In-Network Processing of Medical Applications in Emerging 6G Networks Considering Migration

Nicolai Kröger[1], Hongyu Zhu[1], Franziska Jurosch[3], Sven Kolb[3], Dirk Wilhelm[3,4,5], Wolfgang Kellerer[1], Fidan Mehmeti[1]

[1]*Technical University of Munich, TUM School of Computation, Information and Technology, Chair of Communication Networks*, firstname.lastname@tum.de

[3]*Technical University of Munich, TUM School of Medicine and Health, TUM University Hospital, Research Group MITI*

[4]*Technical University of Munich, TUM School of Medicine and Health, TUM University Hospital, Department of Surgery*

[5]*Technical University of Munich, Munich Institute of Robotics and Machine Intelligence*, firstname.lastname@tum.de

***Abstract*—Medical applications, such as telemedicine or smart operation rooms, pose stringent requirements on the underlying network architecture. 6G as the next-generation communication standard promises to satisfy these needs through advances in technology and networking concepts such as in-network computing. By placing applications on processing nodes at different locations within the network, network and application performance metrics can be optimized. However, obtaining an optimized placement can be challenging. Transferring application functions between Processing Units (PU) to decrease placement costs and to increase performance involves migration costs, especially when the available processing and networking capabilities are not sufficient for all requested medical applications. To consider migration costs with the optimization of processing medical applications within the network, we formulate an Integer Linear Program (ILP) in this paper. We consider different service levels of Modular Application Functions (MAF) and aim to place as many applications as possible with the best possible service quality and the lowest possible placement and migration costs. Furthermore, we propose a heuristic to obtain a suitable solution quickly. The evaluation of our solution and comparison to existing approaches shows an increase of demand admissions in the network by up to 40%.**

***Index Terms*—6G, Migration, In-Network Computing, Medical Technology.**

I. Introduction

The transition from 5G to the next generation of communication networks, 6G, promises to satisfy the requirements of emerging applications. This advance is not only enabled by the incremental improvement in the network architecture, but also by the comprehensive integration of communication, data, computing, and Artificial Intelligence (AI). Exemplary driving applications are multi-sensory Extended Reality (XR), distributed autonomous robots, and wireless brain-computer interaction with very large data rates (up to 1 Tbps), sub-millisecond ultra-low latency, large-scale distributed connectivity, and extremely high reliability. Finally, 6G promises more energy-efficient network design, more advanced data security and privacy protocols, and achieves the sustainability of global networks [1], [2].

6G is envisioned to heavily impact the medical sector. In particular, many medical applications, such as telesurgery [3], remote patient examination [4], or monitoring [5] greatly benefit from the new features in 6G. For example, 6G with its high data transmission rates, ultra-low end-to-end latency and extreme reliability, conducting remote surgeries is expected to become more feasible and safe. This allows surgeons to remotely control surgical robots in near-real time, which increases surgical precision and overall patient safety. Consequently, these surgeries are then not bound to a certain location anymore, but can be conducted world-wide, fostering the democratization of care and surgery [3].

A key aspect for such medical applications is the place where their software components are executed. This influences all important parameters such as latency or throughput. With the envisioned integration of cloud computing and in-network computing in 6G, applications can basically be executed at any place including the network itself. This high amount of possibilities enables flexibility to satisfy various network requirements. However, there are also many considerations for a potentially optimal placement to be addressed. This is especially important for the medical field, since communication issues can potentially threaten a patient's life. Aspects to be considered include but are not limited to:

- How can medical applications be placed on (potentially several) Processing Units (PUs) with different characteristics in such a way that all demands are satisfied?
- How can previous deployment and resulting migration costs be considered in an optimal way?
- What happens in scenarios where the networking resources are not enough to serve all applications?

Placement problems are already widely studied in the literature for Virtual Network Functions (VNFs) [6]–[8]. However, VNFs mainly describe network-related applications. In order to extend network functions and to support application-specific

The authors acknowledge the financial support by the Federal Ministry of Research, Technology and Space of Germany (BMFTR) in the program of "Souverän. Digital. Vernetzt." joint project 6G-life, project identification number 16KIS2414 and 16KIS2416.

ISBN 978-3-903176-82-9 © 2026 IFIP

features, the authors in [9] have introduced Modular Application Functions (MAFs). These MAFs are used to abstract medical applications and their requirements. In further work, Kröger *et al.* [10] have studied the placement of MAFs considering terminability and priority.

Using a similar approach, in this paper we investigate the MAF placement with the focus on migration. In particular, the aim is to allow as many medical applications as possible to be executed within the network with as good performance as possible while keeping the placement and migration costs as low as possible. For that, we consider different performance levels of MAFs and allow MAF migration of already executed MAFs to potentially better suited PUs. However, migration always introduces costs in terms of computing and networking resources as well as delay and therefore should only be executed if necessary. To find a suitable placement solution, we formulate an Integer Linear Program (ILP). Given the NP-hardness of the proposed ILP, we propose a heuristic to find near-optimal solutions within a reasonable time. Finally, we perform extensive simulations to evaluate the performance and compare it to state-of-the-art approaches. The main message of this paper is that with our proposed approach the overall admission ratio of applications can be increased with highest possible performance while keeping placement and migration costs low. In particular, our main contributions are:

- We formulate an ILP to optimize the admission ratio of medical applications and the system performance while reducing the costs for MAF placement and migration.
- We propose a heuristic to solve the ILP near-optimal quickly.
- We evaluate our approach with extensive simulations and also compare it to already existing algorithms to show the benefit of our approach.

The remainder of the paper is structured as follows. First, we introduce relevant background and related work in Section II. Then, we describe the model in Section III. In Section IV, we introduce the ILP, followed by the heuristic in Section V. In Section VI, we evaluate the performance of our approach. Finally, Section VII concludes the paper.

II. Related Work

The optimal placement of VNFs within the network has already been studied widely in the literature. Hereby, the focus of each individual work typically lies on a certain aspect to optimize. Considering mainly the latency of a VNF deployment, the works [11], [12], and [13] use latency-aware VNF placement and scheduling for dynamic VNF management with sensitive latency in the system. While the authors of [11] use a scheduling strategy based on optimal stopping theory to reduce violations of latency requirements, [12] and [13] mainly apply approximation and heuristic algorithms.

Other work has focused on the placement with a different goal such as energy efficiency. Agarwal *et al.* [14] propose a joint optimization method for VNF placement and CPU resource allocation in 5G networks based on queueing theory. In a different work, Tajiki *et al.* [15] introduce a joint energy optimizing VNF placement and path allocation method prioritizing cost reduction while maintaining QoS constraints. The focus of [16] lies more on the energy requirements in IoT scenarios and emphasizes the trade-off between cost and QoS degradation during migration.

