A Benders Decomposition Approach for Resilient Placement of Virtual Process Control Functions in Mobile Edge Clouds

Peiyue Zhao and György Dán
School of Electrical Engineering and Computer Science
KTH Royal Institute of Technology
Stockholm, Sweden. E-mail: {peiyue|gyuri}@kth.se

***Abstract*—Replacing hardware controllers with software-based Virtual Process Control Functions (VPFs) is a promising approach for improving the operational efficiency and flexibility of industrial control systems. VPFs can be executed in edge clouds in 5G mobile networks or in the wireless backhaul, which can further improve efficiency. Nonetheless, for the acceptance of virtualization in industrial control systems, a fundamental challenge is to ensure that the placement of VPFs be resilient to component failures and cyber-attacks, besides being efficient. In this paper we address this challenge by considering that VPF placement costs are incurred by reserving MEC resources, executing VPF instances, and by data communication. We formulate the VPF placement problem as an integer programming (IP) problem, considering resilience as a constraint. We propose a solution based on generalized Benders decomposition and based on linear relaxation of the resulting sub-problems, which effectively reduces the number of integer variables to the number of MEC nodes. We evaluate the proposed solution with respect to operational cost, efficiency, and scalability in a simulated metropolitan area. Our results show that the proposed solution reduces the total cost significantly compared to a greedy baseline algorithm and a local search heuristic, and can scale to moderate problem instances.**

***Index Terms*—Mobile edge computing, Resilient facility location, software controller, virtual function placement, IoT**

I. Introduction

Legacy industrial control systems (ICS) consist of real-time data collection from sensors, data processing in standalone hardware controllers, and control command execution through actuators. Softwarization of the hardware controllers is rapidly emerging in ICS as a key enabler for improving operational efficiency and flexibility, and for reducing capital expenditures [1]–[3]. With softwarization, hardware controllers are replaced by software instances, referred to as Virtual Process Control Functions (VPFs). VPFs can be placed in commodity servers, can be flexibly provisioned on demand, and are easier to upgrade than legacy hardware controllers. Examples include energy distribution, manufacturing, healthcare, and the automotive industry.

The natural choice for placing VPFs in today's industrial network architectures would be self-managed servers on the shop-floor or in a private cloud [4]. In the future, a promising alternative could be Mobile Edge Computing (MEC), which is expected to be a key enabler of 5G [5]. MEC brings distributed computing and storage resources close to end-users in mobile networks, with low latency and high bandwidth. Together with low latency 5G wireless access, industries could further reduce their operational cost and improve the flexibility of their control processes by relying on MEC for VPF placement.

Resilience is a fundamental requirement for the adoption of MEC for VPF placement. VPFs should be resilient to cyber attacks (e.g., denial-of-service attacks and advanced persistent threats) and to the potential failure of communication (wireless access and the mobile backhaul) and computing resources (virtual machines or servers in MEC nodes). Resilience can be achieved by executing redundant copies of VPFs at multiple MEC nodes, akin to 1+1 redundancy, but this approach requires significant amount of redundant communication and computing resources, and would thus lead to high operational cost. A more cost efficient approach is to rely on shared redundancy, akin to N+1 redundancy, and to restore a VPF instance on a different MEC node in case of a failure, depending on the failure scenario. Restoring a VPF instance on a different MEC node may, however, be limited due to other VPF instances running there already, and hence may require other VPF instances to be migrated [6], which makes the problem of optimal resilient VPF placement challenging.

In this paper, we address the problem of resilient VPF placement with the objective of minimizing the expected operational cost, due to making a MEC nodes available for use, due to executing VPF instances on the MEC nodes that are made available for use, and due to transmitting data from the sensors to the VPFs and from the VPFs to the actuators. The set of MEC nodes and communication links that are suitable is determined by a given set of failure scenarios, and the objective is to find the minimum cost VPF placement subject to capacity and to resilience constraints, under all possible scenarios. We formulate the resilient VPF placement problem as an integer programming (IP) problem, and we propose an efficient iterative algorithm based on generalized Benders decomposition and linear relaxation. Through linear relaxation the proposed Resilient VPF Placement (RVP) algorithm converts the large scale IP problem into a mixed integer linear programming (MILP) with as many integral variables as MEC nodes. We prove convergence of the RVP algorithm and provide a bound on the sensitivity of the solution to the algorithm's initialization. As an alternative to RVP, we also propose a low-complexity heuristic based on local search. Our numerical results show that the RVP algorithm can reduce the expected cost significantly compared to a greedy baseline algorithm and compared to the local search heuristic,

and scales significantly better than previous solutions to the problem [7].

The rest of this paper is organized as follows. We review related work in Section II and introduce the system model in Section III. In Section IV we present our solution based on Benders decomposition and provide analytical results. In Section V we present the local search heuristic. We provide numerical results in section VI, and we conclude the paper in section VII.

II. Related Work

There has been a significant interest in cloud computing for industrial use cases, both concerning potential application areas and security aspects [8], [9], but the focus of these works is on architectures and requirements, rather than resource management.

Related to our work are recent works on resource allocation in MEC for sensor networks [10], [11]. Authors in [10] considered the problem of allocating visual sensors with correlated measurements to computing resources so as to maximize system capacity, and proposed a 3-approximation. The problem of allocating health sensors to health cloud servers to maximize system utility is proposed in [11], and was solved by an auction theory based approach. These papers consider resource allocation, but do not capture resilience requirements.

Also closely related to ours are recent works on the Virtual Network Function (VNF) placement problem [12]–[15]. These works consider the VNF placement problem in wireline networks, and focus on individual functions or on function chains. Authors in [12] modeled the individual VNF placement problem as a generalized assignment problem, where the network functions are assigned to different cloud servers to minimize the total assignment cost, and provided an approximation algorithm based on linear relaxation and rounding. [13] formulated the on-demand individual VNF placement problem as a simple lazy facility location (SLFL) problem, and proposed two heuristic algorithms for on-line optimization.

The placement of chains of VNFs was modeled in [14] as an Integer Linear Problem (ILP), and numerical solutions were provided. [15] considered minimizing the number of computing units that each function chain is distributed over, and proposed a heuristic algorithm. The joint placement and path selection problem for VNFs chains was addressed in [16] to maximize the service capacity, and algorithms for estimating link and processing capacity demands, and for allocating VNFs were proposed.

Resilience under link and node failure for VNF placement was considered in [17], the problem was formulated as an ILP, and numerical results were provided. Unlike [17], in our work we consider a set of failure scenarios caused by MEC node failures and communication link failures. We compute VPF placement for each failure scenario so as to minimize the overall cost.

While the placement of VNFs is akin to that of VPFs, the solutions for the VNF placement problem do not fit the VPF placement problem well for three reasons. First, in the case of VPF placement a VPF communicates with multiple sensors and actuators, while the VNF placement usually assumes that a VNF service request has a single entry and a single destination. Second, due to the adoption of MEC for VPF placement, sensors and actuators are associated with base stations (BSs) via radio links, and the BSs are further connected to the MEC nodes by a backhaul network. The heterogeneous network architecture has to be captured by the communication cost model. Third, the objective of VNF placement is usually to maximize the throughput of the network, while in the case of VPF placement, delay is arguably a more appropriate performance metric due to the performance requirement of industrial control processes. Our paper captures these properties of VPF placement which makes it different from the existing works on VNF placement.

