* is implemented with TensorFlow [20] in Python. The training and simulations are performed on a machine equipped with Intel Core i7-6560U CPU and 8 GB of RAM.

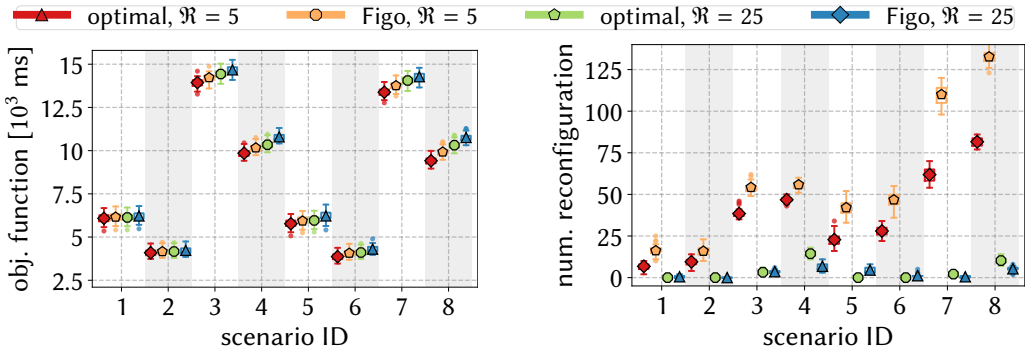
**Table 5.5:** The considered scenarios with $|F| = 50$ with different number of DC nodes $|\mathcal{N}_{DC}|$, flight duration $\mathcal{T}_f$, and DA2G node coverage $\Psi_{DA2G}$.

| Scenario ID | Settings $(\lvert\mathcal{N}_{DC}\rvert, \mathcal{T}_f, \Psi_{DA2G})$ | Scenario ID | Settings $(\lvert\mathcal{N}_{DC}\rvert, \mathcal{T}_f, \Psi_{DA2G})$ |
|---|---|---|---|
| 1 | (3, 4, 70 km) | 2 | (3, 4, 150 km) |
| 3 | (3, 8, 70 km) | 4 | (3, 8, 150 km) |
| 5 | (6, 4, 70 km) | 6 | (6, 4, 150 km) |
| 7 | (6, 8, 70 km) | 8 | (6, 8, 150 km) |

### 5.8.2 Simulation Results

#### 5.8.2.1 Total Delay and Number of Reconfigurations

We start by comparing the performance of *Figo* with the optimal solution in eight different scenarios summarized in Table 5.5. We note that each scenario is run 30 times randomly. As depicted in Fig. 5.7a a higher total delay can be observed for larger reconfiguration delays and an increase of reconfigurations for the smaller $\mathfrak{R}$ value. It can be seen that *Figo* can produce near-optimal results in all the scenarios. Also, the instances with higher $\mathfrak{R}$ have a higher objective function on average. The reason is that for small $\mathfrak{R}$, the algorithm has more flexibility in improving the objective function. This fact can be observed in Fig. 5.7b wherein all scenarios, the number of reconfigurations is lower in the case of $\mathfrak{R} = 25$.




(a) Objective function value

(b) Number of reconfigurations

**Figure 5.7:** Comparison of the objective function value (sum of routing and reconfiguration delays) and the number of reconfigurations for *Figo* and the optimal solution in the eight different scenarios of Table 5.5.

An interesting observation in Fig. 5.7a is when we compare scenarios $(i, j)$ with different number of DC nodes (i.e., scenarios (1,5), (2,6), (3,7), and (4,8)). In these scenarios, the total delay value is generally lower on average for six DCs compared to three. This can give the algorithm more reconfiguration possibility, as Fig. 5.7b indicates. In particular, in scenarios with $|\mathcal{N}_{DC}| = 6$, more reconfigurations are performed, especially when $\mathfrak{R} = 5$ due to even higher flexibility. That is why the improvement is generally lower for $\mathfrak{R} = 25$ compared to $\mathfrak{R} = 5$. Hence, it can be concluded that if the reconfiguration delay of the services is low/high, airline companies should go for a higher/lower number of DCs to achieve the same delay level for their in-flight services.

In addition, Fig. 5.7a shows that the scenarios with higher flight duration (e.g., Scenarios (3, 7) compared to (1, 5)) lead to obviously higher total delay. Also, higher values of $\Psi_{DA2G}$ (e.g., Scenarios (6, 8) compared to (5, 7)) can lead to a lower objective function value, since higher $\Psi_{DA2G}$ makes the flights more reachable to DC locations.

In the following, we investigate the impact of changes in the flight duration, the number of DCs, DA2G coverage, and the number of flights on the optimality gap of *Figo* with different reconfiguration $\mathfrak{R}$ delays.

#### 5.8.2.2 Impact of Flight Duration

Let us choose some of the challenging scenarios for closer analysis. Fig. 5.8 compares the effect of flight duration on the optimality gap by considering scenarios 6 and 8 where $|\mathcal{N}_{DC}|$ and $\Psi_{DA2G}$ are fixed. It shows that the optimality gap is the lowest for Scenario 8 ($\mathcal{T}_f = 8$) and $\mathfrak{R} = 5$, and the same scenario with $\mathfrak{R} = 25$ has the highest gap.

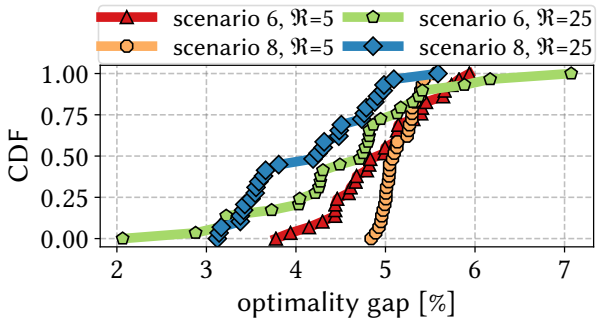



**Figure 5.8:** Impact of flight duration on optimality gap of *Figo*.

The reason is $\mathfrak{R} = 5$ brings more reconfiguration options than the $\mathfrak{R} = 25$ case. In this case, the reconfiguration delay is the deal-breaker in the optimality gap instead of the flight duration. This is an advantage of *Figo*, which performs with a close distance to optimal, even in more realistic scenarios where flights have a longer duration.

#### 5.8.2.3 Impact of Number of DC Nodes

Fig. 5.9 compares the optimality gap for scenarios with fixed $\mathcal{T}_f$ and $\Psi_{DA2Gnode}$ and varying $|\mathcal{N}_{DC}|$ (i.e., Scenarios 4 and 8 with $|\mathcal{N}_{DC}| = 3$ and $|\mathcal{N}_{DC}| = 6$, respectively). We observe that Scenario 4 performs better than 8 when $\mathfrak{R} = 5$; however, for $\mathfrak{R} = 25$, they almost perform similarly. We know that Scenario 8 has a larger state-space since it scales exponentially with the number of DCs, making it harder to achieve a close optimality gap. However, it can achieve a better optimality gap in both

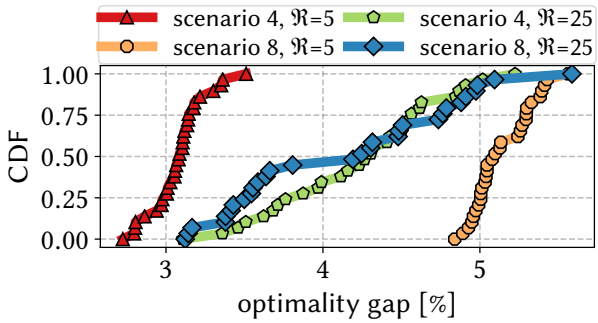



**Figure 5.9:** Impact of number of DC nodes on optimality gap of *Figo*.

$\mathfrak{R} = 5$ and $\mathfrak{R} = 25$. We can conclude that the number of DCs has more impact on the performance than the reconfiguration delay.

#### 5.8.2.4 Impact of DA2G Coverage Range

In this part, we compare the performance of *Figo* against the optimal solution for different $\Psi_{node}$ in Fig. 5.10. The results indicate that the smaller DA2G node range (i.e., Scenario 7) diminishes the number of DA2G node connections. Considering $\mathfrak{R} = 25$, this fact leads to even fewer reconfiguration potentials since the ideal DC assignment would be located near the satellite gateway node.

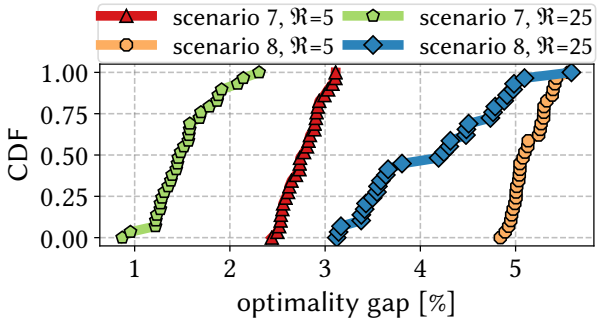



**Figure 5.10:** Impact of DA2G coverage range on optimality gap of *Figo*.

#### 5.8.2.5 Scalability of *Figo*

An essential characteristic of *Figo* is its ability to solve the problem for a large number of users with an acceptable optimality gap. To show this, we ran scenario 8 for an additional 30 runs, taking 20, 50, and 100 flights into account. According to the results, it is evident that the number of users does not influence the performance of *Figo*. In particular, Fig. 5.11 indicates that the optimality gap lies between 3% and 6% on average for different numbers of users.