VNF placement that considers various aspects of VNF migration is studied in further state-of-the-art work. Geng *et al.* [6] explore VNF chain migration in Low Earth orbit (LEO) satellite networks. They aim to maximize the number of VNF chain deployments while minimizing the migration cost. However, they do not consider the migration costs of the performance of the overall system. In other works, Xia *et al.* [17], Cho *et al.* [18], and Nguyen *et al.* [19] propose dynamic migration for minimized costs, latency, and optimized resource utilization. Similarly, in [7] the authors propose a VNF migration mechanism for effective resource allocation considering fluctuations in the network traffic, leading to an improved overall efficiency of the system. Yi *et al.* [8] propose an approach to migrate VNF chains from congested processing nodes to those with more resources available. Optimized for the usage in cloud data centers, the framework in [20] improves the network traffic leveraging VNF migration. Afrasiabi *et al.* [21] studies the joint VNF decomposition and migration to reduce the placement costs. The problem of migrating VNF clusters cost-efficiently while considering inter-VNF latency requirements is addressed in [22]. Tanuboddi *et al.* [23] investigated the usage of ML approaches for optimized VNF migration. However, the previous work on VNFs does not consider the flexible and specific demands of user applications, such as in the medical field.

To address this issue and to also take applications and their needs into account, in a first step, Kröger *et al.* [9] introduce the MAF concept, extending the concept of VNFs by additionally considering application-specific requirements such as service levels or non-terminability for the healthcare scenario. In a follow up work [10], the authors leverage this concept to optimally place medical applications in future 6G networks, considering priorities and adaptive resource allocation based on criticality and different performance requirements. In particular, the service level of an MAF is dynamically adjusted in a scenario of limited network and computing resources to improve the overall system performance.

The goal of this paper is to use the MAF concept and to develop an optimal MAF placement approach utilizing MAF migration. Previous work has already extensively studied various VNF migration problems. However, in this paper we also consider application-specific characteristics such as service levels, where the requirements of an MAF can be dynamically adapted. Using this approach combined with migration allows the placement of more MAF chains with higher performance, especially in resource-congested scenarios. For this, we formulate an ILP, which focuses on accepting as many MAF chains as possible with the best possible performance, considering the trade-off costs for MAF migration.

III. System Model

In [9], the authors proposed a 6G architecture for emerging medical applications using MAFs. This architecture consists of programmable switches, Access Points (AP) to abstract the access technology (e.g., WiFi, 6G, etc.), and PUs to execute MAFs. In the medical context, PUs can be processing capabilities such as PCs in a small rack within an operation area or more sophisticated PUs in a datacenter located further away. MAFs implement parts of a medical application and can be dynamically distributed on PUs and, potentially, programmable switches. This separation into different MAFs enables to define network requirements, such as latency, throughput, etc., for each single MAF. Consequently, it allows to optimize the placement of MAFs with their individual requirements in the network. Fig. 1 provides an overview of this concept. We consider the proposed architecture in combination with the MAF concept, as proposed in [9], for our system model, which is described next.

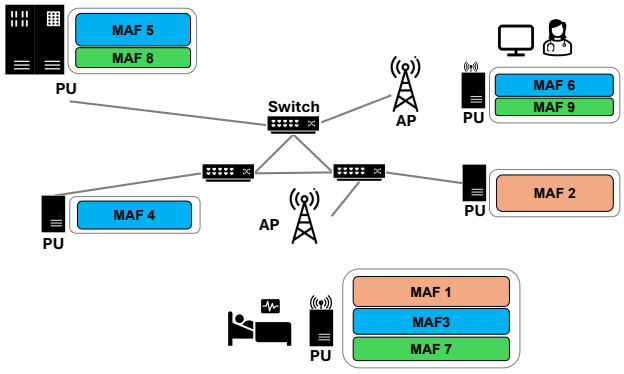

Fig. 1: Illustration of the used architecture.

In our model, the network consists of N processing nodes, i.e., PUs and switches, which are connected through links L. Physical nodes and links form the network topology $\mathcal{G}(N, L)$, which is considered to be a fully-meshed graph for simplicity. Note that more real-world networks can also be abstracted as fully-meshed graph. For that, virtual links are added where physical ones are missing. These virtual links consist of several physical connections. Hereby, the selection of such physical links is left to the operator to optimize. The interconnecting switches are potentially programmable using domain-specific programming languages, such as P4 [24]. Thus, they could execute certain parts of the applications. However, as programmable switches have special limitations and possibilities for processing applications, in a first step, we only consider PUs for the execution of MAFs. Each PU $m \in \mathcal{M}$ has CPU processing capability η_m^c, memory capacity η_m^m, i.e., the memory available for MAF execution, and storage capacity η_m^s. Finally, link $(m, n) \in \mathcal{L}$ connecting two PUs m and n is characterized by bandwidth capability $B_{m,n}$ and propagation delay $D_{m,n}$. The total set of MAFs running on PU m is denoted as $\mathcal{I}_m$.

Medical applications are abstracted into *MAF chains*. Each MAF chain $c \in \mathcal{C}$ consists of one or several MAFs and one or more inter-connecting chain links. $\mathcal{S}_c$ denotes the ordered subset of MAFs for chain c. Each chain has a maximum end-to-end delay requirement L_c. An exemplary practical use case of this model is the *Semi-autonomous Telerobotic Examination Suite* [4], where the MAF chains abstract different features such as robot control or video conferencing. Each chain, in turn, consists of one or more MAFs representing different consecutive tasks such as the robot controller and robot path planner. Additionally, a chain can be *non-terminable*, i.e., the chain must be placed. This is denoted by t_c, which equals 1 if the chain is non-terminable. Note that while non-terminability describes that an MAFC must be placed, its performance in terms of MAF service levels as introduced later can be low. This characteristic is used for tasks which are absolutely necessary but do not require highest performance such as documentation or logistic tasks.

In our model, each MAF $i \in \mathcal{I}$ shows different characteristics. First, the binary variable $v_{i,m}$ denotes whether the MAF can be placed on PU m or not. Further, it can possess different levels of service, i.e., $l \in L$ with $L = \{l_1, l_2, \ldots, l_n\}$ and l_1 being the lowest possible service level. Each service level consists of the following properties:

- Required PU resources in terms of CPU processing capacity $\pi_{i,l}^c$, memory capacity $\pi_{i,l}^m$, and storage $\pi_{i,l}^s$.
- Processing delay $d_{i,l}^p$ of MAF i.
- Throughput $f_{i,j,l}$ as the output of MAF i.

The selected service level impacts other characteristics:

- The higher the service level is, the more PU resources the MAF uses in terms of CPU processing capacity $\pi_{i,l}^c$, and memory capacity $\pi_{i,l}^m$, and vice versa.
- The processing delay, $d_{i,l}^p$, of an MAF is decreased for higher service levels and vice versa.
- The data throughput, $f_{i,j,l}$, between two consecutive MAFs i and j is determined by service level l of MAF i. Note that in a first step we assume that all MAFs are capable of processing the data with any service level.