Also related to ours are the works on Virtual Network Embedding (VNE), which map components of virtual networks to substrate networks. The VNE problem with the objective of minimizing the cost of using the resources of substrate networks was addressed in [18], [19] . These works assumed that all the resources of substrate networks are made available by the infrastructure provider, while we also consider the problem of making the resources of substrate networks available and the associated cost. VNE with link failures was considered in [20], where authors developed a heuristic to maximize the long term profit. [20] modeled the impact of link failures as a penalty to the profit, while we consider resilience to failures as a constraint to guarantee the availability of critical industrial applications.

Closest to our work in terms of the problem formulation is the single source capacitated facility location (SSCFL) problem, which opens a set of facilities to serve a set of clients, with the objective of minimizing the overall cost. The SSCFL problem corresponds to our problem with a single failure scenario; in the SSCFL problem the set of opened facilities and the client assignment depend on the cost and capacities of the facilities only, while in our problem the MEC node availability and the occurrence probability of each scenario should also be considered. Furthermore, in our problem, since different failure scenarios share the same set of available MEC nodes, different failure scenarios should be jointly considered to minimize the overall cost. Due to these differences, existing solution approaches for the SSCFL problem are not directly applicable to our problem. Techniques based on linear programming and local search heuristics have been developed to solve the SSCFL problem. The linear programming based techniques usually solve the problem with exceeding the capacity limits of the facilities [21], [22], which is not practical for the VPF placement. The local search heuristics [23] solve the problem of facility opening and client assignment iteratively in weakly polynomial time, and thus do not scale well to large problem instances. Furthermore, the performance bounds of local search based heuristics are not applicable to the VPF placement problem, since they usually require a metric space based cost model to guarantee an approximation ratio, which does not hold in general network topologies.

Closest to our work in terms of methodology is [24], where the problem of capacitated facility location subject to facility disruptions was addressed using Benders decomposition. Unlike [24], we also use infeasibility cuts to handle the infeasible

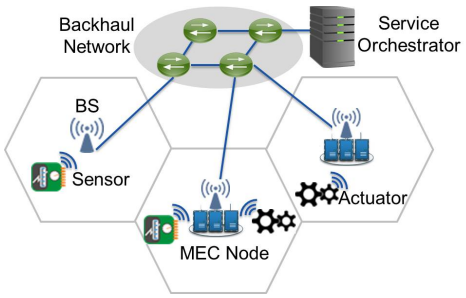


Figure 1. The communication and computing infrastructure consists of BSs, MEC nodes and a backhaul network. A service orchestrator monitors the status of the system and coordinates the allocation of communication and computing resources.

region, which makes our algorithm more efficient. This paper is an extension of our previous work [7], and improves it in three important ways. First, this paper proposes an improved initialization method for the Benders decomposition, two additional cuts, and per failure scenario based sub-problem solving. The numerical results clearly show significant improvements in terms of the convergence speed and scalability, compared to the algorithm proposed in [7]. Second, in Section V we propose a heuristic based on local search with low computational complexity, and in the numerical results section we compare its performance with that of RVP algorithm. Third, this paper provides a significantly extended performance evaluation, both in terms of problem instance sizes and the considered solutions.

III. System Model and Problem Formulation

We consider a system that consists of a set $\mathcal{B}$ of BSs, a set $\mathcal{S}$ of sensors, and a set $\mathcal{A}$ of actuators. A subset $\mathcal{M} \subseteq \mathcal{B}$ of the BSs is equipped with computational and storage resources and serve as MEC nodes [4], [25]. We denote by ω_m the computing capacity of MEC node $m \in \mathcal{M}$, and assume it to be an integer. This assumption is reasonable, for example, if capacity is the number of Virtual Machines (VMs), as industrial applications have tight delay requirements and require isolation for performance and security reasons; hence a single VPF would be allocated per VM.

The BSs are interconnected by a backhaul network (e.g., software defined mobile backhaul network). Via BSs, sensors and actuators can send and receive data wirelessly. Data communication through the wireless links and the backhaul network incurs cost, which we define in Section III-B. Our focus is primarily on delay critical applications, and thus in our model we do not consider rate requirements or link capacity limits. Instead, the communication cost is intended to model the communication delay, as we will discuss later.

Within the infrastructure above, we consider a set $\mathcal{F}$ of VPFs that process the data from sensors and send commands to actuators, and need to be allocated to MEC nodes for execution. For each function $f \in \mathcal{F}$ there is a set of sensors that capture data needed by f, and there is a set of actuators that VPF f sends commands to. We use $y_{f,s} \in \mathbf{y}$ and $z_{f,a} \in \mathbf{z}$ to indicate whether sensor $s \in \mathcal{S}$ and actuator $a \in \mathcal{A}$ are required by VPF f. We consider that each VPF requires one unit of computing resource, i.e., 1 VM. This model is reasonable, among others, for smart grid control functions that share similar control structures, e.g., Volt/Var Control (VVC) [26].

We assume that a service orchestrator monitors the status of all system components and is able to coordinate the allocation of communication and computing resources. The specification of the required backhaul network and MEC APIs is outside of the scope of our work, but it could be implemented by combining the Openflow and the OpenStack APIs [27]. Figure 1 illustrates the components of the considered communication and computing infrastructure.

A. Failure Scenarios

We consider that the communication and computing infrastructure is subject to the occasional failure of its components. We use the term failure scenario to refer to the system when a set of its components has failed, and we denote by $\mathcal{L}$ the set of all failure scenarios. A failure scenario can include a combination of communication and computing resources.

The failure of a wireless communication link, either due to equipment failure, due to jamming, or due to a denial of service attack, results in the failure of the communication between a sensor or an actuator and its associated BS. To recover from the failure, the sensor or actuator needs to be re-associated to another BS. We consider that BS association is taken care of by the mobile network, and we denote by $b_{l,i}$ the associated BS for a sensor or actuator $i \in \mathcal{A} \cup \mathcal{S}$ in scenario l. Similarly, the failure of a component of the backhaul network (e.g., an SDN switch or an optical cable) is handled by the mobile network, but it could result in increased delay.

The failure of MEC nodes, due to a communication failure, hardware failure or a DoS attack, results in a MEC node to be unsuitable for VPF placement. We use the binary variable $r_{l,m}$ to indicate the suitability of MEC node m for VPF placement in scenario l.

We assume that the system operator is able to estimate the occurrence probability of each failure scenario, and we denote the estimated occurrence probability of scenario l by π_l. The occurrence probabilities of failure scenarios can be estimated by using the mean time between failures and the mean time to repair, which can be obtained by the historical failure and maintenance records of the components of the system [28]. By definition $\sum_{l \in \mathcal{L}} \pi_l = 1$. Note that this model allows to capture correlated failures, and is thus able to capture various link layer, network layer and cloud failure recovery mechanisms. As a remark, as the number of components in the system increases the number of failure scenarios may increase. In practice, scenarios with a low occurrence probability (e.g., scenarios with multiple component failures) can be ignored to reduce the dimension of the problem.