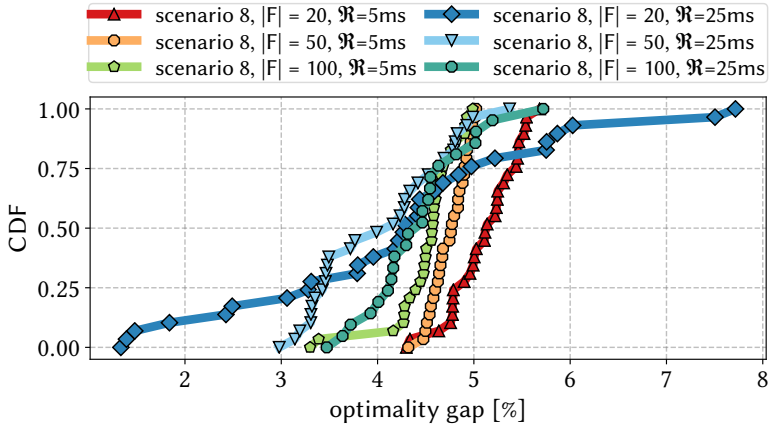



**Figure 5.11:** Optimality gap of *Figo* for different number of flights.

#### 5.8.2.6 Training and Runtime

Training duration is mainly influenced by two parameters, the sample size and convergence time. To reach an acceptable level of performance, the agent needs to go through a huge number of simulation runs. The sample size affects our algorithm in two ways. Firstly, a larger sample size means a larger graph $G$ to analyze. Since we use Dijkstra to calculate the shortest path, a higher number of nodes

leads to a longer calculation time for each Dijkstra run. Secondly, RL needs a lot of different samples to learn a generalized behavior. This means that while we would be able to achieve convergence within seconds and a few hundred iterations with a single flight, we need to perform thousands of iterations to teach the agent something meaningful. In our case, training phase takes around 90 minutes for $|F| = 100$, $\mathcal{T}_f = 8$, and $|\mathcal{N}_{DC}| = 6$.

After training, *Figo* is able to solve the problem for scenarios 7 and 8 with $|F| = 100$ in 26 seconds on average. Also, it takes around 18 seconds to solve scenarios 3 and 4 for the same number of flights (fewer DCs leads to having lower runtime). Finally, we report that in an online manner, given a single flight, *Figo* takes around 500 ms to find a solution on average for all the scenarios.

## 5.9 Summary

This chapter presented a study on best-effort delay minimization in networks with mobile users. We initiated the study of an online service provisioning problem in the context of emerging dynamic networks, such as 5G/6G. We presented a formal model together with two efficient online algorithms to solve it heuristically. The D-JSARR problem is to decide about user-to-DC assignments, routing, and possible reconfiguration decisions over the activity period, such that the sum of routing and reconfiguration delays are minimized over the whole time horizon.

We first presented an offline optimization formulation that solves the D-JSARR optimally. After that, we proved its NP-hardness and the need for fast algorithms for realistic and large problem instances. As the first solution, we presented *Homa* heuristic framework. It first converts the D-JSARR problem, into a special matching problem in context of graph theory, DC-MWBM. Then, using a transformation to the shortest-path problem, it solves the D-JSARR in a user-by-user approach. *Homa* showed a good performance in realistic scenarios, achieving a near-optimal objective function value while significantly reducing the runtime. Moreover, even in some scenarios, we showed that *Homa* could even achieve an optimal solution.

As the second solution, we presented *Figo*, which is a framework based on Deep Reinforcement Learning (DRL). We discussed the concept of DRL and its applicability to dynamic problems such as D-JSARR. *Figo* takes the user mobility into account, overlooking short-term rewards for the long-term ones. Further, the evaluations show the applicability of Figo in real-world scenarios while performing the assignment, routing, and reconfiguration decisions in a cost- and time-effective manner while still keeping a near-optimal solution.

# Chapter 6

# Conclusion and Future Work

This chapter concludes the dissertation with a summary and discussion of the key findings and the outlook for future work and open research questions. The main goal of this thesis was to advance the state-of-the-art approaches for network resource allocation and reconfiguration algorithms. Especially, the targeted networks were the *programmable networks* which employs Software-Defined Networking (SDN) and Network Function Virtualization (NFV) technologies.

## 6.1 Power-Aware and Delay-Guaranteed Service Function Chaining

### 6.1.1 Summary

We started by focusing on online resource allocation of Service Function Chaining (SFC) requests. The challenges were to determine, *i)* where to place the Virtual Network Functions (VNFs) of the SFC, *ii)* which routing path to use, while guaranteeing the bandwidth and delay requirements, and different resource capacity constraints. Motivated by the low power-proportionality of both Physical Machines (PMs) and network switches, we aimed at minimizing the number of online devices that can lead to improved resource utilization and more power-proportionality. To achieve this goal, we first propose an Integer Linear Program (ILP) model to be able to determine the optimal solution. Due to its scalability issues, we presented an online heuristic framework named *Holu*. *Holu* tackles the problem by breaking it into two sub-problems which are solved sequentially: *i) VNF placement*, and *ii)* routing. The VNF placement suggests a set of candidate PMs to host the requested SFC. It uses a PM ranking mechanism considering the centrality of the PM and the power consumption impact by hosting a VNF. We show that centrality is an important metric to consider for PM ranking since it can improve resource utilization and power efficiency in the long term. Also, to find a path traversing through the set of suggested candidate PMs (returned by the VNF placement sub-problem), we employed a complete algorithm based on Lagrange Relaxation based Aggregated Cost (LARAC) heuristic to solve a Delay-Constrained Least-Cost (DCLC) shortest-path routing problem. We showed that our algorithm could efficiently split the delay budget between two consecutive VNFs in an SFC, leading to a high acceptance ratio compared to state-of-the-art algorithms.

### 6.1.2 Future Work

We believe there are many interesting open challenges to be tackled, especially in improving the coordination of the VNF placement and routing sub-problems. For example, one can introduce a distance metric from the source and/or destination to the PM ranking calculation (See Eq. 2.18). Using that, similar to the A* algorithm [117], this information can be used to prune the candidate PMs that are far from source and/or destinations. It can improve the completeness of *Holu*, hence, its performance, especially in terms of acceptance rate. In addition, PM selection methods can be improved to generate *better* candidate PM sets. For example, one can replace multiple PMs based on more complex PM selection methods at each iteration to improve the search space exploration. These improvements can also lead to reducing the average number of VNF placement iterations, thus, reducing the runtime. Finally, a resource (PM and network) consolidation module can be added to *Holu*, which can be triggered periodically or in case of resource over/under-load. For example, in case of resource overload, it can use live Virtual Machine (VM) migration to migrate the workload away from the *hot* spots, thus, potentially improving the acceptance rate. Also, in the case of under-load, it can move the load away from an under-utilized region to further improve the total power efficiency.

The above problem focused on allocating the resources to the incoming SFC request; however, it does not consider the dynamicity of the networks. To shed light on this gap, we focused on proposing resource allocation and reconfiguration algorithms by considering two dynamicity sources: *i)* the unexpected traffic changes, *ii)* the mobility of users.

## 6.2 Reconfigurations Due to the Traffic Changes

### 6.2.1 Summary

In this chapter, we studied a scenario where the network reconfiguration is triggered due to traffic changes in the network. In more detail, we focused on the dynamic network reconfiguration problem in a multi-period resource planning scenario with an unexpected traffic increase. We introduced an Routing, Configuration, and Spectrum Assignment (RCSA) algorithm to answer these three main questions: *when*, *what* to reconfigure, and *how* to reconfigure the network to efficiently cope with the increased traffic? To achieve it, we have considered three types of reconfiguration: upgrading, adding, and/or rerouting the demands. Our proposed RCSA algorithm benefits from all the three reconfiguration methods, leading to a performant solution. In this work, we formulated and compared four different variations of our proposed RCSA algorithm, mainly in terms of network throughput, deployed Lightpaths (LPs), and power-efficiency. It has been shown that the multi-objective approach, *Min LP, Max DR*, performs the best by achieving the highest power efficiency while reducing the number of required LPs and lower over-provisioned throughput.

Further, the simulation results indicated that combining the three reconfiguration methods can significantly increase the network throughput, using almost the same network resources (e.g., number of deployed LPs). As a result, huge cost savings and spectrum efficiencies for network providers can be achieved. Additionally, our results showed that the rerouting method performs better on LPs with a low data rate. Therefore, it is evident that greedily upgrading LPs to high data rates can reduce

the benefits of this method. Hence, opposed to intuition, the *LP Upgrade* method is not as efficient since it can lead to a lower number of rerouting possibilities and lower overall throughput.

### 6.2.2 Future Work

Several interesting future works can be foreseen in this research area. We know that the introduced LP reconfiguration methods are not for free; indeed, they come with a cost. However, the exact overhead of these methods is not still clearly known. Therefore, using a lab setup with some commercial hardware equipment, the reconfiguration overhead of these methods can be measured. Especially, the metrics such as reconfiguration time, performance overheads, and service downtime can be exciting. These measurement results can be used for forming and studying a trade-off between reconfiguration cost and the achieved throughput in flexible optical networks.