Note that we assume that the application designer ensures that the (medical) application can still perform without any issues at the lowest service level. For availability considerations, all MAFs i of the same chain c cannot be placed on the same PU m. Note that we do not consider the analytical calculation of the availability in a first step, because a proper investigation with possible backup paths (on demand) requires further research and is deferred to future work.

We consider two time slots, T and $T+1$. Slot T represents the state of the system, i.e., some MAF chains may or may not have been placed already *before* the placement of new MAF chains is triggered. The placement of MAF i at time T on PU m with service level l is denoted with the binary variable $x_{i,m,l}$. Consequently, in slot $T+1$ the system is in the condition *after* new demands are placed, i.e., potential MAF migration, related re-routing, etc., has taken place. Migration can occur during the optimized placement due to

TABLE I: Notation

Variable	Explanation
η_m^c	CPU capability of PU m
η_m^m	Memory capability of PU m
η_m^s	Storage capability of PU m
$B_{m,n}$	Bandwidth of link (m,n) connecting PUs m and n
$D_{m,n}$	Delay of link (m,n) connecting PUs m and n
L_c	Maximal allowed latency for chain c
t_c	Terminability of chain c
$v_{i,m}$	Placeability of MAF i on PU m
$\pi_{i,l}^c$	CPU requirement of MAF i at level l
$\pi_{i,l}^m$	Memory requirement of MAF i at level l
$\pi_{i,l}^s$	Storage requirement of MAF i at level l
$d_{i,l}^p$	Processing time of MAF i at level l
$f_{i,j,l}$	Throughput of MAF i to MAF j at service level l
$x_{i,m,l}$	Placement of MAF i on PU m with level l at time T
$\epsilon_{i,m,n}$	Weight for the response traffic of MAF i placed on PU m to PU n (Constraint (17))
$y_{i,m,l}$	Placement of MAF i on PU m with level l at time $T+1$
z_c	Placement of chain c at time $T+1$
$m_{i,m,n}$	Migration of MAF i from PU m to PU n at time $T+1$
$w_{i,m,n}$	Link usage of MAF i on PU m to PU n at time $T+1$

more admitted chains, better MAF performance or lower costs. The transition time between T and $T+1$ is denoted as t and describes the time needed to perform all necessary steps for placement and migration, including all related tasks as starting the migrated MAF on the next PU, setting up the rules for routing, etc. In a first step, we consider time t to be feasible for all scenarios. Furthermore, to simplify the migration modeling, we assume all MAFs to be stateless, i.e., they do not store data which need to be transferred when being migrated.

Table I summarizes the notation used in this paper.

IV. Problem Formulation

In this section, an ILP is formulated based on the prior introduced system model. First, the decision variables and the approaches to calculate the processing cost as well as the cost of migrations are introduced. Then, the constraints are described in detail, followed by the objective function and the overall ILP formulation.

A. Decision Variables

The following binary decision variables are used in the ILP:

$y_{i,m,l} \in \{0,1\}$: Indicates whether MAF i is deployed on PU m at time $T+1$ with service level l, where 1 means deployment and 0 no deployment.

$z_c \in \{0,1\}$: Indicates whether chain c is fully activated at time $T+1$, i.e., if all of its related MAFs are placed, with 1 meaning active and 0 otherwise.

$m_{i,m,n} \in \{0,1\}$: Indicates whether MAF i has migrated from PU m to PU n at time $T+1$, with 1 indicating migration and 0 otherwise.

$w_{i,m,n} \in \{0,1\}$: Indicates whether the virtual link between PU m with placed MAF $i \in \mathcal{S}_c$ and PU n with the placed subsequent MAF $j \in \mathcal{S}_c$ of chain c is used at time $T+1$, where 1 means deployment and 0 not.

B. Placement and Migration Costs

Deploying MAFs on PUs occurs with a certain placement cost in terms of power consumption. This cost can be obtained by summing up the resource requirements of all MAFs deployed on a single PU m with service level l multiplied by their unit costs. Overall, the placement cost $\psi_{i,m}^p$ of processing MAF i on PU m can be obtained by

$$\begin{aligned}\psi_{i,m}^p = \psi_m^c \sum_l \pi_{i,l}^c \cdot y_{i,m,l} + \psi_m^m \sum_l \pi_{i,l}^m \cdot y_{i,m,l} \\ +\psi_m^s \sum_l \pi_{i,l}^s \cdot y_{i,m,l}, \quad \forall l \in L,\ \forall m \in \mathcal{M},\ \forall i \in \mathcal{I}.\end{aligned} \tag{1}$$

Hereby, ψ_m^c, ψ_m^m, and ψ_m^s represent the unit cost for CPU, memory, and storage resources of PU m, respectively. Note that we consider PUs always to be ON for simplification. Therefore, we only consider the dynamic costs, i.e., caused by the placement of MAFs, and not by the static power consumption costs.

The migration of an MAF i from PU m to PU n leads to costs for the overall placement. These costs can include the size of the migrated MAF, impact of the start-up at PU n on other MAFs, etc. Thus, we assume a general cost ψ_i^m to cover all costs related to the migration of MAF i. Consequently, the migration costs of migrating MAF i from PU m to PU n are calculated from

$$\psi_{i,m,n}^m = m_{i,n,m} \cdot \psi_i^m, \quad \forall i \in \mathcal{I},\ \forall n,m \in \mathcal{M}, \tag{2}$$

where $m_{i,n,m}$ denotes whether migration takes place or not.

C. Constraints

In the following, all constraints of our ILP, given the system model as described in Section III, are introduced. The first constraint ensures that each MAF i of chain c can only be placed on one PU m with only one service level l if chain c is also deployed:

$$\sum_m \sum_l y_{i,m,l} \ \leq \ z_c,\ l \in L, \forall m \in \mathcal{M}, \forall i \in \mathcal{S}_c, \forall c \in \mathcal{C}. \tag{3}$$

Further, an MAF i can only be placed on PU m if it is allowed to be placed there. The constraint is formulated as

$$\sum_l y_{i,m,l} \ \leq \ v_{i,m}, \quad \forall l \in L,\ \forall m \in \mathcal{M},\ \forall i \in \mathcal{I}. \tag{4}$$

Note that this constraint can also be used to determine a source and a destination PU. Next, MAFs of chain c cannot be placed on the same PUs, which is expressed as

$$\sum_i \sum_l y_{i,m,l} \ \leq 1,\ \forall l \in L, i \in \mathcal{S}_c, \forall m \in \mathcal{M}, \forall c \in \mathcal{C}. \tag{5}$$

The following constraints address correct migration. Additionally, they ensure that an MAF placed at time T on PU m must be placed on PU n at time $T+1$ again if not migrated or the chain is rejected. Furthermore, each MAF is allowed to migrate at most once, which is captured by

$$\sum_m \sum_n m_{i,m,n} \ \leq \ 1, \quad \forall m,n \in \mathcal{M},\ \forall i \in \mathcal{I}. \tag{6}$$