B. Cost Model

Our cost model for VPF placement accounts for costs in terms of computing and storage resources and in terms of the communication between the MEC nodes and the sensors and actuators. We denote by F_m the availability cost of MEC node m, which has to be paid if the node is to be available for VPF placement in any scenario. The availability cost F_m is justified by the cost of storing the virtual machine images in the MEC nodes (storage cost) and by the potential need for

reserving computational power and memory for the eventual execution of the VPFs (availability fee). We denote by $p_{m,f}$ the placement cost of an instance of VPF f on node m. This cost is justified by the computational and memory resources needed for the execution of the VPF.

The cost of communication consists of the wireless transmission cost and the backhaul network transmission cost. We denote by $c_{i,b}$ the cost of wireless communication between a sensor or an actuator $i \in \mathcal{S} \cup \mathcal{A}$ and a BS b. In general it is reasonable to assume that $c_{i,b}$ is inverse proportional to the achievable rate; this cost model is well suited for capturing the delay introduced by communication and the usage of radio resource blocks.

We denote by $\bar{c}_{l,b,b'}$ the communication cost over the backhaul network between BSs b and b' in scenario l. Note that in different failure scenarios the routing between BSs b and b' may be different, thus $\bar{c}_{l,b,b'}$ is scenario dependent.

The total communication cost $c_{i,b,m}$ between a sensor or an actuator $i \in \mathcal{A} \cup \mathcal{S}$ and a MEC node $m \in \mathcal{M}$ if the sensor or actuator is associated to BS $b \in \mathcal{B}$ is then the sum of the wireless transmission cost and the backhaul network communication cost, $c_{l,m,i} = c_{i,b} + \bar{c}_{l,b,m}$. Note that the amount of traffic related to VPFs is relatively small and therefore it does not affect the delay on a link. Thus, the delay on a specific link mainly depends on its capacity and load, and therefore we consider $c_{i,b}$, $\bar{c}_{l,b,m}$ and $c_{l,m,i}$ as input parameters.

We consider that the availability cost F_m, placement cost $p_{m,f}$, and communication cost $c_{l,m,i}$ can be normalized to a common scale. Appropriate normalization factors could be introduced in the system model without affecting the problem complexity, but they would adversely affect presentation clarity, hence we decided to omit them.

C. Problem Formulation

We are now ready to formulate the resilient VPF placement problem, which consists of deciding which MEC nodes to keep available and on which MEC node to place each VPF, subject to resilience and MEC resource capacity constraints. We use the decision variable $v_m \in \{0,1\}$ to denote whether MEC node m is kept available, and let $\mathbf{v} = \{v_1, \ldots, v_m\}$. Furthermore, we use the decision variable $x_{l,f,m} \in \{0,1\}$ to denote whether VPF f is placed in MEC node m in scenario l, and let $\mathbf{x} = \{x_{1,1,1}, \ldots, x_{l,f,m}\}$.

VPF placement is subject to resilience and capacity constraints. For resilience we require that each VPF f should be placed at a MEC node in each scenario $l \in \mathcal{L}$,

$$\sum_{m \in \mathcal{M}} x_{l,f,m} \geq 1, \ \forall l \in \mathcal{L}, \ \forall f \in \mathcal{F}. \tag{1}$$

Furthermore, we consider two MEC resource capacity constraints. First, a VPF f can only be placed on a MEC node that is made available and is accessible in failure scenario l,

$$x_{l,f,m} \leq r_{l,m} v_m, \ \forall l \in \mathcal{L}, \ \forall f \in \mathcal{F}, \ \forall m \in \mathcal{M}. \tag{2}$$

Second, the sum of the computing resource requirements of the VPFs placed on MEC node m cannot exceed its computing resource capacity,

$$\sum_{f \in \mathcal{F}} x_{l,f,m} \leq \omega_m v_m, \ \forall l \in \mathcal{L}, \ \forall m \in \mathcal{M}. \tag{3}$$

The VPF placement problem is then to minimize the VPF placement cost subject to resilience and capacity constraints,

$$\underset{\mathbf{x},\mathbf{v}}{\text{minimize}} \quad O(\mathbf{v},\mathbf{x}) = \underbrace{\sum_{m \in \mathcal{M}} v_m F_m}_{\text{Availability cost}} + \sum_{l \in \mathcal{L}} \pi_l \sum_{m \in \mathcal{M}} \left(\underbrace{\sum_{f \in \mathcal{F}} x_{l,f,m} p_{m,f}}_{\text{Placement cost}} + \underbrace{\sum_{s \in \mathcal{S}} \left(c_{l,m,s} \sum_{f \in \mathcal{F}} y_{f,s} x_{l,f,m} \right) + \sum_{a \in \mathcal{A}} \left(c_{l,m,a} \sum_{f \in \mathcal{F}} z_{f,a} x_{l,f,m} \right)}_{\text{Communication cost}} \right) \tag{4}$$

$$\text{s.t.} \quad (1) - (3)$$

$$x_{l,f,m} \in \{0,1\}, \ \forall l \in \mathcal{L}, \ \forall f \in \mathcal{F}, \ \forall m \in \mathcal{M} \tag{5}$$

$$v_m \in \{0,1\}, \ \forall m \in \mathcal{M} \tag{6}$$

Our focus is on instances where the VPF placement problem above is feasible. The most frequently used notations are summarized in Table I.

Table I
TABLE OF NOTATIONS

Sets	
$\mathcal{A}$	Set of actuators
$\mathcal{B}$	Set of base stations
$\mathcal{F}$	Set of VPFs
$\mathcal{L}$	Set of failure scenarios
$\mathcal{M}$	Set of MEC nodes
$\mathcal{S}$	Set of sensors
Parameters	
$c_{l,m,i}$	Communication cost between $i \in \mathcal{A} \cap \mathcal{S}$ and $m \in \mathcal{M}$ in scenario $l \in \mathcal{L}$
F_m	Availability cost of $m \in \mathcal{M}$
$p_{m,f}$	Placement cost of $f \in \mathcal{F}$ on $m \in \mathcal{M}$
$r_{l,m}$	Indicator, suitability of $m \in \mathcal{M}$ in scenario $l \in \mathcal{L}$
$y_{f,s}$	Indicator, if sensor $s \in \mathcal{S}$ is required by $f \in \mathcal{F}$
$z_{f,a}$	Indicator, if actuator $a \in \mathcal{A}$ is required by $f \in \mathcal{F}$
π_l	Occurrence probability of $l \in \mathcal{L}$
ω_m	Capacity of MEC node $m \in \mathcal{M}$
Decision Variables	
v_m	$= \begin{cases} 1, & \text{if } m \in \mathcal{M} \text{ is made available} \\ 0, & \text{otherwise} \end{cases}$
$x_{l,f,m}$	$= \begin{cases} 1, & \text{if } f \in \mathcal{F} \text{ is placed on } m \in \mathcal{M} \text{ in } l \in \mathcal{L} \\ 0, & \text{otherwise} \end{cases}$