The currently proposed approach works reactively. As a future research direction, Machine Learning (ML) techniques can be used to predict the network traffic, optical transmission status, and based on that, proactively reconfigure the network.

Another possible future direction is the improvement of the noise estimation calculations. There is a trade-off between the complexity (and hence runtime) and the accuracy of the noise calculation model, which can be applied to specific problems. Further, the development of automated network management frameworks with ML approaches can be used for both noise estimation and reconfiguration tasks.

## 6.3 Reconfigurations Due to the User Mobility: Cost Optimization

### 6.3.1 Summary

In this chapter, we studied the minimization of operational costs for the use cases with mobile users. More specifically, considering different access point mediums, we have focused on minimizing the cost of resources such as VMs, network links, and also the cost of reconfigurations such as VM migrations. We have presented a generic system model and formulated two ILPs models for the Joint Service Placement, Assignment, Routing, and Migration (JSPARM) problem: *i)* The static case, Static Joint Service Placement, Assignment, Routing, and Migration (S-JSPARM), and *ii)* the mobility-aware/dynamic one, Mobility-Aware Joint Service Placement, Assignment, Routing, and Migration (MA-JSPARM). The former considers one single timeslot, while the latter one considers a set of future user positions for solving the JSPARM problem and minimizing the total operational costs.

Further, we have used a realistic case study for the flying airplanes for a European-based Space-Air-Ground Integrated Network (SAGIN), which includes the satellite network, the Direct-Air-to-Ground (DA2G) network, the ground network, as well as the flights and the distributed Data Center (DC) locations. We demonstrated MA-JSPARM is able to utilize DA2G and satellite connections to satisfy the passenger service requests in a cost-effective manner, compared to S-JSPARM. In addition, we showed a trade-off between routing and migration costs. Also, we showed that a VM migration model could avoid unnecessary migrations and improve the long-term cost-efficiency of MA-JSPARM. Additionally, our evaluation results showed that S-JSPARM achieves higher delay levels and lower runtime compared to MA-JSPARM.

The main issue with the presented optimization models is their scalability. Due to the NP-hardness of the problem, S-JSPARM and MA-JSPARM cannot solve the large problem instances in a tractable time. Therefore, we came up with two heuristic algorithms, which break the problem into smaller sub-problems. On the one hand, we introduced the User-by-User (UbU) algorithm to solve the JSPARM problem for each user sequentially over the whole time horizon. On the other hand, we presented the Timeslot-by-Timeslot (TbT) algorithm, which solves the problem at each timeslot (similar to S-JSPARM). The results indicate that while reducing the runtime from hours to seconds, the heuristics can achieve near-optimal solutions.

### 6.3.2 Future Work

The limitation of TbT approach is the consideration of the deployed VM and routing from only the previous timeslot. On the other hand, UbU is limited to one user at each round of the algorithm. Therefore, the solution depends on the location of the previous users. Also, a *Rolling Horizon* method might be helpful to solve the problem over time horizon $T$. Thus, an improvement to TbT (UbU) can be the consideration of two or even more previous timeslots (users) to reduce the *blindness* of the algorithm (using time window to control the size of the sub-problem).

Another improvement could be to combine TbT and UbU. For example, a problem can be solved sequentially firstly by TbT and then UbU considering the solution of the TbT. In this way, we can consider both timeslots and the user positions at the same time. Additionally, clustering algorithms (based on the path and geographical location of users), local search, and randomization algorithms can help to improve the solution quality.

## 6.4 Reconfigurations Due to the User Mobility: Delay Optimization

### 6.4.1 Summary

This chapter presented a study on best-effort delay minimization in networks with mobile users. We initiated the study of an online service provisioning problem in the context of emerging dynamic networks, such as 5G/6G. We presented a formal model together with two efficient online algorithms to solve it heuristically. The Dynamic Joint Service Assignment, Routing, and Reconfiguration (D-JSARR) problem is to decide about user-to-DC assignments, routing, and possible reconfiguration decisions over the activity period, such that the sum of routing and reconfiguration delays are minimized over the whole time horizon.

We first presented an offline optimization formulation that solves the D-JSARR optimally. After that, we proved its NP-hardness and the need for fast algorithms for realistic and large problem instances. As the first solution, we presented *Homa* heuristic framework. It first converts the D-JSARR problem, into a special matching problem in context of graph theory, Dynamic Constrained Minimum-Weight Bipartite Matching (DC-MWBM). Then, using a transformation to the shortest-path problem, it solves the D-JSARR in a user-by-user approach. *Homa* showed a good performance in realistic scenarios, achieving a near-optimal objective function value while significantly reducing the runtime. Moreover, even in some scenarios, we showed that *Homa* could even achieve an optimal solution.

As the second solution, we presented *Figo*, which is a framework based on Deep Reinforcement Learning (DRL). We discussed the concept of DRL and its applicability to dynamic problems such as D-JSARR. *Figo* takes the user mobility into account, overlooking short-term rewards for the long-term ones. Further, the evaluations show the applicability of Figo in real-world scenarios while performing the assignment, routing, and reconfiguration decisions in a cost- and time-effective manner while still keeping a near-optimal solution.

### 6.4.2 Future Work

We see our work as a first step and believe it opens several interesting avenues for future research. In particular, it would be interesting to explore the achievable competitive ratio in deterministic and randomized online settings and explore how to optimize and re-optimize schedules in scenarios that feature some uncertainty and in which a bounded number of unexpected events may occur.

Another important, yet challenging research direction could be in direction of implementation of the dynamic resource management techniques. In particular, developing technologies to support zero-touch reconfigurations for container/VM migrations, traffic re-routing, and user handover protocols and mechansism seems a challenging task. Further, the resource management and reconfiguration for use-cases with non-deterministic mobility patterns is another future research direction.

# Bibliography

## Publications by the Author

### Journal Publications


[1] N. Deric, A. Varasteh, A. Van Bemten, A. Blenk, and W. Kellerer. "Enabling SDN hypervisor provisioning through accurate CPU utilization prediction." In: *IEEE Transactions on Network and Service Management* 18.2 (2021), pp. 1360–1374.

[2] M. He, A. Varasteh, and W. Kellerer. "Toward a flexible design of SDN dynamic control plane: An online optimization approach." In: *IEEE Transactions on Network and Service Management* 16.4 (2019), pp. 1694–1708.

[3] A. Varasteh, S. Akhoondian Amiri, and C. Mas-Machuca. "HOMA: Online In-Flight Service Provisioning with Dynamic Bipartite Matching." In: *IEEE Transactions on Network and Service Managament (TNSM)*. 2022.

[4] A. Varasteh, B. Madiwalar, A. Van Bemten, W. Kellerer, and C. Mas-Machuca. "Holu: Power-Aware and Delay-Constrained VNF Placement and Chaining." In: *IEEE Transactions on Network and Service Management (TNSM)* (2021). DOI: 10.1109/TNSM.2021.3055693.


### Conference Publications


[5] S. Amjad, A. Varasteh, N. Deric, and C. Mas-Machuca. "Delay-aware dynamic hypervisor placement and reconfiguration in virtualized SDN." In: *2021 12th International Conference on Network of the Future (NoF)*. IEEE. 2021, pp. 1–9.

[6] N. Deric, A. Varasteh, A. Basta, A. Blenk, and W. Kellerer. "SDN hypervisors: How much does topology abstraction matter?" In: *2018 14th International Conference on Network and Service Management (CNSM)*. IEEE. 2018, pp. 328–332.

[7] N. Deric, A. Varasteh, A. Basta, A. Blenk, R. Pries, M. Jarschel, and W. Kellerer. "Coupling VNF orchestration and SDN virtual network reconfiguration." In: *2019 International Conference on Networked Systems (NetSys)*. IEEE. 2019, pp. 1–3.

[8] N. Deric, A. Varasteh, A. Van Bemten, C. Mas-Machuca, and W. Kellerer. "Towards Understanding the Performance of Traffic Policing in Programmable Hardware Switches." In: *2021 IEEE 7th International Conference on Network Softwarization (NetSoft)*. IEEE. 2021, pp. 70–78.

[9] R. Durner, A. Varasteh, M. Stephan, C. M. Machuca, and W. Kellerer. "Hnlb: Utilizing hardware matching capabilities of nics for offloading stateful load balancers." In: *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE. 2019, pp. 1–7.

[10] F. Tashtarian, A. Varasteh, A. Montazerolghaem, and W. Kellerer. "Distributed VNF scaling in large-scale datacenters: An ADMM-based approach." In: *2017 IEEE 17th international conference on communication technology (ICCT)*. IEEE. 2017, pp. 471–480.

[11] A. Van Bemten, N. Deric, A. Varasteh, A. Blenk, S. Schmid, and W. Kellerer. "Empirical Predictability Study of SDN Switches." In: *Proceedings of the ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*. ACM/IEEE. 2019, pp. 1–13.

[12] A. Van Bemten, N. Deric, A. Varasteh, S. Schmid, C. Mas-Machuca, A. Blenk, and W. Kellerer. "Chameleon: Predictable Latency and High Utilization with Queue-Aware and Adaptive Source Routing." In: *Proceedings of the ACM International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*. 2020, pp. 451–465.