Additionally, an MAF i cannot be migrated from PU m to the same PU, which is expressed by

$$\sum_{m} m_{i,m,m} = 0, \quad \forall m \in \mathcal{M},\ \forall \in \mathcal{I}. \tag{7}$$

However, an MAF can only migrate from PU m to PU n at time $T+1$ if it has been placed on PU m at time T:

$$m_{i,m,n} \ \leq \ \sum_{l} x_{i,m,l}, \quad \forall l \in L,\ \forall i \in \mathcal{I},\ \forall m,n \in \mathcal{M}. \tag{8}$$

If MAF i is not deployed on PU n after migration, then $m_{i,m,n}$ must be 0:

$$m_{i,m,n} \ \leq \ \sum_{l} y_{i,n,l}, \quad \forall i \in \mathcal{I},\ \forall m,n \in \mathcal{M}. \tag{9}$$

Finally, MAF i is migrated successfully from PU m to PU n if it is deployed on PU n at service level l after migration, i.e., at time $T+1$, and was deployed on PU m at service level l before migration, i.e., at time T. This is described by

$$\begin{aligned} \sum_{l} y_{i,n,l} + \sum_{l} x_{i,m,l} - 1 \ \leq \ m_{i,m,n}, \\ \forall l \in L,\ \forall i \in \mathcal{I},\ \forall m,n \in \mathcal{M}. \end{aligned} \tag{10}$$

Note that this constraint also captures the condition that MAF i placed on PU m at time T can only be placed on PU n at time $T+1$ via migration. Next, the resources of each PU m in terms of CPU capacity, memory capacity, and storage capacity are limited and cannot be exceeded. This is ensured by the following constraints:

$$\sum_{i}\sum_{l} \pi^{c}_{i,l} \cdot y_{i,m,l} \leq \eta^{c}_{m}, \quad \forall l \in L,\ \forall i \in \mathcal{I},\ \forall m \in \mathcal{M}, \tag{11}$$

$$\sum_{i}\sum_{l} \pi^{m}_{i,l} \cdot y_{i,m,l} \leq \eta^{m}_{m}, \quad \forall l \in L,\ \forall i \in \mathcal{I},\ \forall m \in \mathcal{M}, \tag{12}$$

$$\sum_{i}\sum_{l} \pi^{s}_{i,l} \cdot y_{i,m,l} \leq \eta^{s}_{m}, \quad \forall l \in L,\ \forall i \in \mathcal{I},\ \forall m \in \mathcal{M}. \tag{13}$$

MAF chain c is only successfully deployed if all of its corresponding MAFs $i \in \mathcal{S}_c$ are also successfully deployed. With $|\mathcal{S}_c|$ denoting the number of MAFs related to chain c, the constraint is formulated as

$$z_c|S_c| = \sum_{i}\sum_{m}\sum_{l} y_{i,m,l},\ \forall i \in \mathcal{S}_c, \forall m \in \mathcal{M}, \forall c \in \mathcal{C}. \tag{14}$$

The next constraint ensures that the end-to-end delay requirement L_c of each chain c is met. Hereby, L_c is composed of two parts: *(i)* the processing delay $d^{p}_{i,l}$ of each MAF, and *(ii)* the transmission delay $D_{m,n}$ between adjacent MAFs i and j placed on PUs m and n in the chain. Given i_n as the final node of S_c, the constraint is expressed as

$$\begin{aligned} \sum_{j \in S_c}\sum_{m}\sum_{l} d^{p}_{i,l} y_{j,m,l} + \sum_{i \in S_c \setminus \{i_n\}}\sum_{m}\sum_{n} w_{i,m,n} D_{m,n} \\ \leq L_c \cdot z_c, \quad \forall l \in L, \forall n,m \in \mathcal{M}, \forall c \in \mathcal{C}. \end{aligned} \tag{15}$$

Further, all the non-terminable MAF chains must be placed:

$$t_c \ \leq \ z_c,\ \forall c \in \mathcal{C}. \tag{16}$$

The total traffic on each link cannot exceed its available bandwidth capacity. This traffic includes the data traffic between MAFs and the migration traffic that occurs when migrating MAFs between PUs. However, for simplicity, we assume that the MAF can be transmitted at time t between T and $T+1$. This way, all bandwidth resources are available for data traffic at time $T+1$. With i_n being the last MAF in S_c, the following constraint defines the traffic on each link (m,n), connecting PU m with MAF i and PU n with MAF j at time $T+1$ as

$$\begin{aligned} \sum_{l} \epsilon_{i,m,n} \cdot \ w_{i,m,n} \cdot f_{i,j,l} \ \leq \ B_{m,n}, \\ \forall m,n \in \mathcal{M},\ \forall i \in \mathcal{S}_c \setminus \{n_i\},\ \forall j \in \mathcal{S}_c \setminus \{n_1\}. \end{aligned} \tag{17}$$

Hereby, $\epsilon_{i,m,n}$ is a weight to adjust the traffic on link (m,n). This simplification enables to also consider response traffic back from MAF j and MAF i, which might differ from the traffic flowing from MAF i to MAF j. The next constraints concern the proper routing, i.e., the activation of the interconnecting links. In the following, i_1 denotes the first MAF and i_n the last MAF in S_c. The following constraints ensure that only if MAF i is placed on PU m and consecutive MAF j on PU n, the interconnecting link $\lambda \in L$ must be activated. Firstly, the link (m,n) can only be activated if MAF i is placed:

$$\begin{aligned} w_{i,m,n} \ \leq \ \sum_{l} y_{i,m,l}, \\ \forall l \in L,\ \forall m,n \in \mathcal{M},\ \forall i \in \mathcal{S}_c \setminus \{i_m\},\ \forall c \in \mathcal{C}. \end{aligned} \tag{18}$$

Secondly, link (m,n) also needs a placed MAF j, which is expressed as

$$\begin{aligned} w_{i,m,n} \ \leq \ \sum_{l} y_{j,n,l}, \quad \forall l \in L, \\ \forall m,n \in \mathcal{M}, \forall i \in \mathcal{S}_c \setminus \{i_m\},\ \forall j \in \mathcal{S}_c \setminus \{n_i\},\ \forall c \in \mathcal{C}. \end{aligned} \tag{19}$$

Finally, if MAF i and MAF j are placed, the interconnecting link must be activated:

$$\begin{aligned} \sum_{l} y_{i,m,l} + \sum_{l} y_{j,n,l} - 1 \ \leq \ w_{i,m,n}, \quad i \neq j, \\ \forall i \in S_c \setminus \{i_n\},\ \forall j \in S_c \setminus \{i_1\},\ \forall m,n \in \mathcal{M},\ \forall c \in \mathcal{C}. \end{aligned} \tag{20}$$