IV. Resilient VPF Placement Algorithm

Observe that the resilient VPF placement problem (4) is an IP problem, and in fact for $|\mathcal{L}| = 1$ the VPF placement problem is equivalent to an instance of the SSCFL problem,

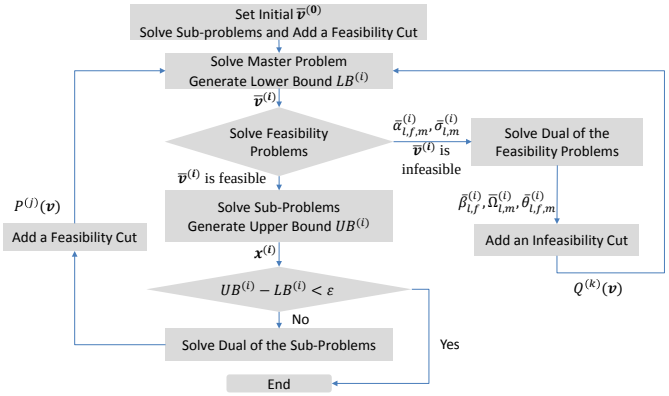


Figure 2. Flowchart of Resilient VPF Placement Algorithm

which is known to be NP-hard [23]. In what follows we show that it is possible to leverage the structure of the VPF placement problem to obtain an algorithm that is polynomial in the number of scenarios and the number of VPFs.

We notice that the constraint matrix of the VPF placement problem has a special block structure, known as block-ladder structure,

$$\begin{bmatrix} D_1 & G_1 & & & \\ D_2 & & G_2 & & \\ \vdots & & & \ddots & \\ D_{|\mathcal{L}|} & & & & G_{|\mathcal{L}|} \end{bmatrix} \tag{7}$$

where nonzero elements only appear in matrices $D_1, \ldots, D_{|\mathcal{L}|}$, which correspond to the variables $\mathbf{v}$, and in $G_1, \ldots, G_{|\mathcal{L}|}$, which correspond to the variables $x_{l,f,m}$ for each scenario, respectively.

Constraint matrices with a block-ladder structure typically arise in stochastic programming problems, for which an efficient solution method is the Generalized Benders Decomposition (GBD) [29]. In what follows we leverage this insight to propose a solution to the resilient VPF problem. The overall structure of the proposed RVP algorithm is shown in Figure 2.

Following the idea of GBD, RVP partitions the variables of the optimization problem into two sets, and decomposes the original problem into a master problem and a sub-problem. The master problem and the sub-problem are solved iteratively over different partitions, and in each iteration they generate an upper bound and a lower bound of the original problem's objective value, respectively. The iteration stops when the upper bound and the lower bound approach each other and match the termination condition. In what follows we formulate the master problem, the sub-problem, introduce the feasibility check and discuss how to initialize the algorithm. Finally, we prove that the RVP algorithm terminates in a finite number of iterations.

Throughout the section we use $\bar{\mathbf{v}}^{(i)} = \{\bar{v}_1^{(i)}, \ldots, \bar{v}_m^{(i)}\}$ and $\bar{\mathbf{x}}^{(i)} = \{\bar{x}_{1,1,1}^{(i)}, \ldots, \bar{x}_{l,f,m}^{(i)}\}$ to denote the optimal solution for the master problem and for the sub-problem in iteration i, respectively.

A. Master Problem: Availability Cost Minimization

The purpose of the master problem is to choose the MEC nodes to be made available. The number of MEC nodes that have to be made available can be lower bounded by

$$\hat{n} = \max\{\hat{n}_l | l \in \mathcal{L}\}, \tag{8}$$

where $\hat{n}_l$ is the lower bound on the number of MEC nodes required by scenario l, and is the cardinality of the minimal set of suitable MEC nodes that satisfies the capacity requirement of the VPFs.

Given $\hat{n}$, in iteration i we formulate the master problem by fixing the variable $\mathbf{x} = \bar{\mathbf{x}}^{(i-1)}$ and optimizing the expected total cost over $\mathbf{v}$,

$$\min_{\mathbf{v}} \quad g \tag{9}$$

$$\text{s.t.} \quad g \geq P^{(j)}(\mathbf{v}),\ j = 1, \ldots, J \tag{10}$$

$$0 \geq Q^{(k)}(\mathbf{v}),\ k = 1, \ldots, K \tag{11}$$

$$\sum_{m \in \mathcal{M}} v_m \geq \hat{n} \tag{12}$$

$$\sum_{l \in \mathcal{L}} \sum_{m \in \mathcal{M}} v_m r_{l,m} \omega_m \geq |\mathcal{F}| \cdot |\mathcal{L}| \tag{13}$$

$$v_m \in \{0, 1\},\ \forall m \in \mathcal{M} \tag{14}$$

The constraints (10) are so called feasibility cuts, and $P^{(j)}(\mathbf{v})$ is a support function obtained in a previous iteration by solving the dual of the sub-problem, which we will introduce in section IV-C. Note that $P^{(j)}(\mathbf{v})$ provides a lower bound on the total cost, and is a function of $\mathbf{v}$. The constraints (11) are so called infeasibility cuts. They are used to form a region of $\mathcal{V}$ that satisfies constraints (2)-(3) of the VPF placement problem, and will be introduced in Section IV-B. In each iteration the algorithm adds either a feasibility cut or an infeasibility cut, hence $J + K = i$.

Besides the usual infeasibility cuts (11) in GBD, we develop the constraints (12) and (13). These new constraints are introduced to ensure that constraints (2)-(3) are satisfied by each set of MEC nodes given by the master problem, and to tighten the solution of the master problem. Constraint (12) requires that the number of MEC nodes that are made available is no less than the lower bound $\hat{n}$ on the number of MEC nodes. Constraint (13) requires that the sum of the available capacity (e.g, the sum of the capacities of the suitable MEC nodes that are made available) over all the scenarios shall be sufficient to host $|\mathcal{F}| \cdot |\mathcal{L}|$ VPFs, since the available capacity in each scenario must be sufficient to accommodate the VPFs. As we will show later, the proposed constraints (12) and (13) can accelerate the convergence significantly.

In what follows we denote by $LB^{(i)} = \bar{g}^{(i)}$ the optimal objective function value of the master problem obtained in iteration i. Observe that the master problem is an IP in $|\mathcal{M}|$ binary variables.