[13] A. Varasteh, M. De Andrade, C. Mas-Machuca, L. Wosinska, and W. Kellerer. "Power-aware virtual network function placement and routing using an abstraction technique." In: *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE. 2018, pp. 1–7.

[14] A. Varasteh, S. Hofmann, N. Deric, M. He, D. Schupke, W. Kellerer, and C. Mas-Machuca. "Mobility-aware joint service placement and routing in space-air-ground integrated networks." In: *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE. 2019, pp. 1–7.

[15] A. Varasteh, W. Kellerer, and C. Mas-Machuca. "Network in the Air." In: *Proceedings of the 15th International Conference on emerging Networking EXperiments and Technologies*. 2019, pp. 20–22.

[16] A. Varasteh, S. K. Patri, A. Autenrieth, and C. Mas-Machuca. "Evaluation of Lightpath Deployment Strategies in Flexible-Grid Optical Networks." In: *Optical Fiber Communication Conference (OFC)*. 2021.

[17] A. Varasteh, S. K. Patri, A. Autenrieth, and C. Mas-Machuca. "Towards Dynamic Network Reconfigurations for Flexible Optical Network Planning." In: *25th International Conference on Optical Network Design and Modelling (ONDM)*. 2021.

[18] A. Varasteh, H. Soares Frutuoso, M. He, W. Kellerer, and C. Mas-Machuca. "Figo: Mobility-Aware In-Flight Service Assignment and Reconfiguration with Deep Q-Learning." In: *IEEE Global Communications Conference (Globecom)*. 2020.

## Demos

[19] A. Varasteh, S. Hofmann, N. Deric, A. Blenk, D. Schupke, W. Kellerer, and C. Mas-Machuca. "Toward optimal mobility-aware VM placement and routing in space-air-ground integrated networks." In: *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE. 2019, pp. 1–2.

## General Publications


[20] M. Abadi. "TensorFlow: learning functions at scale." In: *Proceedings of the 21st ACM SIGPLAN International Conference on Functional Programming*. 2016, pp. 1–1.

[21] Z. Abbasi, G. Varsamopoulos, and S. K. Gupta. "Tacoma: Server and workload management in internet data centers considering cooling-computing power trade-off and energy proportionality." In: *ACM Transactions on Architecture and Code Optimization (TACO)* 9.2 (2012), pp. 1–37.

[22] A. Abdulkadiroglu and T. Sonmez. "Matching markets: Theory and practice." In: *Advances in Economics and Econometrics* 1 (2013), pp. 3–47.

[23] D. Abts, M. R. Marty, P. M. Wells, P. Klausler, and H. Liu. "Energy proportional datacenter networks." In: *Proceedings of the 37th annual international symposium on Computer architecture*. 2010, pp. 338–347.

[24] T. Ahmed, S. Rahman, S. Ferdousi, M. Tornatore, A. Mitra, B. C. Chatterjee, and B. Mukherjee. "Dynamic routing, spectrum, and modulation-format allocation in mixed-grid optical networks." In: *Journal of Optical Communications and Networking* 12.5 (2020), pp. 79–88.

[25] S. Ahvar, H. P. Phyu, S. M. Buddhacharya, E. Ahvar, N. Crespi, and R. Glitho. "CCVP: Cost-efficient centrality-based VNF placement and chaining algorithm for network service provisioning." In: *2017 IEEE Conference on Network Softwarization (NetSoft)*. IEEE. 2017, pp. 1–9.

[26] *Airbus A320 CEO*. https://goo.gl/p6ZT8k. 2018.

[27] A. Aissioui, A. Ksentini, A. M. Gueroui, and T. Taleb. "On enabling 5G automotive systems using follow me edge-cloud concept." In: *IEEE Transactions on Vehicular Technology* 67.6 (2018), pp. 5302–5316.

[28] N. Alliance. "5G white paper." In: *Next generation mobile networks, white paper* (2015), pp. 1–125.

[29] *Amazon EC2 Reserved Instances Pricing*. https://goo.gl/dWmqTg. 2018.

[30] R. P. Anstee. "A polynomial algorithm for b-matchings: an alternative approach." In: *Information Processing Letters* 24.3 (1987), pp. 153–157.

[31] Y. Anzai. *Pattern recognition and machine learning*. Elsevier, 2012.

[32] O. Atalik, M. Bakir, and S. Akan. "The role of in-flight service quality on value for money in business class: a logit model on the airline industry." In: *Administrative Sciences* 9.1 (2019), p. 26.

[33] P. Auer, N. Cesa-Bianchi, and P. Fischer. "Finite-time analysis of the multiarmed bandit problem." In: *Machine learning* 47.2 (2002), pp. 235–256.

[34] G. Avino et al. "A MEC-based extended virtual sensing for automotive services." In: *IEEE Transactions on Network and Service Management* 16.4 (2019), pp. 1450–1463.

[35] T. Bahreini and D. Grosu. "Efficient placement of multi-component applications in edge computing systems." In: *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*. 2017, pp. 1–11.

[36] F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba, and O. C. M. B. Duarte. "Orchestrating virtualized network functions." In: *IEEE Transactions on Network and Service Management* 13.4 (2016), pp. 725–739.

[37] J. Barr. *Choosing the Right EC2 Instance Type for Your Application*. https://goo.gl/V4CGjH. 2018.

[38] L. A. Barroso, U. Hölzle, and P. Ranganathan. "The datacenter as a computer: Designing warehouse-scale machines." In: *Synthesis Lectures on Computer Architecture* 13.3 (2018), pp. i–189.

[39] J. Baste, B.-M. Bui-Xuan, and A. Roux. "Temporal matching." In: *Theoretical Computer Science* 806 (2020), pp. 184–196.

[40] A. Beloglazov, J. Abawajy, and R. Buyya. "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing." In: *Future generation computer systems* 28.5 (2012), pp. 755–768.

[41] A. Beloglazov, R. Buyya, Y. C. Lee, and A. Zomaya. "A taxonomy and survey of energy-efficient data centers and cloud computing systems." In: *Advances in computers*. Vol. 82. Elsevier, 2011, pp. 47–111.

[42] C. Berge. "Two theorems in graph theory." In: *Proceedings of the National Academy of Sciences of the United States of America* 43.9 (1957), p. 842.

[43] A. Berl, E. Gelenbe, M. Di Girolamo, G. Giuliani, H. De Meer, M. Q. Dang, and K. Pentikousis. "Energy-efficient cloud computing." In: *The computer journal* 53.7 (2010), pp. 1045–1051.

[44] D. Bhamare, R. Jain, M. Samaka, and A. Erbad. "A survey on Service Function Chaining." In: *Journal of Network and Computer Applications* 75 (2016), pp. 138–155.

[45] S. J. Bigelow. *Computing with a price tag: VM cost calculation guide*. URL: https://searchservervirtualization.techtarget.com/feature/Computing-with-a-price-tag-VM-cost-calculation-guide.

[46] B. Birnbaum and C. Mathieu. "On-line bipartite matching made simple." In: *Acm Sigact News* 39.1 (2008), pp. 80–87.

[47] R. Bolla, R. Bruschi, A. Carrega, F. Davoli, D. Suino, C. Vassilakis, and A. Zafeiropoulos. "Cutting the energy bills of Internet Service Providers and telecoms through power management: An impact analysis." In: *Computer Networks* 56.10 (2012), pp. 2320–2342.

[48] M. Bonamy, N. Bousquet, M. Heinrich, T. Ito, Y. Kobayashi, A. Mary, M. Mühlenthaler, and K. Wasa. "The perfect matching reconfiguration problem." In: *arXiv preprint arXiv:1904.06184* (2019).

[49] S. P. Borgatti and M. G. Everett. "A graph-theoretic perspective on centrality." In: *Social networks* 28.4 (2006), pp. 466–484.

[50] D. Boru, D. Kliazovich, F. Granelli, P. Bouvry, and A. Y. Zomaya. "Energy-efficient data replication in cloud computing datacenters." In: *Cluster computing* 18.1 (2015), pp. 385–402.

[51] B. Bosek, D. Leniowski, P. Sankowski, and A. Zych. "Online bipartite matching in offline time." In: *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*. IEEE. 2014, pp. 384–393.

[52] B. Boudreau. *Global Bandwidth & IP Pricing Trends*. https://goo.gl/V4CGjH. 2017.

[53] M. Bouet and V. Conan. "Mobile edge computing resources optimization: A geo-clustering approach." In: *IEEE Transactions on Network and Service Management* 15.2 (2018), pp. 787–796.

[54] S. Brin and L. Page. "The anatomy of a large-scale hypertextual web search engine." In: (1998).

[55] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. "Openai gym." In: *arXiv preprint arXiv:1606.01540* (2016).

[56] S. Burroughs, H. Dickel, M. van Zijl, V. Podolskiy, M. Gerndt, R. Malik, and P. Patros. "Towards Autoscaling with Guarantees on Kubernetes Clusters." In: *2021 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion (ACSOS-C)*. IEEE. 2021, pp. 295–296.

[57] Y. Cao, H. Guo, J. Liu, and N. Kato. "Optimal satellite gateway placement in space-ground integrated networks." In: *IEEE Network* 32.5 (2018), pp. 32–37.