D. Objective Function

The optimization goal is to accept as many MAF chains as possible with the best possible performance, reflected by the chosen service level. Thus, the first objective is formally described as

$$\max \ \mathcal{O}_1 = \max \ \sum_{c} z_c, \quad \forall c \in \mathcal{C}. \tag{21}$$

The performance of MAFs is characterized by their chosen service level and thus, the second objective is expressed as

$$\max \mathcal{O}_2 = \max \sum_{l} l \cdot y_{i,m,l},\ \forall l \in L, \forall m \in \mathcal{M}, \forall i \in \mathcal{I}. \tag{22}$$

On the other hand, the placement of MAF chains with high performance leads to a higher overall cost of the system. These costs should be as low as possible in order to make the approach more profitable and sustainable. Thus, using (1), the third objective is calculated as

$$\min \ \mathcal{O}_3 = \min \ \sum_{m} \psi^{p}_{i,m}, \quad \forall m \in \mathcal{M}. \tag{23}$$

Lastly, migrating already placed MAFs to other PUs at time slot $T+1$ can, on the one hand, increase the number of admitted MAF chains and their performance, as well as to lead to overall lower system costs. On the other hand,

MAF migration also comes with certain costs as a trade-off. Therefore, the aim is to reduce migration costs, which is denoted by

$$\min\ \mathcal{O}_4 = \min\ \psi^m_{i,n,m}, \quad \forall m,n \in \mathcal{M}, \forall i \in \mathcal{I}. \tag{24}$$

Finally, using (21)-(24) and the weights $\lambda_1, \lambda_2, \lambda_3$, and λ_4, the optimization problem can be formulated together as

$$\begin{aligned} \max_{y_{i,m,l},\, z_c,\, m_{i,m,n},\, w_{i,m,n}} \quad & \lambda_1 \cdot \mathcal{O}_1 + \lambda_2 \cdot \mathcal{O}_2 \\ & - \lambda_3 \cdot \mathcal{O}_3 - \lambda_4 \cdot \mathcal{O}_4 \\ \text{s. t.} \quad & (3) - (20). \end{aligned} \tag{25}$$

Increasing λ_1 leads to higher priority of accepted chains in comparison to their performance, while increasing λ_2 leads to the opposite. The weights λ_3 and λ_4 are weighting coefficients for prioritizing the overall system costs or the migration costs. Increasing their values may lead to worse performance or less accepted chains. Note that a lower λ_4 potentially leads to a higher occurrence of migrations due to the lower significance of migration costs. The optimization problem at hand can be seen as a type of the General Assignment Problem and is therefore considered NP-hard [25].

V. Proposed Heuristic

In order to obtain near-optimal results in a more reasonable amount of time and for better scalability, we propose the Performance-Aware Chain Admission and Migration (PACAM) heuristic. Note that we decided to develop a heuristic rather than AI/ML-based solution approaches for their predictable behavior. AI/ML-based methods are deferred to future work. PACAM consists of four steps, as described in Algorithm 1: *(i)* Placing non-terminable chains, *(ii)* trying to place all terminable chains, *(iii)* increasing the service levels of all placed MAFs if possible, and *(iv)* migrating MAFs if higher service levels are possible on other PUs. In particular, in the first step all previously placed non-terminable MAF chains are placed on the previous PUs with the lowest service level. Then, all remaining non-terminable MAF chains are placed using an adapted Viterbi algorithm [26]. Hereby, the actual placement costs of MAFs placed on PU m are used for the path decision. Furthermore, in this step, it is ensured that all relevant constraints such as chain latency, PU capacity, link resources, only one MAF per chain on one PU, etc., are satisfied. In Step 2, Step 1 is repeated for the terminable chains with the only difference that the placement of a chain might not be possible and therefore the chain might not be placed due to the lack of resources. Then, in Step 3, the service levels of all placed MAFs are increased considering the PU capacity. In the final step (Step 4), possible migration is checked for all placed MAFs. In particular, for all MAFs on PU n the possibility of being placed on different PU m with a higher service level is evaluated considering all constraints. This step ends when no changes can be done anymore.

The time complexity of Step 1 and Step 2 are dominated by the Viterbi algorithms used for finding a suitable placement for non-terminable and terminable chains in Step 1, Line 14, and Step 2, Line 23. and given as $\mathcal{O}(C \cdot S \cdot N^2)$, where C denotes the number of MAF chains, S is the maximal number of MAFs per chain, and N stands for the number of PUs. In Step 3, for all MAFs on all nodes the service level is increased, leading to a time complexity of $\mathcal{O}(NI)$, with I being the total number of MAFs. Due to the three loops, the time complexity of Step 4 is given as $\mathcal{O}(N^2k)$, with k as the maximum number of MAFs placed on a PU. If $k << N$, the overall time complexity of *PACAM* is dominated by the Viterbi algorithms and therefore be denoted with $\mathcal{O}(C \cdot S \cdot N^2)$. However, if k approaches N, the time complexity becomes $\mathcal{O}(N^3)$.

Algorithm 1 *PACAM*

1: **Initialize:**
2: $C^t_{nt} \leftarrow$ List of placed non-terminable chains at time T
3: $C^p_{nt} \leftarrow$ List of placed non-terminable chains at time $T+1$
4: $C_{nt} \leftarrow$ List of all non-terminable chains
5: $C^t_t \leftarrow$ List of placed terminable chains at time T
6: $C_t \leftarrow$ List of all terminable chains
7: $C^p_t \leftarrow$ List of placed terminable chains at time $T+1$
8: **Step 1: Non-terminable Chains**
9: **for** each chain c in C^t_{nt} **do**
10: $C^p_{nt} \leftarrow$ *place_prev_non_term_chains()*
11: **end for**
12: **for** each chain c in C_{nt} **do**
13: **if** chain c **not in** C^p_{nt} **then**
14: $C^p_{nt} \leftarrow$ *place_non_term_chains()*
15: **end if**
16: **end for**
17: **Step 2: Terminable Chains**
18: **for** each chain c in C^t_{nt} **do**
19: $C^p_t \leftarrow$ *place_prev_term_chains()*
20: **end for**
21: **for** each chain c in C_t **do**
22: **if** chain c not in C^p_t **then**
23: $C^p_t \leftarrow$ *place_term_chains()*
24: **end if**
25: **end for**
26: **Step 3: Increase service level**
27: *increase_maf_service_levels(),* $\forall i \in S_c, \forall c \in C^p_{nt}, C^p_t$
28: **Step 4: Test migration for better performance**
29: *migrate_maf_if_possible_and_better()*
30: **return** Placement solution

VI. Performance Evaluation

In this section, we first describe the evaluation setup in detail. Then, the performance of our approach is evaluated through simulations and compared to existing approaches.