B. Feasibility Check of Available MEC Nodes

As a next step, before we proceed with placing the VPFs to the available MEC nodes, we need to check if the available MEC nodes meet the capacity requirements (2) and (3). Observe that the two constraints and the VPF placement $\mathbf{x}$ are independent between scenarios, we can thus check feasibility for each scenario in parallel by formulating the following feasibility problem,

$$\min_{\boldsymbol{\alpha},\,\boldsymbol{\sigma},\,\mathbf{x}} \quad \sum_{f\in\mathcal{F}}\sum_{m\in\mathcal{M}} \alpha_{l,f,m} + \sum_{m\in\mathcal{M}} \sigma_{l,m} \tag{15}$$

s.t. (1)

$$x_{l,f,m} \le r_{l,m}\bar{v}_m^{(i)} + \alpha_{l,f,m},\ \forall l,\ \forall f,\ \forall m \tag{16}$$

$$\sum_{f\in\mathcal{F}} x_{l,f,m} \le \omega_m \bar{v}_m^{(i)} + \sigma_{l,m},\ \forall l,\ \forall m \tag{17}$$

$$x_{l,f,m} \in [0,1],\ \forall l,\ \forall f,\ \forall m \tag{18}$$

$$\alpha_{l,f,m}, \sigma_{l,m} \ge 0,\ \forall l,\ \forall f,\ \forall m \tag{19}$$

Note that variables $\boldsymbol{\alpha} = \{\alpha_{1,1,1},\dots,\alpha_{l,f,m}\}$ and $\boldsymbol{\sigma} = \{\sigma_{1,1,1},\dots,\sigma_{l,m}\}$ are added to the constraints involving $\bar{v}_m^{(i)}$ to ensure that the constraints (2) and (3) can be satisfied, and hence the problem has an optimal solution. Notice that the feasibility check does not require $x_{l,f,m}$ to be binary; as we will show in Lemma 1 linear relaxation does not change the solution set (and hence it does not change feasibility either).

Let us denote by $\bar{\alpha}^{(i)} = \{\bar{\alpha}_{1,1,1}^{(i)},\dots,\bar{\alpha}_{l,f,m}^{(i)}\}$ and $\bar{\sigma}^{(i)} = \{\bar{\sigma}_{1,1}^{(i)},\dots,\bar{\sigma}_{l,m}^{(i)}\}$ the optimal solution of the feasibility problem. If the optimal value of the objective function (15) is zero then the algorithm continues with the sub-problem described in Section IV-C.

Otherwise, we continue with formulating the dual of the feasibility problem for each failure scenario,

$$\min_{\boldsymbol{\beta},\,\boldsymbol{\theta},\,\boldsymbol{\Omega}} \quad \sum_{f\in\mathcal{F}} \beta_{l,f} - \sum_{m\in\mathcal{M}} \Omega_{l,m}\omega_m\bar{v}_m^{(i)} - \sum_{f\in\mathcal{F}}\sum_{m\in\mathcal{M}} \theta_{l,f,m}\bar{v}_m^{(i)} r_{l,m} \tag{20}$$

s.t.

$$\Omega_{l,m} + \theta_{l,f,m} - \beta_{l,f} \ge 0,\ \forall l,\ \forall f,\ \forall m \tag{21}$$

$$1 - \theta_{l,f,m} \ge 0,\ \forall l,\ \forall f,\ \forall m \tag{22}$$

$$1 - \Omega_{l,m} \ge 0,\ \forall l,\ \forall m \tag{23}$$

Let us denote by $\bar{\beta}^{(i)} = \{\bar{\beta}_{1,1}^{(i)},\dots,\bar{\beta}_{l,f}^{(i)}\}$, $\bar{\theta}^{(i)} = \{\bar{\theta}_{1,1,1}^{(i)},\dots,\bar{\theta}_{l,f,m}^{(i)}\}$ and $\bar{\boldsymbol{\Omega}}^{(i)} = \{\bar{\Omega}_{1,1}^{(i)},\dots,\bar{\Omega}_{l,m}^{(i)}\}$ the set of optimal multipliers of the duals of the feasibility problems. We use these optimal multipliers to construct an infeasibility cut

$$\begin{aligned} Q^{(k)}(\mathbf{v}) = &\sum_{l\in\mathcal{L}}\sum_{f\in\mathcal{F}} \bar{\beta}_{l,f}^{(i)}\left(1 - \sum_{m\in\mathcal{M}} \bar{x}_{l,f,m}^{(i)}\right) \\ &+ \sum_{l\in\mathcal{L}}\sum_{m\in\mathcal{M}} \bar{\Omega}_{l,m}^{(i)}\left(\sum_{f\in\mathcal{F}} \bar{x}_{l,f,m}^{(i)} - \omega_m v_m\right) \\ &+ \sum_{l\in\mathcal{L}}\sum_{f\in\mathcal{F}}\sum_{m\in\mathcal{M}} \bar{\theta}_{l,f,m}^{(i)}\left(\bar{x}_{l,f,m}^{(i)} - r_{l,m}v_m\right), \end{aligned} \tag{24}$$

which we add to the master problem (Chapter 6.3.5, [30]). After adding the infeasibility cut, the algorithm continues with the master problem. Observe that the infeasibility cut tightens the master problem, and ensures the feasibility of the sub-problem in the next iteration.

C. Sub-problems: Placement cost minimization

While the master problem chooses the set of MEC nodes that are made available to minimize the total cost, the sub-problems minimize the placement cost and the communication cost over $\mathbf{x}$ for $\bar{\mathbf{v}}^{(i)}$ computed by the master problem. Again, by observing that constraints (1)-(3) and variables $\mathbf{x}$ for the VPF placement are independent between scenarios, we can formulate a sub-problem for each scenario,

$$\begin{aligned} \min_{\mathbf{x}}\ \pi_l \sum_{m\in\mathcal{M}} &\left(\sum_{s\in\mathcal{S}}\left(c_{l,m,s}\sum_{f\in\mathcal{F}} y_{f,s}x_{l,f,m}\right) + \right. \\ &\left. \sum_{a\in\mathcal{A}}\left(c_{l,m,a}\sum_{f\in\mathcal{F}} z_{f,a}x_{l,f,m}\right) + \sum_{f\in\mathcal{F}} x_{l,f,m}p_{m,f}\right) \end{aligned} \tag{25}$$

s.t. $(1)-(3)$ and (5)

To avoid solving the sub-problems as IPs we use linear relaxation and replace constraint (5) in each sub-problem by

$$x_{l,f,m} \in [0,1],\ \forall l \in \mathcal{L},\ \forall f \in \mathcal{F},\ \forall m \in \mathcal{M}. \tag{26}$$

As we show next, linear relaxation does not affect the result of the algorithm, while reducing its computational complexity.