[58] Y. Cao, Y. Shi, J. Liu, and N. Kato. "Optimal satellite gateway placement in space-ground integrated network for latency minimization with reliability guarantee." In: *IEEE Wireless Communications Letters* 7.2 (2017), pp. 174–177.

[59] J. C. Cardona Restrepo, C. Gruber, and C. Mas-Machuca. "Energy Profile Aware Routing." In: *First Int. Workshop on Green Communications IEEE Int. Conference on Communications (ICC)*. 2009.

[60] D. G. Cattrysse and L. N. Van Wassenhove. "A survey of algorithms for the generalized assignment problem." In: *European journal of operational research* 60.3 (1992), pp. 260–272.

[61] J. Chabarek, J. Sommers, P. Barford, C. Estan, D. Tsiang, and S. Wright. "Power awareness in network design and routing." In: *IEEE INFOCOM 2008-The 27th Conference on Computer Communications*. IEEE. 2008, pp. 457–465.

[62] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doyle. "Managing energy and server resources in hosting centers." In: *ACM SIGOPS operating systems review* 35.5 (2001), pp. 103–116.

[63] G. Chen, W. He, J. Liu, S. Nath, L. Rigas, L. Xiao, and F. Zhao. "Energy-Aware Server Provisioning and Load Dispatching for Connection-Intensive Internet Services." In: *NSDI*. Vol. 8. 2008, pp. 337–350.

[64] Y.-H. Chen, T.-L. Chin, C.-Y. Huang, S.-H. Shen, and R.-Y. Huang. "Time Efficient Energy-Aware Routing in Software Defined Networks." In: *2018 IEEE 7th International Conference on Cloud Networking (CloudNet)*. IEEE. 2018, pp. 1–7.

[65] W. Chen, X. Yin, Z. Wang, and X. Shi. "Placement and Routing Optimization Problem for Service Function Chain: State of Art and Future Opportunities." In: *arXiv preprint arXiv:1910.02613* (2019).

[66] M. Chimani, N. Troost, and T. Wiedera. "Approximating multistage matching problems." In: *International Workshop on Combinatorial Algorithms*. Springer. 2021, pp. 558–570.

[67] M. Chiosi et al. *Network Functions Virtualisation—Introductory White Paper Network*. 2012.

[68] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz. "Near optimal placement of virtual network functions." In: *2015 IEEE Conference on Computer Communications (INFOCOM)*. IEEE. 2015, pp. 1346–1354.

[69] O. D. S. F. Community. *Open Daylight SDN Framework*. https://www.opendaylight.org/. 2021.

[70] R. S. F. Community. *Ryu SDN Framework*. https://osrg.github.io/ryu/. 2021.

[71] P. T. Congdon, P. Mohapatra, M. Farrens, and V. Akella. "Simultaneously reducing latency and power consumption in openflow switches." In: *IEEE/ACM Transactions On Networking* 22.3 (2013), pp. 1007–1020.

[72] P. Corcoran and A. Andrae. "Emerging trends in electricity consumption for consumer ICT." In: *National University of Ireland, Galway, Connacht, Ireland, Tech. Rep* (2013).

[73] G. Cornuéjols, R. Sridharan, and J.-M. Thizy. "A comparison of heuristics and relaxations for the capacitated plant location problem." In: *European journal of operational research* 50.3 (1991), pp. 280–297.

[74] F. Cugini et al. "Push-pull defragmentation without traffic disruption in flexible grid optical networks." In: *Journal of Lightwave Technology* 31.1 (2012), pp. 125–133.

[75] R. Cziva, C. Anagnostopoulos, and D. P. Pezaros. "Dynamic, latency-optimal VNF placement at the network edge." In: *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE. 2018, pp. 693–701.

[76] *Datacenter Map*. https://www.datacentermap.com/cloud.html. 2020.

[77] M. Dayarathna, Y. Wen, and R. Fan. "Data center energy consumption modeling: A survey." In: *IEEE Communications Surveys & Tutorials* 18.1 (2015), pp. 732–794.

[78] S. De Maesschalck et al. "Pan-European optical transport networks: An availability-based comparison." In: *Photonic Network Communications* 5.3 (2003), pp. 203–225.

[79] N. R. Devanur and K. Jain. "Online matching with concave returns." In: *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*. 2012, pp. 137–144.

[80] M. Dieye, S. Ahvar, J. Sahoo, E. Ahvar, R. Glitho, H. Elbiaze, and N. Crespi. "CPVNF: Cost-efficient proactive VNF placement and chaining for value-added services in content delivery networks." In: *IEEE Transactions on Network and Service Management* 15.2 (2018), pp. 774–786.

[81] E. W. Dijkstra et al. "A note on two problems in connexion with graphs." In: *Numerische mathematik* 1.1 (1959), pp. 269–271.

[82] E. Dinc, M. Vondra, and C. Cavdar. "Multi-user Beamforming and Ground Station Deployment for 5G Direct Air-to-Ground Communication." In: *GLOBECOM 2017-2017 IEEE Global Communications Conference*. IEEE. 2017, pp. 1–7.

[83] E. Dinc et al. "In-flight broadband connectivity: Architectures and business models for high capacity air-to-ground communications." In: *IEEE Communications Magazine* 55.9 (2017), pp. 142–149.

[84] A. Dwaraki and T. Wolf. "Adaptive service-chain routing for virtual network functions in software-defined networks." In: *Proceedings of the 2016 workshop on Hot topics in Middleboxes and Network Function Virtualization*. 2016, pp. 32–37.

[85] N. El Khoury, S. Ayoubi, and C. Assi. "Energy-aware placement and scheduling of network traffic flows with deadlines on virtual network functions." In: *2016 5th IEEE International Conference on Cloud Networking (Cloudnet)*. IEEE. 2016, pp. 89–94.

[86] J. Elmirghani, T. Klein, K. Hinton, L. Nonde, A. Lawey, T. El-Gorashi, M. Musa, and X. Dong. "GreenTouch GreenMeter core network energy-efficiency improvement measures and optimization." In: *Journal of Optical Communications and Networking* 10.2 (2018), A250–A269.

[87] V. Eramo, M. Ammar, and F. G. Lavacca. "Migration energy aware reconfigurations of virtual network function instances in NFV architectures." In: *IEEE Access* 5 (2017), pp. 4927–4938.

[88] V. Eramo, E. Miucci, M. Ammar, and F. G. Lavacca. "An approach for service function chain routing and virtual function network instance migration in network function virtualization architectures." In: *IEEE/ACM Transactions on Networking* 25.4 (2017), pp. 2008–2025.

[89] *ETSI TS 123.203 V13.6.0*. https://goo.gl/KvekZM. 2016.

[90] *European Aviation Network (EAN)*. http://www.europeanaviasionnetwork.com/. 2020.

[91] B. Evans, M. Werner, E. Lutz, M. Bousquet, G. E. Corazza, G. Maral, and R. Rumeau. "Integration of satellite and terrestrial systems in future multimedia communications." In: *IEEE Wireless Communications* 12.5 (2005), pp. 72–80.

[92] A. Fallahpour, H. Beyranvand, S. A. Nezamalhosseini, and J. A. Salehi. "Energy efficient routing and spectrum assignment with regenerator placement in elastic optical networks." In: *Journal of Lightwave Technology* 32.10 (2014), pp. 2019–2027.

[93] X. Fan, W.-D. Weber, and L. A. Barroso. "Power provisioning for a warehouse-sized computer." In: *ACM SIGARCH computer architecture news* 35.2 (2007), pp. 13–23.

[94] W. Fang, M. Zeng, X. Liu, W. Lu, and Z. Zhu. "Joint spectrum and IT resource allocation for efficient vNF service chaining in inter-datacenter elastic optical networks." In: *IEEE Communications Letters* 20.8 (2016), pp. 1539–1542.

[95] V. Farhadi, F. Mehmeti, T. He, T. F. La Porta, H. Khamfroush, S. Wang, K. S. Chan, and K. Poularakis. "Service placement and request scheduling for data-intensive applications in edge clouds." In: *IEEE/ACM Transactions on Networking* 29.2 (2021), pp. 779–792.

[96] B. Farkiani, B. Bakhshi, and S. A. Mirhassani. "A Fast Near-Optimal Approach for Energy-Aware SFC Deployment." In: *IEEE Transactions on Network and Service Management* 16.4 (2019), pp. 1360–1373.

[97] I. Farris, T. Taleb, M. Bagaa, and H. Flick. "Optimizing service replication for mobile delay-sensitive applications in 5G edge network." In: *2017 IEEE International Conference on Communications (ICC)*. IEEE. 2017, pp. 1–6.

[98] *Fast Internet for aircraft in Europe*. https://goo.gl/KUFGGL. 2018.

[99] X. Fei, F. Liu, H. Xu, and H. Jin. "Adaptive VNF scaling and flow routing with proactive demand prediction." In: *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE. 2018, pp. 486–494.

[100] M. L. Fisher, R. Jaikumar, and L. N. Van Wassenhove. "A multiplier adjustment method for the generalized assignment problem." In: *Management science* 32.9 (1986), pp. 1095–1103.

[101] A. Gandhi, M. Harchol-Balter, R. Das, and C. Lefurgy. "Optimal power allocation in server farms." In: vol. 37. 1. ACM New York, NY, USA, 2009, pp. 157–168.