A. Setup Description

We evaluate the *Optimal* approach as described in Section IV and the *PACAM* heuristic, as introduced in Section V, in regards to their performance behavior. For that, the *Optimal* approach is implemented with Gurobi [27] and *PACAM* in Python. The corresponding results are then compared with the output of four comparison approaches:

TABLE II: MAF parameters

Parameter	Value
Initial CPU	$\{100, 400, 800\}$
Initial Memory	$\{6, 15\}$
Initial Storage	$\{10, 20\}$
Initial Data Rate	$\{20, 50, 100, 200\}$
Initial Processing Time	$\{0.01, 0.02\}$
Service Level Scales	$\{0.6, 0.853, 1.0, 1.25, 1.6\}$
Migration	$\{20, 100\}$

TABLE III: PU Parameters

Parameter	Value
Max CPU Capacity	$\{100, 200, 400, 800, 5000, 8000\}$
Max MEM Capacity	$\{64, 128, 256, 2560, 5120\}$
Initial STO Usage	$\{256, 1000, 2000, 4000, 40000\}$
CPU Unit Cost	$\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$
MEM Unit Cost	$\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$
STO Unit Cost	$\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$

- **Greedy:** This algorithm is based on *PACAM*, but does not consider previously placed chains in Step 1 and Step 2. In other words, it places MAF and corresponding service levels greedily on available PUs.
- **LSMO5:** This approach is based on the optimal approach from [6], optimizing for admission ratio, used bandwidth, and migration costs. We adapted their work by relaxing all constraints regarding path visibility and added our constraints on chain admission only when all of its MAFs are placed, MAFs of one MAF chain cannot be placed on the same PU, and MAFs can only be placed on allowed nodes. Additionally, we used our definition for migration costs instead of theirs. For the MAFs, we chose the highest possible service level, i.e., $l = 5$, because [6] does not consider changing requirements.
- **LSMO3:** The solution of this approach is obtained with the same method as *LSMO5*. However, it uses service level 3 for all MAFs.
- **LSMO1:** Similarly, the solution of this approach is obtained with the same method as *LSMO5*. However, it uses service level 1 for all MAFs.

We consider a fully meshed network topology consisting of 10 PUs. For the evaluation, we mainly focus on reducing one resource, the CPU capacity, in such a way that not all MAF chains can be deployed anymore with full performance, i.e., with the highest service level. Note that our approach also covers the case when other resources, such as the memory, storage capacity, or the link throughput are scarce. However, since the link bandwidth is obtained from the set $\{250, 500, 10000\}$, the lack of networking resources can potentially impact the placement as well. The link propagation delay is negligible. The goal is to place 20 MAF chains, each consisting of $2-5$ MAFs (70 MAFs in total), within such a network at time $T+1$. Hereby, 10 chains are already placed and executed at time T using our *Optimal* approach, leaving 10 MAF chains to be placed at time $T+1$. Note that MAFs have a randomly chosen probability of 0.95 to be allowed for placement on each PU m. Such a placement restriction can occur for various reasons

TABLE IV: ILP parameters

Parameter	Value
λ_1 (Weight for MAFC admission ratio)	200000
λ_2 (Weight for MAF performance)	10000
λ_3 (Weight for MAF placement costs)	1
λ_4 (Weight for MAF migration costs)	1
$\epsilon_{i,m,n}$ (Weight for link traffic (Constraint (17))	2

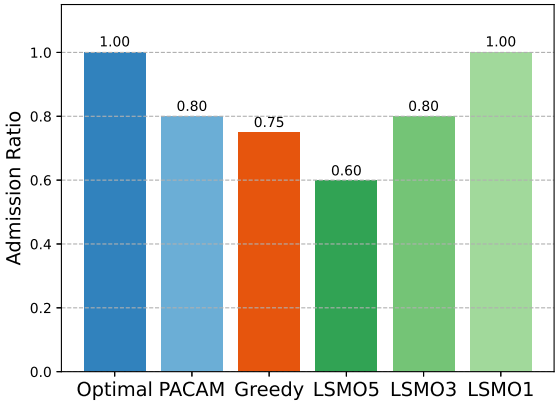

Fig. 2: Chain Acceptance Ratio.

such as required special hardware (hardware accelerators), a certain level of PU security, etc. An example in the medical area is the *Semi-autonomous Telerobotic Examination Suite* in [4], where some MAFs such as the robot controllers need to be placed closer to the robot.

Each MAF can have one of five possible service levels with 1 being the lowest and 5 the highest. For the simulation, we choose initial values from an interval and then scale it according to the service level. All MAF characteristics can be found in Table II. For the placement, we heavily prioritize the admission ratio and MAF performance and therefore use higher numbers for their weighting factors, i.e., λ_1 and λ_2, for the *Optimal*, as shown in Table IV, and the same for all three *LSMO* approaches. Further details on the used parameters for the PUs are shown in Table III. Note that those values are arbitrarily chosen in order to cover many possible applications with a variety of requirements. The reason for this lies in the lack of knowledge of the requirements of real future application. However, the real values as well as their corresponding units are not as interesting as their relation to the overall available resources. This way, we show that our approach still holds even if the concrete values are changed.

B. Results

We evaluate the chain admission ratio, the execution time, the performance of the MAFs, the number and costs of migrations and finally, the placement costs for all our introduced as well as the comparison approaches described before.

1) Admission Ratio: Fig. 2 shows the chain admission ratio. In the evaluated resource constrained scenario, not all chains might be placed by some approaches. The *Optimal* approach,

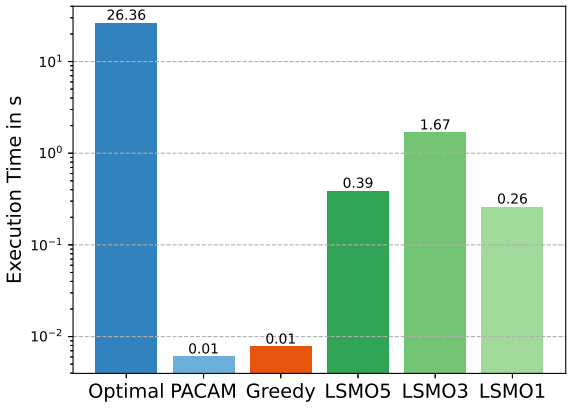

Fig. 3: Execution Time.

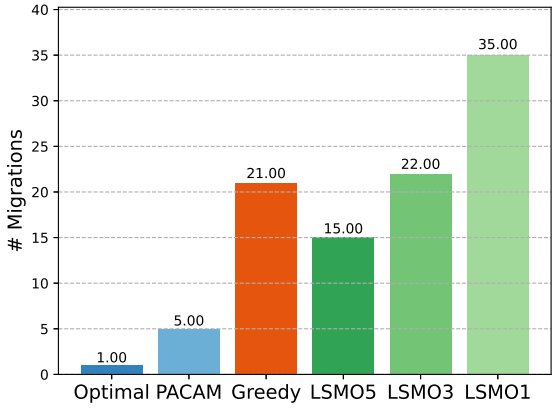

Fig. 5: Number of migrations.