Lemma 1. *The coefficient matrix of constraints* (1)-(3) *for each failure scenario $l \in \mathcal{L}$ is totally unimodular. Furthermore, the linear relaxation of the sub-problem for each failure scenario $l \in \mathcal{L}$ has integral optimal solutions, which are optimal for the corresponding sub-problem.*

Proof. First we prove that the coefficient matrix of constraints (1)-(3) for each failure scenario $l \in \mathcal{L}$ is totally unimodular. We write constraints (1)-(3) according to the failure scenarios in matrix form,

$$R_l\mathbf{x}_l \le -\mathbf{1}, \tag{27}$$

$$I\mathbf{x}_l \le \mathbf{s}_l, \tag{28}$$

$$T_l\mathbf{x}_l \le \mathbf{t}_l, \tag{29}$$

where

$$R_{l,f,m} = \begin{cases} -1, & \text{if } (f-1)|\mathcal{F}|+1 \le m \le f|\mathcal{F}| \\ 0, & \text{otherwise} \end{cases} \tag{30}$$

$$\mathbf{x}_l = [x_{l,1,1},\dots,x_{l,1,m},\dots,x_{l,f,m}]^T, \tag{31}$$

$$\mathbf{s}_{l,f} = [r_{l,1}v_1,\dots,r_{l,m}v_m] \forall f \in \mathcal{F}, \tag{32}$$

$$\mathbf{s}_l = [\underbrace{\mathbf{s}_{l,1}\dots\mathbf{s}_{l,|\mathcal{F}|}}_{|\mathcal{F}|}]^T, \tag{33}$$

$$T_l = [\underbrace{I_{m,m}\dots I_{m,m}}_{|\mathcal{F}|}], \tag{34}$$

$$\mathbf{t}_l = [\omega_1 v_1,\dots,\omega_m v_m]^T. \tag{35}$$

We denote by G_l the coefficient matrix of scenario l and

$$G_l = \begin{bmatrix} G_l' \\ \mathbf{I} \end{bmatrix}, \text{ and } G_l' = \begin{bmatrix} R_l \\ T_l \end{bmatrix}. \tag{36}$$

For convenience, let us define

$$\bar{G}_l' = \begin{bmatrix} -R_l \\ T_l \end{bmatrix}. \tag{37}$$

According to the matrix formulation above, each column of $-R_l$ contains only one non-zero element with value 1, and each column of T_l contains only one non-zero element with value 1. Thus matrix $\bar{G}'_l$ has two non-zero elements in each column. To show that $\bar{G}'_l$ is totally unimodular, it suffices to show that the rows of $\bar{G}'_l$ can be partitioned into two disjoint sets such that if there are two non-zero entries in a column of $\bar{G}'_l$ then the two rows where the two entries are should be in different sets of rows (Theorem 2.3.3 in [31]). This clearly holds, if one set of the rows of $\bar{G}'_l$ consists of the rows of matrix $-R_l$ and the other set of rows of $\bar{G}'_l$ consists of the rows of matrix T_l.

Now, since multiplying some rows of a totally unimodular matrix by -1 preserves total unimodularity, G'_1 is also a totally unimodular matrix. Following Theorem 19.3 in [32], G_l is a totally unimodular matrix.

Finally, since each element of $\mathbf{s}_l$ and $\mathbf{t}_l$ is an integer and the coefficient matrix G_l is totally unimodular, according to Corollary 19.2a in [32], the sub-problem (25) for each failure scenario $l \in \mathcal{L}$ with linear relaxation has integral optimal solutions, which are optimal for the original sub-problem (25). □

Given the optimal solution $\bar{\mathbf{x}}^{(i)} = \{\bar{x}^{(i)}_{1,1,1}, \ldots, \bar{x}^{(i)}_{l,f,m}\}$ of the sub-problems, we can express the upper bound (UB) of the total cost in iteration i as

$$\begin{aligned}\text{UB}^{(i)} = \sum_{m\in\mathcal{M}} \bar{v}^{(i)}_m F_m + \sum_{l\in\mathcal{L}} \pi_l \sum_{m\in\mathcal{M}} \Bigg(\sum_{s\in\mathcal{S}} \Bigg(c_{l,m,s} \sum_{f\in\mathcal{F}} y_{f,s} \bar{x}^{(i)}_{l,f,m} \Bigg) \\ + \sum_{a\in\mathcal{A}} \Bigg(c_{l,m,a} \sum_{f\in\mathcal{F}} z_{f,a} \bar{x}^{(i)}_{l,f,m} \Bigg) + \sum_{f\in\mathcal{F}} \bar{x}^{(i)}_{l,f,m} p_{m,f} \Bigg)\end{aligned} \tag{38}$$

The algorithm terminates here if $\text{UB}^{(i)} - \text{LB}^{(i)} < \epsilon$, where $\epsilon > 0$ is the termination threshold. Otherwise the algorithm continues with adding one more constraint to the master problem. We obtain the constraint by formulating the duals of the sub-problems,

$$\begin{aligned}\max_{\boldsymbol{\lambda},\boldsymbol{\gamma},\boldsymbol{\mu}} \quad & \sum_{f\in\mathcal{F}} \lambda_{l,f} - \sum_{l\in\mathcal{L}} \sum_{m\in\mathcal{M}} \gamma_{l,m}\omega_m \\ & - \sum_{f\in\mathcal{F}} \Bigg(\sum_{m\in\mathcal{M}} \mu_{l,f,m} r_{l,m} \bar{v}^{(i)}_m \Bigg)\end{aligned} \tag{39}$$

$$\begin{aligned}\text{s.t.} \quad & \pi_l \Bigg(c_{l,m,s} \sum_{s\in\mathcal{S}} y_{f,s} + c_{l,m,a} \sum_{a\in\mathcal{A}} z_{f,a} + p_{m,f} \Bigg) \geq \\ & \lambda_{l,f} - \mu_{l,f,m} - \gamma_{l,m},\ \forall l \in \mathcal{L}, \forall f \in \mathcal{F},\ \forall m \in \mathcal{M}\end{aligned} \tag{40}$$

$$\lambda_{l,f}, \mu_{l,f,m}, \gamma_{l,m} \geq 0,\ \forall l \in \mathcal{L}, \forall f \in \mathcal{F},\ \forall m \in \mathcal{M} \tag{41}$$

where multiplier $\lambda_{l,f}$ is the marginal revenue for a placed VPF f in scenario l, $\gamma_{l,m}$ is the penalty for violation of the capacity of MEC node m in scenario l, and $\mu_{l,f,m}$ is the penalty for each violated suitability constraint in scenario l for node m. Let us denote the optimal multipliers obtained by solving the duals in iteration i by $\bar{\boldsymbol{\lambda}}^{(i)} = \{\bar{\lambda}^{(i)}_{1,1}, \ldots, \bar{\lambda}^{(i)}_{l,f}\}$, $\bar{\boldsymbol{\gamma}} = \{\bar{\gamma}^{(i)}_{1,1}, \ldots, \bar{\gamma}^{(i)}_{l,m}\}$, and $\bar{\boldsymbol{\mu}} = \{\bar{\mu}^{(i)}_{1,1}, \ldots, \bar{\mu}^{(i)}_{l,f,m}\}$.

Based on the solution of the sub-problems and of the duals of the sub-problems, we build the support function $P^{(j)}(\mathbf{v})$. In GBD the support function can be built in various ways. Since the objective function and the constraints of the VPF placement problem (4) are linearly separable in $\mathbf{v}$ and $\mathbf{x}$, we build the support function based on the Lagrange function (Chapter 6.3.5, [30]),

$$\begin{aligned}P^{(j)}(\mathbf{v}) = O(\mathbf{v}, \bar{\mathbf{x}}^{(i)}) + \sum_{l\in\mathcal{L}} \sum_{f\in\mathcal{F}} \bar{\lambda}^{(i)}_{l,f} \Bigg(1 - \sum_{m\in\mathcal{M}} \bar{x}^{(i)}_{l,f,m} \Bigg) \\ + \sum_{l\in\mathcal{L}} \sum_{m\in\mathcal{M}} \bar{\gamma}^{(i)}_{l,m} \Bigg(\sum_{f\in\mathcal{F}} \bar{x}^{(i)}_{l,f,m} - \omega_m v_m \Bigg) \\ + \sum_{l\in\mathcal{L}} \sum_{f\in\mathcal{F}} \sum_{m\in\mathcal{M}} \bar{\mu}^{(i)}_{l,f,m} \left(\bar{x}^{(i)}_{l,f,m} - r_{l,m} v_m \right).\end{aligned} \tag{42}$$

Finally, we add the support function $P^{(j)}(\mathbf{v})$ as a feasibility cut to the master problem in the next iteration.