[102] B. Gao, Z. Zhou, F. Liu, and F. Xu. "Winning at the starting line: Joint network selection and service placement for mobile edge computing." In: *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE. 2019, pp. 1459–1467.

[103] M. R. Garey and D. S. Johnson. *Computers and intractability*. Vol. 174. freeman San Francisco, 1979.

[104] M. Ghaznavi, N. Shahriar, S. Kamali, R. Ahmed, and R. Boutaba. "Distributed service function chaining." In: *IEEE Journal on Selected Areas in Communications* 35.11 (2017), pp. 2479–2489.

[105] A. Giorgetti, N. Sambo, I. Cerutti, N. Andriolli, and P. Castoldi. "Label preference schemes for lightpath provisioning and restoration in distributed GMPLS networks." In: *Journal of lightwave technology* 27.6 (2009), pp. 688–697.

[106] P. Y. Glorennec and L. Jouffe. "Fuzzy Q-learning." In: *Proceedings of 6th international fuzzy systems conference*. Vol. 2. IEEE. 1997, pp. 659–662.

[107] *GoGo, The Inflight Internet Company*. https://gogoair.com/. 2020.

[108] R. Gouareb, V. Friderikos, and A.-H. Aghvami. "Virtual network functions routing and placement for edge cloud latency minimization." In: *IEEE Journal on Selected Areas in Communications* 36.10 (2018), pp. 2346–2357.

[109] E. Grigoreva. "Strategic Network Planning for Converged Optical Networks." Dissertation. München: Technische Universität München, 2020.

[110] J. W. Guck, A. Van Bemten, M. Reisslein, and W. Kellerer. "Unicast QoS routing algorithms for SDN: A comprehensive survey and performance evaluation." In: *IEEE Communications Surveys & Tutorials* 20.1 (2017), pp. 388–415.

[111] A. Gupta, M. F. Habib, P. Chowdhury, M. Tornatore, and B. Mukherjee. "Joint virtual network function placement and routing of traffic in operator networks." In: *UC Davis, Davis, CA, USA, Tech. Rep* (2015).

[112] A. Gupta, B. Jaumard, M. Tornatore, and B. Mukherjee. "A scalable approach for service Chain mapping with multiple SC instances in a wide-area network." In: *IEEE Journal on Selected Areas in Communications* 36.3 (2018), pp. 529–541.

[113] *Gurobi Optimization Solver*. https://www.gurobi.com/. 2021.

[114] P. Hall. "On representatives of subsets." In: Springer, 1987, pp. 58–62.

[115] J. Halpern and C. Pignataro. "RFC 7665: Service Function Chaining (SFC) Architecture." In: *IETF Datatracker* (2015).

[116] J. Halpern, C. Pignataro, et al. "Service function chaining (sfc) architecture." In: *RFC 7665*. 2015.

[117] P. E. Hart, N. J. Nilsson, and B. Raphael. "A formal basis for the heuristic determination of minimum cost paths." In: *IEEE transactions on Systems Science and Cybernetics* 4.2 (1968), pp. 100–107.

[118] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown. "Elastictree: Saving energy in data center networks." In: *Nsdi*. Vol. 10. 2010, pp. 249–264.

[119] M. Henzinger, S. Khan, R. Paul, and C. Schulz. "Dynamic Matching Algorithms in Practice." In: *arXiv preprint arXiv:2004.09099* (2020).

[120] C. Hepburn. "Regulating by prices, quantities or both: an update and an overview." In: *Oxford Review of Economic Policy* 22.2 (2006), pp. 226–247.

[121] J. G. Herrera and J. F. Botero. "Resource allocation in NFV: A comprehensive survey." In: *IEEE Transactions on Network and Service Management* 13.3 (2016), pp. 518–532.

[122] D. S. Hochba. "Approximation algorithms for np-hard problems." In: *ACM Sigact News* 28.2 (1997), pp. 40–52.

[123] F. Hoffmann. "Routing and internet gateway selection in aeronautical ad hoc networks." In: (2015).

[124] S. Hofmann, A. E. Garcia, D. Schupke, H. E. Gonzalez, and F. H. Fitzek. "Connectivity in the air: Throughput analysis of air-to-ground systems." In: *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE. 2019, pp. 1–6.

[125] H. Huang, S. Guo, J. Wu, and J. Li. "Green datapath for TCAM-based software-defined networks." In: *IEEE Communications Magazine* 54.11 (2016), pp. 194–201.

[126] M. Huang, W. Liang, X. Shen, Y. Ma, and H. Kan. "Reliability-Aware Virtualized Network Function Services Provisioning in Mobile Edge Computing." In: *IEEE Transactions on Mobile Computing* (2019).

[127] *HughesNet Internet*. https://goo.gl/VoLzci. 2018.

[128] N. Huin, A. Tomassilli, F. Giroire, and B. Jaumard. "Energy-efficient service function chain provisioning." In: *Journal of Optical Communications and Networking* 10.3 (2018), pp. 114–124.

[129] IEA. "Digitalization and Energy." In: *Int. Energy Agency*. Nov. 2017.

[130] I. Jang, D. Suh, S. Pack, and G. Dán. "Joint optimization of service function placement and flow distribution for service function chaining." In: *IEEE Journal on Selected Areas in Communications* 35.11 (2017), pp. 2532–2541.

[131] C. Jiang, Y. Wang, D. Ou, B. Luo, and W. Shi. "Energy proportional servers: Where are we in 2016?" In: *2017 IEEE 37th Int. Conference on Distributed Computing Systems (ICDCS)*. IEEE. 2017, pp. 1649–1660.

[132] A. Juttner, B. Szviatovski, I. Mécs, and Z. Rajkó. "Lagrange relaxation based method for the QoS routing problem." In: *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No. 01CH37213)*. Vol. 2. IEEE. 2001, pp. 859–868.

[133] B. Kalyanasundaram and K. Pruhs. "Online weighted matching." In: *Journal of Algorithms* 14.3 (1993), pp. 478–488.

[134] B. Kalyanasundaram and K. R. Pruhs. "An optimal deterministic algorithm for online b-matching." In: *Theoretical Computer Science* 233.1-2 (2000), pp. 319–325.

[135] K. Kannan and S. Banerjee. "Compact TCAM: Flow entry compaction in TCAM for power aware SDN." In: *International conference on distributed computing and networking*. Springer. 2013, pp. 439–444.

[136] B. Kar, E. H.-K. Wu, and Y.-D. Lin. "Energy cost optimization in dynamic placement of virtualized network function chains." In: *IEEE Transactions on Network and Service Management* 15.1 (2017), pp. 372–386.

[137] C. Karande, A. Mehta, and P. Tripathi. "Online bipartite matching with unknown distributions." In: *Proceedings of the forty-third annual ACM symposium on Theory of computing*. 2011, pp. 587–596.

[138] M. Karimzadeh-Farshbafan, V. Shah-Mansour, and D. Niyato. "A Dynamic Reliability-Aware Service Placement for Network Function Virtualization (NFV)." In: *IEEE Journal on Selected Areas in Communications* (2019).

[139] R. M. Karp, U. V. Vazirani, and V. V. Vazirani. "An optimal algorithm for on-line bipartite matching." In: *Proceedings of the twenty-second annual ACM symposium on Theory of computing*. 1990, pp. 352–358.

[140] C. V. Karsten, D. Pisinger, S. Ropke, and B. D. Brouer. "The time constrained multi-commodity network flow problem and its application to liner shipping network design." In: *Transportation Research Part E: Logistics and Transportation Review* 76 (2015), pp. 122–138.

[141] S. K. Kasi et al. "Heuristic edge server placement in Industrial Internet of Things and cellular networks." In: *IEEE Internet of Things Journal* (2020).

[142] K. Kaur, V. Mangat, and K. Kumar. "A Study of OpenStack Networking and Auto-Scaling Using Heat Orchestration Template." In: *Intelligent Computing and Communication Systems*. Springer, 2021, pp. 169–176.

[143] N. Kherraf, H. A. Alameddine, S. Sharafeddine, C. M. Assi, and A. Ghrayeb. "Optimized provisioning of edge computing resources with heterogeneous workload in IoT networks." In: *IEEE Transactions on Network and Service Management* 16.2 (2019), pp. 459–474.

[144] S. Kim, S. Park, Y. Kim, S. Kim, and K. Lee. "VNF-EQ: dynamic placement of virtual network functions for energy efficiency and QoS guarantee in NFV." In: *Cluster Computing* 20.3 (2017), pp. 2107–2117.

[145] A. Laghrissi and T. Taleb. "A survey on the placement of virtual resources and virtual network functions." In: *IEEE Communications Surveys & Tutorials* 21.2 (2018), pp. 1409–1434.

[146] Y. C. Lee and A. Y. Zomaya. "Energy efficient utilization of resources in cloud computing systems." In: *The Journal of Supercomputing* 60.2 (2012), pp. 268–280.

[147] D. Li, P. Hong, K. Xue, and J. Pei. "Availability Aware VNF Deployment in Datacenter Through Shared Redundancy and Multi-Tenancy." In: *IEEE Transactions on Network and Service Management* 16.4 (2019), pp. 1651–1664.