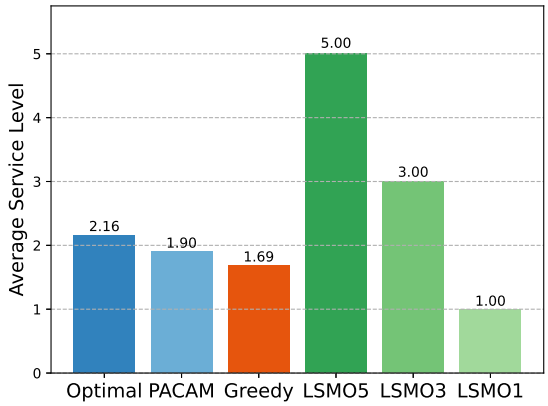

Fig. 4: System Performance in terms of average service level.

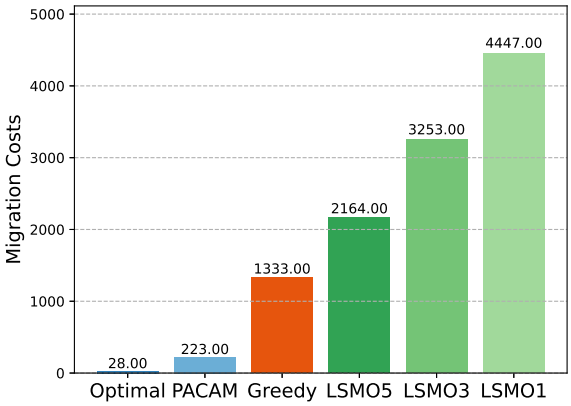

Fig. 6: Costs of all migrations.

however, is able to place all chains with the trade-off of less performance, i.e., lower service levels. *PACAM* is not able to place all chains, but places more than *Greedy* and *LSMO5*. For a decreasing service level for the *LSMO* approaches the admission ratio increases. The reason for this lies in the sacrificed performance in order to allow more chains. Note that *PACAM* and *LSMO3* have the same admission ratio. Thus, the individual performance is of interest and is discussed later.

2) Execution Time: Our *Optimal* approach, as shown in Fig. 3, shows the expected highest execution time, which is significantly higher than of the other approaches. Hereby, *LSMO5* and *LSMO1* execute with below $0.5\,\mathrm{s}$ faster than *LSMO3*, while *PACAM* and *Greedy* stay below $10\,\mathrm{ms}$. Note that the execution time of the *Optimal* approach varies and increases the lower the resources are, and therefore more possible placements have to be evaluated.

3) System Performance: The system performance for each approach is calculated as the sum of the used service level of each MAF divided by the maximum possible service levels and is shown in Fig. 4. It can be observed that the performance of *Optimal* and *PACAM* are rather low and close together while still outperforming *Greedy*. The reason for this low performance lies in the increased chain admission ratio, which has been heavily favored with a high weight. As the *LSMO* approaches do not consider service levels, their performance corresponds with the configured service level. Note that the performance of *LSMO3* is higher than that of *PACAM*, even though the admission ratio is the same. However, the disadvantage of *LSMO3* is that the performance cannot be increased further while the service level of *PACAM* can increase to 5.

4) Number of Migrations: Fig. 5 shows the number of migrations for all approaches. While the number is very low for *Optimal* and *PACAM*, it increases significantly for *Greedy* and all *LSMO* approaches. Note that it increases more for the *LSMO* approaches the lower the service level gets. The reason for this lies most likely in higher admission ratios, allowing more chains to be placed with lower service level and possible migrations. *Greedy* does not consider prior placement which leads to higher numbers of migrations compared to our algorithm.

5) Migration Cost: The number of migrations shown in Fig. 5 combined with the costs for each migration lead to the results for the migration costs shown in Fig. 6. As can be observed, the migration costs for the *LSMO* approaches are

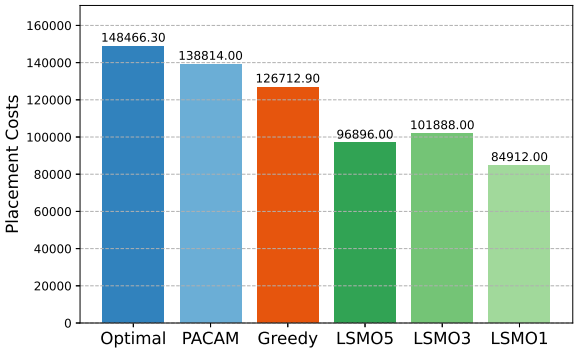

Fig. 7: Costs of the placement.

significantly higher than for *Optimal* and *PACAM*.

6) Placement Costs: Finally, the overall placement costs for all approaches are shown in Fig. 7. The costs for *Optimal*, *PACAM*, and *Greedy* are higher than for the *LSMO* approaches due to a higher number of placed chains with higher performance. This can be observed best by comparing *Optimal* and *LSMO1*. While both have the same admission ratio, the performance of *LSMO1* is much worse and therefore uses less resources, which leads to less placement costs. Note that placement costs of *PACAM* are significantly higher than for *LSMO3*, which has the same admission ratio and higher performance. The reason for this most likely lies in the way both approaches work. *LSMO3* can find the best possible solution within certain bounds. *PACAM* needs to trade off between certain objectives which, in this case, most likely leads to the placement of MAFs on higher priced PUs. This assumption is supported by the low number of migrations for *PACAM* and the high number of migrations for *LSMO3*.

7) Summary: The primary goal of our proposed approaches, *Optimal* and *PACAM*, is to allow as many chains with the highest service level as possible. The results show that both achieve the targeted goals very well. *Optimal* allows the placement of all chains, while *PACAM* accepts most of the chains. Even though the performance of *PACAM* is worse than that of *LSMO3* with the same chain admission ratio, the number of migrations and therefore of introduced migrations costs is significantly lower. Additionally, *PACAM* can increase the MAF performance with more available resources while the performance is fixed in *LSMO3* and obtains results faster. Even though our approaches exceed in the number of admitted chains, the performance level and the low number of migrations, the trade-off lies in the higher costs of overall placements, which are significantly higher compared to the benchmarks. Note that the *LSMO* approaches do not consider non-terminable chains, which have to be placed. Even though in our experiments the non-terminable chains are always placed for all approaches even if not all chains are admitted, this is not guaranteed by the *LSMO* approaches.

VII. Conclusion

In this paper, we propose a new approach to optimally place Modular Application Functions (MAFs) within a network for medical scenarios considering migration costs. The primary objective is to place as many MAF chains as possible with the highest possible performance inside the network. At the same time, we aim to reduce the overall placement and migration costs. We formulate an ILP that captures MAF migration with different service levels for each MAF as well as the non-terminability of MAF chains. To solve the problem at hand within a reasonable timeframe, we introduce a heuristic. Evaluations show that our approach outperforms benchmarks in terms of admission ratio by up to 40% while still showing acceptable performance and a low number of migrations. In the future, we plan to extend our work by adding the migration to a more holistic MAF placement approach potentially, covering various aspects of placement such as priority and availability with on-demand backup paths. Hereby, we plan to investigate the usage of AI/ML-based approaches for the placement of medical MAFs. Furthermore, we plan to evaluate our approach in a real-world medical testbed.