Remark 1. *Since the feasibility problem* (15)−(19) *and its dual* (20)−(23)*, and the subproblem* (25) *and its dual* (39)−(41)*, are independent among the scenarios, they can be solved in parallel in the corresponding steps to shorten the running time of the RVP algorithm.*

Remark 2. *Observe that through linear relaxation of the sub-problem RVP reduces the number of integer decision variables from $|\mathcal{M}|(|\mathcal{L}||\mathcal{F}| + 1)$ to $|\mathcal{M}|$. As we will see this allows us to solve up to moderate instances of the VPF problem.*

D. Initializing RVP

We initialize the algorithm with a feasible set of MEC nodes $\bar{\mathbf{v}}^{(0)}$ and start with solving the sub-problems and their duals to obtain $\text{UB}^{(0)}$ and the feasibility cut $P^{(1)}(\mathbf{v})$, which can then be added to the master problem. A straightforward way to initialize the algorithm is to set $\bar{\mathbf{v}}^{(0)} = \mathbf{e}$, which makes all the MEC nodes available. However, the feasibility cut $P^{(1)}(\mathbf{v})$ generated by $\bar{\mathbf{v}}^{(0)} = \mathbf{e}$ can be very loose since all the MEC nodes are made available.

We propose an alternative way for initializing $\bar{\mathbf{v}}^{(0)}$. Our proposal is based on formulating and solving the linear relaxation of (4) for each scenario l, and rounding up the element-wise maximum of the optimal solutions $\bar{\mathbf{v}}$ of each scenario as $\bar{\mathbf{v}}^{(0)}$. Since the relaxed solution $\bar{\mathbf{v}}$ is feasible to the corresponding sub-problem, so is $\bar{\mathbf{v}}^{(0)}$. The $P^{(1)}(\mathbf{v})$ generated by this alternative approach is tighter, and can speed up the convergence as we will show later.

E. Convergence of the RVP Algorithm

Since RVP is iterative, a fundamental question is whether it is guaranteed to terminate. The following result shows that this is the case.

Theorem 1. *The RVP algorithm terminates in a finite number of iterations for any $\epsilon > 0$.*

Proof. We prove the result by showing that the RVP algorithm satisfies the following five properties:

1) The domain of $\mathbf{x}$ is a nonempty, convex set.

2) The objective function (4), and constraints (1)-(3) are convex for each fixed $\mathbf{v}$. The constraints are convex real functions on the domain of $\mathbf{x}$ for each fixed $\mathbf{v}$.
3) $\mathbf{x}$ is bounded and closed, and the constraint functions are continuous on the domain of $\mathbf{x}$ for each fixed $\mathbf{v}$.
4) The sub-problem (25) has a finite solution.
5) The sub-problem (25) has optimal Lagrangian multipliers for constraints (1)-(3).

Condition 1) holds due to Lemma 1, which showed that linear relaxation provides integral solutions to the sub-problems. Thus, even if $\mathbf{x}$ are binary in the original problem formulation, we can consider the domain of $\mathbf{x}$ as the domain of its linear relaxation, which is a convex set. Since $O(\mathbf{v}, \mathbf{x})$ and constraints (1)-(3) are linear functions, they are convex, and continuous in $\mathbf{x}$ for each fixed $\mathbf{v}$, therefore condition 2) is satisfied. According to the problem formulation the domain of $\mathbf{x}$ is bounded, and hence due to the linearity of constraints (1)-(3) condition 3) is satisfied. Since all the parameters are finite and the domain of $\mathbf{x}$ is bounded, condition 4) is satisfied as well. Finally, since we assumed that there is at least one feasible solution to problem (1)-(4), the sub-problems are feasible. As a consequence, there exist optimal Lagrangian multipliers for the constraints (1)-(3).

Since the RVP algorithm satisfies conditions 1) to 5), by Theorem 6.4.3 in [30] it terminates in a finite number of iterations for any $\epsilon > 0$. □

Theorem 1 indicates that the convergence of RVP does not depend on the initialization, i.e., $\mathbf{v}^{(0)}$. Furthermore, as the following result shows, the difference of the total cost obtained by RVP starting from two different initializations and using different sets of constraints can be bounded by the sum of the convergence thresholds.

Lemma 2. *Consider a VPF placement problem instance. Let $O(\mathbf{v}^+, \mathbf{x}^+)$ and $O(\mathbf{v}', \mathbf{x}')$ be the total cost given by RVP with initialization $\mathbf{v}^{(0)+}$ and $\mathbf{v}^{(0)\prime}$ and convergence threshold ϵ^+ and ϵ', with constraints* (12) *and* (13) *and without, respectively. Then $|O(\mathbf{v}', \mathbf{x}') - O(\mathbf{v}^+, \mathbf{x}^+)| \leq \epsilon^+ + \epsilon'$.*

Proof. First, observe that the constraints (12) and (13) speed up convergence by excluding sets of MEC nodes that make the subproblem infeasible, but they do not have an impact on set of feasible solutions considered by RVP. To prove the result, let us denote by $O(\mathbf{v}^\star, \mathbf{x}^\star)$ the optimal solution of the VPF placement problem. At the iteration when RVP converges, $O(\mathbf{v}^\star, \mathbf{x}^\star)$ is also bounded by the upper bound and lower bound generated by the RVP algorithm, thus

$$|O(\mathbf{v}^\star, \mathbf{x}^\star) - O(\mathbf{v}^+, \mathbf{x}^+)| \leq \epsilon^+,$$

$$\text{and } |O(\mathbf{v}', \mathbf{x}') - O(\mathbf{v}^\star, \mathbf{x}^\star)| \leq \epsilon'.$$

Since

$$\begin{aligned} &|O(\mathbf{v}^\star, \mathbf{x}^\star) - O(\mathbf{v}^+, \mathbf{x}^+)| + |O(\mathbf{v}', \mathbf{x}') - O(\mathbf{v}^\star, \mathbf{x}^\star)| \\ &\geq \left|O(\mathbf{v}^\star, \mathbf{x}^\star) - O(\mathbf{v}^+, \mathbf{x}^+) + O(\mathbf{v}', \mathbf{x}') - O(\mathbf{v}^\star, \mathbf{x}^\star)\right| \\ &= \left|O(\mathbf{v}', \mathbf{x}') - O(\mathbf{v}^+, \mathbf{x}^+)\right|, \end{aligned}$$

we obtain

$$\left|O(\mathbf{v}', \mathbf{x}') - O(\mathbf{v}^+, \mathbf{x}^+)\right| \leq \epsilon^+ + \epsilon',$$

which proves the lemma.