[148] Y.-D. Lin, Y.-C. Lai, J.-X. Huang, and H.-T. Chien. "Three-tier capacity and traffic allocation for core, edges, and devices for mobile edge computing." In: *IEEE Transactions on Network and Service Management* 15.3 (2018), pp. 923–933.

[149] H. Liu, C.-Z. Xu, H. Jin, J. Gong, and X. Liao. "Performance and energy modeling for live migration of virtual machines." In: *Proceedings of the 20th international symposium on High performance distributed computing*. ACM. 2011, pp. 171–182.

[150] J. Liu, Y. Shi, Z. M. Fadlullah, and N. Kato. "Space-air-ground integrated network: A survey." In: *IEEE Communications Surveys & Tutorials* 20.4 (2018), pp. 2714–2741.

[151] S. Liu, Z. Gao, Y. Wu, D. W. K. Ng, X. Gao, K.-K. Wong, S. Chatzinotas, and B. Ottersten. "LEO Satellite Constellations for 5G and Beyond: How Will They Reshape Vertical Domains?" In: *arXiv preprint arXiv:2106.09897* (2021).

[152] V. Lopez, L. Velasco, et al. "Elastic optical networks." In: *Architectures, Technologies, and Control, Switzerland: Springer Int. Publishing* (2016).

[153] Y. Ma, W. Liang, and S. Guo. "Mobility-aware delay-sensitive service provisioning for mobile edge computing." In: *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE. 2019, pp. 270–276.

[154] P. Mahadevan, S. Banerjee, P. Sharma, A. Shah, and P. Ranganathan. "On energy efficiency for enterprise and data center networks." In: *IEEE Communications Magazine* 49.8 (2011), pp. 94–100.

[155] P. Mahadevan, P. Sharma, S. Banerjee, and P. Ranganathan. "A power benchmarking framework for network devices." In: *Int. Conference on Research in Networking*. Springer. 2009, pp. 795–808.

[156] P. Mahadevan, P. Sharma, S. Banerjee, and P. Ranganathan. "Energy aware network operations." In: *IEEE INFOCOM Workshops 2009*. IEEE. 2009, pp. 1–6.

[157] M. Mahdian and Q. Yan. "Online bipartite matching with random arrivals: an approach based on strongly factor-revealing lps." In: *Proceedings of the forty-third annual ACM symposium on Theory of computing*. 2011, pp. 597–606.

[158] S. Malla and K. Christensen. "The effect of server energy proportionality on data center power oversubscription." In: *Future Generation Computer Systems* 104 (2020), pp. 119–130.

[159] U. Mansmann. *Fast Internet for aircraft in Europe*. https://goo.gl/KUFGGL. 2018.

[160] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. "OpenFlow: enabling innovation in campus networks." In: *ACM SIGCOMM computer communication review* 38.2 (2008), pp. 69–74.

[161] M. M. McMahon and R. Rathburn. *Measuring latency in Iridium satellite constellation data services*. Tech. rep. NAVAL ACADEMY ANNAPOLIS MD DEPT OF COMPUTER SCIENCE, 2005.

[162] A. Mehta. "Online Matching and Ad Allocation." In: *Foundations and Trends® in Theoretical Computer Science* 8.4 (2013), pp. 265–368.

[163] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. "Playing atari with deep reinforcement learning." In: *arXiv preprint arXiv:1312.5602* (2013).

[164] D. Moniz, J. Pedro, and J. Pires. "Network design framework to optimally provision services using higher-symbol rate line interfaces." In: *Journal of Optical Communications and Networking* 11.2 (2019), A174–A185.

[165] G. Nagy and S. Salhi. "Location-routing: Issues, models and methods." In: *European journal of operational research* 177.2 (2007), pp. 649–672.

[166] *NetFlix Internet Connection Speed Recommendations*. https://goo.gl/VNst2M. 2018.

[167] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi. "Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications." In: *IEEE transactions on cybernetics* 50.9 (2020), pp. 3826–3839.

[168] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski. "SNDlib 1.0—Survivable network design library." In: *Networks: An International Journal* 55.3 (2010), pp. 276–286.

[169] A. Papa, T. De Cola, P. Vizarreta, M. He, C. M. Machuca, and W. Kellerer. "Dynamic SDN controller placement in a LEO constellation satellite network." In: *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE. 2018, pp. 206–212.

[170] D. Papadimitriou, D. Colle, and P. Demeester. "Mixed integer optimization for the combined capacitated facility location-routing problem." In: *Annals of Telecommunications* 73.1 (2018), pp. 37–62.

[171] S. K. Patri. *Github, Physical Network Information*. www.github.com/SaiPatri/PhyNWInfo. 2021.

[172] S. K. Patri, A. Autenrieth, J.-P. Elbers, and C. M. Machuca. "Planning optical networks for unexpected traffic growth." In: *2020 European Conference on Optical Communications (ECOC)*. IEEE. 2020, pp. 1–4.

[173] S. K. Patri, A. Autenrieth, D. Rafique, J.-P. Elbers, and C. Mas-Machuca. "HeCSON: heuristic for configuration selection in optical network planning." In: *Optical Fiber Communication Conference*. Optical Society of America. 2020, Th2A–32.

[174] K. Pepple. *Deploying openstack*. " O'Reilly Media, Inc.", 2011.

[175] C. Pham, N. H. Tran, S. Ren, W. Saad, and C. S. Hong. "Traffic-aware and energy-efficient VNF placement for service chaining: Joint sampling and matching approach." In: *IEEE Transactions on Services Computing* (2017).

[176] F. L. Pires and B. Barán. "A virtual machine placement taxonomy." In: *2015 15th IEEE/ACM Int. Symposium on Cluster, Cloud and Grid Computing*. IEEE. 2015, pp. 159–168.

[177] J. Plachy, Z. Becvar, E. C. Strinati, and N. di Pietro. "Dynamic Allocation of Computing and Communication Resources in Multi-Access Edge Computing for Mobile Users." In: *IEEE Transactions on Network and Service Management* (2021).

[178] P. Poggiolini, M. R. Zefreh, G. Bosco, F. Forghieri, and S. Piciaccia. "Accurate non-linearity fully-closed-form formula based on the GN/EGN model and large-data-set fitting." In: *Optical Fiber Communication Conference*. Optical Society of America. 2019, pp. M1I–4.

[179] L. Qu, C. Assi, M. Khabbaz, and Y. Ye. "Reliability-Aware Service Function Chaining With Function Decomposition and Multipath Routing." In: *IEEE Transactions on Network and Service Management* (2019).

[180] Z. Qu, G. Zhang, H. Cao, and J. Xie. "LEO satellite constellation for Internet of Things." In: *IEEE access* 5 (2017), pp. 18391–18401.

[181] S. Ricciardi, F. Palmieri, U. Fiore, D. Careglio, G. Santos-Boada, and J. Solé-Pareta. "An energy-aware dynamic RWA framework for next-generation wavelength-routed networks." In: *Computer Networks* 56.10 (2012), pp. 2420–2442.

[182] F. A. Rodrigues. "Network centrality: an introduction." In: *A Mathematical Modeling Approach from Nonlinear Dynamics to Complex Systems*. Springer, 2019, pp. 177–196.

[183] M. Rost and S. Schmid. "Charting the complexity landscape of virtual network embeddings." In: *2018 IFIP Networking Conference (IFIP Networking) and Workshops*. IEEE. 2018, pp. 1–9.

[184] M. Rost and S. Schmid. "Virtual network embedding approximations: Leveraging randomized rounding." In: *IEEE/ACM Transactions on Networking* 27.5 (2019), pp. 2071–2084.

[185] C. Rottondi, P. Boffi, P. Martelli, and M. Tornatore. "Routing, modulation format, baud rate and spectrum allocation in optical metro rings with flexible grid and few-mode transmission." In: *Journal of Lightwave Technology* 35.1 (2016), pp. 61–70.

[186] G. Sallam, G. R. Gupta, B. Li, and B. Ji. "Shortest path and maximum flow problems under service function chaining constraints." In: *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE. 2018, pp. 2132–2140.

[187] Y. Sang, B. Ji, G. R. Gupta, X. Du, and L. Ye. "Provably efficient algorithms for joint placement and allocation of virtual network functions." In: *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*. IEEE. 2017, pp. 1–9.

[188] G. Sayfan. *Mastering kubernetes*. Packt Publishing Ltd, 2017.

[189] T. Schaul, J. Quan, I. Antonoglou, and D. Silver. "Prioritized experience replay." In: *arXiv preprint arXiv:1511.05952* (2015).

[190] J. Sherry, S. Ratnasamy, and J. S. At. "A survey of enterprise middlebox deployments." In: (2012).

[191] Y. Shi and J. Liu. "Inter-Segment Gateway Selection for Transmission Energy Optimization in Space-Air-Ground Converged Network." In: *2018 IEEE International Conference on Communications (ICC)*. IEEE. 2018, pp. 1–6.

[192] Y. Shin, K. Kim, S. Lee, and H.-C. An. "Online Graph Matching Problems with a Worst-Case Reassignment Budget." In: *arXiv preprint arXiv:2003.05175* (2020).

[193] O. Soualah, M. Mechtri, C. Ghribi, and D. Zeghlache. "Energy efficient algorithm for VNF placement and chaining." In: *2017 17th IEEE/ACM Int. Symposium on Cluster, Cloud and Grid Computing (CCGRID)*. IEEE. 2017, pp. 579–588.