References

[1] M. Hoffmann, G. Kunzmann, T. Dudda, R. Irmer, A. Jukan, G. Macher, A. Ahmad, F. R. Beenen, A. Bröring, F. Fellhauer, G. P. Fettweis, F. H. P. Fitzek, N. Franchi, F. Gast, B. Haberland, S. Hoppe, S. Joodaki, N. P. Kuruvatti, C. Li, M. Lopez, F. Mehmeti, T. Meyerhoff, L. Miretti, G. T. Nguyen, M. Parvini, R. Pries, R. F. Schaefer, P. Schneider, D. A. Schupke, S. Strassner, H. Stubbe, and A. M. Voicu, "A secure and resilient 6G architecture vision of the german flagship project 6G-ANNA," *IEEE Access*, vol. 11, pp. 102 643–102 660, 2023.

[2] W. Saad, M. Bennis, and M. Chen, "A vision of 6G wireless systems: Applications, trends, technologies, and open research problems," *IEEE Network*, vol. 34, no. 3, pp. 134–142, 2020.

[3] N. A. Mohammedali, T. Kanakis, and M. O. Agyeman, "Survey on the impact of AI, robotics and 6G networks on the remote surgery," in *Proc. of IEEE International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, 2024.

[4] S. Kolb, A. Madden, N. Kröger, F. Mehmeti, F. Jurosch, L. Bernhard, W. Kellerer, and D. Wilhelm, "6G in medical robotics: Development of network allocation strategies for a telerobotic examination system," *International Journal of Computer Assisted Radiology and Surgery (CARS)*, vol. 20, 2024.

[5] F. Jurosch, N. Kröger, S. Kolb, F. Mehmeti, E. Martens, S. Speidel, W. Kellerer, D. Wilhelm, and J. Fuchtmann, "6G networks for the operating room of the future," *Progress in Biomedical Engineering*, vol. 6, no. 4, 2024.

[6] G. Yuhui, W. Niwei, C. Xi, X. Xiaofan, Z. Changsheng, Y. Junyi, X. Zhenyu, and C. Xianbin, "Service function chain migration in LEO satellite networks," *China Communications*, vol. 21, no. 3, 2024.

[7] C. Gonzalez and B. Tang, "AggVNF: Aggregate VNF allocation and migration in dynamic cloud data centers," in *Proc. of IEEE International Conference on Network Softwarization (NetSoft)*, 2024.

[8] B. Yi, X. Wang, M. Huang, and K. Li, "Design and implementation of network-aware VNF migration mechanism," *IEEE Access*, vol. 8, 2020.

[9] N. Kröger, S. Kolb, F. Jurosch, D. Wilhelm, and W. Kellerer, "Processing modular application functions in future medical 6G radio access networks," in *Proc. of IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, 2024.

[10] N. Kröger, G. Gattulli, F. Jurosch, S. Kolb, D. Wilhelm, W. Kellerer, and F. Mehmeti, "Processing prioritization of modular medical applications in future 6G radio access networks," in *Proc. of IEEE/IFIP Network Operations and Management Symposium (NOMS)*, 2025.

[11] R. Cziva, C. Anagnostopoulos, and D. P. Pezaros, "Dynamic, latency-optimal vnf placement at the network edge," in *Proc. of IEEE Conference on Computer Communications*, 2018.

[12] S. Yang, F. Li, S. Trajanovski, X. Chen, Y. Wang, and X. Fu, "Delay-aware virtual network function placement and routing in edge clouds," *IEEE Transactions on Mobile Computing*, vol. 20, no. 2, 2021.
[13] L. Qu, C. Assi, and K. Shaban, "Delay-aware scheduling and resource optimization with network function virtualization," *IEEE Transactions on Communications*, vol. 64, no. 9, 2016.
[14] S. Agarwal, F. Malandrino, C.-F. Chiasserini, and S. De, "Joint VNF placement and CPU allocation in 5G," in *Proc. of IEEE Conference on Computer Communications*, 2018.
[15] 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," *IEEE Transactions on Network and Service Management*, vol. 16, no. 1, 2019.
[16] S. Xu, B. Liao, B. Hu, C. Han, C. Yang, Z. Wang, and A. Xiong, "A reliability-and-energy-balanced service function chain mapping and migration method for internet of things," *IEEE Access*, vol. 8, 2020.
[17] J. Xia, Z. Cai, and M. Xu, "Optimized virtual network functions migration for NFV," in *Proc. of IEEE International Conference on Parallel and Distributed Systems (ICPADS)*, 2016.
[18] D. Cho, J. Taheri, A. Y. Zomaya, and P. Bouvry, "Real-time virtual network function (VNF) migration toward low network latency in cloud environments," in *Proc. of IEEE International Conference on Cloud Computing (CLOUD)*, 2017.
[19] N. T. Khài, A. Baumgartner, and T. Bauschert, "Optimising virtual network functions migrations: A flexible multi-step approach," in *Proc. of IEEE Conference on Network Softwarization (NetSoft)*, 2019.
[20] V. Tran, J. Sun, B. Tang, and D. Pan, "Traffic-optimal virtual network function placement and migration in dynamic cloud data centers," in *Proc. of IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2022.
[21] S. N. Afrasiabi, A. Ebrahimzadeh, A. Azhdari, C. Mouradian, W. Li, R. Szabó, and R. H. Glitho, "Joint VNF decomposition and migration for cost-efficient VNF forwarding graph embedding," in *Proc. of IEEE Global Communications Conference*, 2023.
[22] S. N. Afrasiabi, A. Ebrahimzadeh, N. Promwongsa, C. Mouradian, W. Li, Recse, R. Szabó, and R. H. Glitho, "Cost-efficient cluster migration of VNFs for service function chain embedding," *IEEE Transactions on Network and Service Management*, vol. 21, no. 1, 2024.
[23] B. R. Tanuboddi, G. Gad, Z. M. Fadlullah, and M. M. Fouda, "Optimizing VNF migration in B5G core networks: A machine learning approach," in *Proc. of International Conference on Smart Applications, Communications and Networking (SmartNets)*, 2024, pp. 1–5.
[24] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker, "P4: programming protocol-independent packet processors," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, p. 87–95, Jul. 2014.
[25] O. Kundakcioglu and S. Alizamir, *Generalized assignment problem Generalized Assignment Problem*. Springer, Boston, MA, 01 2008, pp. 1153–1162.
[26] G. Forney, "The viterbi algorithm," *Proceedings of the IEEE*, vol. 61, no. 3, 1973.
[27] Gurobi, "Gurobi optimizer," https://www.gurobi.com/solutions/gurobi-optimizer/ [Accessed: (12/06/2025)].