[194] "SpecPOWER." In: (2020).

[195] A. Strunk. "Costs of virtual machine live migration: A survey." In: *Services (SERVICES), 2012 IEEE Eighth World Congress on*. IEEE. 2012, pp. 323–329.

[196] R. Summerhill. "The new Internet2 network." In: *6th GLIF Meeting*. 2006.

[197] G. Sun, Y. Li, H. Yu, A. V. Vasilakos, X. Du, and M. Guizani. "Energy-efficient and traffic-aware service function chaining orchestration in multi-domain networks." In: *Future Generation Computer Systems* 91 (2019), pp. 347–360.

[198] M. M. Tajiki, S. Salsano, L. Chiaraviglio, M. Shojafar, and B. Akbari. "Joint energy efficient and QoS-aware path allocation and VNF placement for service function chaining." In: *IEEE Transactions on Network and Service Management* 16.1 (2018), pp. 374–388.

[199] T. Taleb, M. Bagaa, and A. Ksentini. "User mobility-aware virtual network function placement for virtual 5G network infrastructure." In: *2015 IEEE International Conference on Communications (ICC)*. IEEE. 2015, pp. 3879–3884.

[200] T. Taleb, A. Ksentini, and P. A. Frangoudis. "Follow-me cloud: When cloud services follow mobile users." In: *IEEE Transactions on Cloud Computing* 7.2 (2016), pp. 369–382.

[201] F. Tashtarian, M. F. Zhani, B. Fatemipour, and D. Yazdani. "CoDeC: a cost-effective and delay-aware SFC deployment." In: *IEEE Transactions on Network and Service Management* (2019).

[202] *The Dream of Affordable Internet Access for Everyone is Getting Closer*. https://goo.gl/eTkRnL. 2017.

[203] A. Tomassilli, F. Giroire, N. Huin, and S. Pérennes. "Provably efficient algorithms for placement of service function chains with ordering constraints." In: *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE. 2018, pp. 774–782.

[204] D. Tomić, S. Hofmann, M. Ozger, D. Schupke, and C. Cavdar. "Quality of Service Aware Traffic Management for Aircraft Communications." In: *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*. IEEE. 2020, pp. 1–6.

[205] M. F. Tuysuz, Z. K. Ankarali, and D. Gözüpek. "A survey on energy efficiency in software defined networks." In: *Computer Networks* 113 (2017), pp. 188–204.

[206] A. Van Bemten, J. W. Guck, C. Mas-Machuca, and W. Kellerer. "Routing metrics depending on previous edges: The Mn taxonomy and its corresponding solutions." In: *2018 IEEE Int. Conference on Communications (ICC)*. IEEE. 2018, pp. 1–7.

[207] A. Van Bemten, J. W. Guck, P. Vizarreta, C. Mas-Machuca, and W. Kellerer. "LARAC-SN and Mole in the Hole: Enabling routing through service function chains." In: *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*. IEEE. 2018, pp. 298–302.

[208] A. Van Bemten and W. Kellerer. "Network calculus: A comprehensive guide." In: (2016).

[209] H. Van Hasselt, A. Guez, and D. Silver. "Deep reinforcement learning with double q-learning." In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 30. 1. 2016.

[210] W. Van Heddeghem, F. Idzikowski, W. Vereecken, D. Colle, M. Pickavet, and P. Demeester. "Power consumption modeling in optical multilayer networks." In: *Photonic Network Communications* 24.2 (2012), pp. 86–102.

[211] A. Varasteh and M. Goudarzi. "Server consolidation techniques in virtualized data centers: A survey." In: *IEEE Systems Journal* 11.2 (2017), pp. 772–783.

[212] S. S. Vegesna, A. C. Nara, and N. M. Sk. "A Novel Rule Mapping on TCAM for Power Efficient Packet Classification." In: *ACM Transactions on Design Automation of Electronic Systems (TODAES)* 24.5 (2019), pp. 1–23.

[213] P. Vizarreta, M. Condoluci, C. Mas-Machuca, T. Mahmoodi, and W. Kellerer. "QoS-driven function placement reducing expenditures in NFV deployments." In: *IEEE Int. Conference on Communications*. 2017.

[214] M. Vondra, E. Dinc, M. Prytz, M. Frodigh, D. Schupke, M. Nilson, S. Hofmann, and C. Cavdar. "Performance study on seamless DA2GC for aircraft passengers toward 5G." In: *IEEE Communications Magazine* 55.11 (2017), pp. 194–201.

[215] L. Wang, L. Jiao, J. Li, and M. Mühlhäuser. "Online resource allocation for arbitrary user mobility in distributed edge clouds." In: *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE. 2017, pp. 1281–1290.

[216] S. Wang, J. Xu, N. Zhang, and Y. Liu. "A survey on service migration in mobile edge computing." In: *IEEE Access* 6 (2018), pp. 23511–23528.

[217] S. Wang, R. Urgaonkar, T. He, K. Chan, M. Zafer, and K. K. Leung. "Dynamic service placement for mobile micro-clouds with predicted future costs." In: *IEEE Transactions on Parallel and Distributed Systems* 28.4 (2016), pp. 1002–1016.

[218] S. Wang, R. Urgaonkar, T. He, K. Chan, M. Zafer, and K. K. Leung. "Dynamic service placement for mobile micro-clouds with predicted future costs." In: *IEEE Transactions on Parallel and Distributed Systems* 28.4 (2016), pp. 1002–1016.

[219] X. Wang, Y. Yao, X. Wang, K. Lu, and Q. Cao. "Carpo: Correlation-aware power optimization in data center networks." In: *2012 Proceedings IEEE INFOCOM*. IEEE. 2012, pp. 1125–1133.

[220] M. Waqas, Y. Niu, M. Ahmed, Y. Li, D. Jin, and Z. Han. "Mobility-aware fog computing in dynamic environments: Understandings and implementation." In: *IEEE Access* 7 (2018), pp. 38867–38879.

[221] Q. Wu, F. Ishikawa, Q. Zhu, and Y. Xia. "Energy and migration cost-aware dynamic virtual machine consolidation in heterogeneous cloud datacenters." In: *IEEE transactions on Services Computing* 12.4 (2016), pp. 550–563.

[222] F. Xia, L. T. Yang, L. Wang, and A. Vinel. "Internet of things." In: *International journal of communication systems* 25.9 (2012), p. 1101.

[223] Y. Xiong, J. Shi, Y. Yang, Y. Lv, and G. N. Rouskas. "Lightpath management in SDN-based elastic optical networks with power consumption considerations." In: *Journal of Lightwave Technology* 36.9 (2017), pp. 1650–1660.

[224] J. Xu, X. Li, X. Liu, C. Zhang, L. Fan, L. Gong, and J. Li. "Mobility-aware workflow offloading and scheduling strategy for mobile edge computing." In: *International Conference on Algorithms and Architectures for Parallel Processing*. Springer. 2019, pp. 184–199.

[225] K. Yang, H. Zhang, and P. Hong. "Energy-aware service function placement for service function chaining in data centers." In: *2016 IEEE Global Communications Conference (GLOBECOM)*. IEEE. 2016, pp. 1–6.

[226] H. Yao, C. Bai, D. Zeng, Q. Liang, and Y. Fan. "Migrate or not? Exploring virtual machine migration in roadside cloudlet-based vehicular cloud." In: *Concurrency and Computation: Practice and Experience* 27.18 (2015), pp. 5780–5792.

[227] F. Zhang, G. Liu, X. Fu, and R. Yahyapour. "A survey on virtual machine migration: Challenges, techniques, and open issues." In: *IEEE Communications Surveys & Tutorials* 20.2 (2018), pp. 1206–1243.

[228] J. Zhang, J. Lu, X. Yan, X. Xu, L. Qi, and W. Dou. "Quantified Edge Server Placement With Quantum Encoding in Internet of Vehicles." In: *IEEE Transactions on Intelligent Transportation Systems* (2021).

[229] N. Zhang, S. Zhang, P. Yang, O. Alhussein, W. Zhuang, and X. S. Shen. "Software defined space-air-ground integrated vehicular networks: Challenges and solutions." In: *IEEE Communications Magazine* 55.7 (2017), pp. 101–109.

[230] X. Zhang, C. Wu, Z. Li, and F. C. Lau. "Proactive vnf provisioning with multi-timescale cloud resources: Fusing online learning and online optimization." In: *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*. IEEE. 2017, pp. 1–9.

[231] Y. Zhang, B. Anwer, V. Gopalakrishnan, B. Han, J. Reich, A. Shaikh, and Z.-L. Zhang. "Parabox: Exploiting parallelism for virtual network functions in service chaining." In: *Proceedings of the Symposium on SDN Research*. 2017, pp. 143–149.

[232] X. Zhao, Y. Shi, and S. Chen. "MAESP: Mobility aware edge service placement in mobile edge networks." In: *Computer Networks* 182 (2020), p. 107435.

[233] Z. Zhou, J. Feng, C. Zhang, Z. Chang, Y. Zhang, and K. M. S. Huq. "SAGECELL: Software-defined space-air-ground integrated moving cells." In: *IEEE Communications Magazine* 56.8 (2018), pp. 92–99.

# List of Figures

# List of Tables