□

Algorithm 1: Single Exchange VPF Placement Algorithm

Let $\mathbf{v} = \mathbf{e}$.
Solve sub-problem (25) for each failure scenario to obtain $\mathbf{x}$.
Let $v_m = \min(1, \sum_{l \in \mathcal{L}} \sum_{f \in \mathcal{F}} x_{l,f,m})\ \forall m \in \mathcal{M}$
$C = O(\mathbf{v}, \mathbf{x})$
do
  $C' = C$
  for $\forall m$ *with* $v_m = 1$ **do**
    $\mathbf{v}' = \{v_1, \ldots, v_{m-1}, 0, v_{m+1}, \ldots, v_{|\mathcal{M}|}\}$
    $(\mathbf{v}, \mathbf{x}, C)$=Update($\mathbf{v}'$)
  end
  for $\forall m$ *with* $v_m = 0$ **do**
    $\mathbf{v}' = \{v_1, \ldots, v_{m-1}, 1, v_{m+1}, \ldots, v_{|\mathcal{M}|}\}$
    $(\mathbf{v}, \mathbf{x}, C)$=Update($\mathbf{v}'$)
  end
  for $\forall m$ *with* $v_m = 1$ **do**
    for $\forall v_{m'} = 0$ **do**
      $\mathbf{v}' = \{v_1, \ldots, v_{m-1}, 0, v_{m+1}, \ldots, v_{m'-1}, 1, v_{m'+1}, \ldots, v_{|\mathcal{M}|}\}$
      $(\mathbf{v}, \mathbf{x}, C)$=Update($\mathbf{v}'$)
    end
  end
while $C < C'$

Function Update($\mathbf{v}'$)
Solve sub-problem (25) for each failure scenario l to obtain $\mathbf{x}'$
if *Sub-problem* (25) *is feasible for* $\forall l$ **then**
  if $C > O(\mathbf{v}', \mathbf{x}')$ **then**
    $C = O(\mathbf{v}', \mathbf{x}')$, $\mathbf{v} = \mathbf{v}'$, and $\mathbf{x} = \mathbf{x}'$
  end
end
return $(\mathbf{v}, \mathbf{x}, C)$

V. Single Exchange VPF Placement Algorithm

The RVP algorithm decomposes the VPF placement problem into multiple subproblems, which can be solved in polynomial time. However, the master problem is still an IP and may be computationally intensive to solve for many MEC nodes. To potentially overcome this computational issue, in this section we propose a heuristic based on local search, which we call the Single Exchange VPF Placement (SEVP) algorithm, to solve the VPF placement problem. The proposed SEVP algorithm, shown in Algorithm 1, is an extension of local search algorithms [23] developed for the SSCFL problem to multiple failure scenarios.

The algorithm starts with making all MEC nodes available, and solves the sub-problem for each scenario to get a VPF

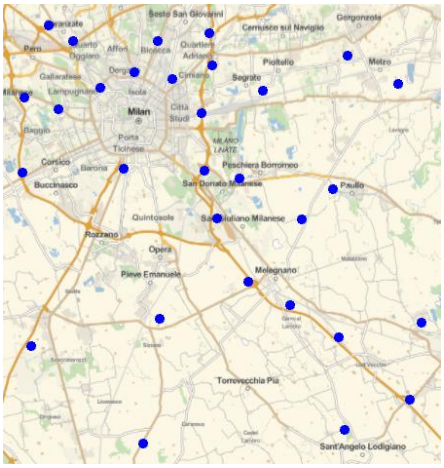


Figure 3. The simulated $50km \times 50km$ metropolitan area, based on a map of the city of Milan, Italy. Blue dots show the location of the MEC nodes.

allocation that minimizes the sub-problem cost. It then makes available the MEC nodes that are used by any VPF in any scenario (Lines 1-3). The cost of the obtained initial solution is $C = O(\mathbf{v}, \mathbf{x})$ (Line 4).

Starting from this initial solution, the SEVP algorithm improves the solution iteratively. In each iteration the SEVP algorithm stores the cost of the previous iteration as C' (Line 6), attempts to find a new solution $\mathbf{v}'$ by closing a MEC node that is made available (Line 8), and then it calls the *Update()* function (Line 9). The *Update()* function computes the optimal VPF placement $\mathbf{x}'$ with respect to $\mathbf{v}'$ by solving sub-problem (25) for each failure scenario (Line 22). If sub-problem (25) is feasible for each scenario and the cost $\mathcal{O}(\mathbf{v}', \mathbf{x}') < C$, the new solution is adopted, otherwise, it is ignored (Lines 24-26). The SEVP algorithm further attempts to find a new solution by making an extra MEC node available, and by closing an available MEC node and making an extra MEC node available simultaneously in Lines 11-14 and Lines 15-20, respectively. The SEVP algorithm terminates when C cannot be reduced any more.

Operations like closing multiple available MEC nodes and making multiple MEC nodes available simultaneously would increase the complexity of local search significantly, and therefore are not considered. Note that different from local search algorithms developed for the SSCFL problem [23], the SEVP algorithm assigns VPFs to MEC nodes by solving Linear Programming (LP) problems instead of using local search, and therefore it can scale to large systems. In what follows we use the SEVP algorithm as a basis of comparison for the RVP algorithm.

VI. Numerical Results

In what follows we evaluate the RVP and the SEVP algorithms in terms of operational cost, efficiency, and scalability. For the evaluation we simulated a $50km \times 50km$ metropolitan area, based on a map of the city of Milan, Italy, including locations of the BSs (http://opencellid.org/), and used the k-means algorithm for selecting the subset of MEC nodes, as shown by the blue dots in Figure 3.

Table II
System Parameters

Parameter	Value
Number of BSs	$\lvert\mathcal{B}\rvert = 15, 30$
Number of MEC nodes	$\lvert\mathcal{M}\rvert = 15, 30$
Number of Sensors	$\lvert S\rvert = 100$
Number of VPFs	$\lvert F\rvert = 8, 16, \ldots, 80$
Availability Cost	$F_m \sim \text{Unif}(1, 100)$
Placement Cost	$p_{m,f} \sim \text{Unif}(1, 10)$
Termination Threshold	$\epsilon = 2\%$ of UB

Sensors and actuators were uniformly placed in the simulation area. Each sensor and actuator sends and receives data through the nearest BSs. Each VPF is associated with an actuator and with 5 sensors chosen at random. The probability that sensor s is associated with f is inversely proportional to the square of the distance $d_{a,s}$ between the sensor s and the actuator a associated to f,

$$P\left(y_{f,s} = 1 \mid z_{f,a} = 1\right) = \frac{1/d_{a,s}^2}{\sum_{s' \in \mathcal{S}} 1/d_{a,s'}^2}. \tag{43}$$

We model the wireless transmission cost $c_{i,b}$ between a sensor or an actuator $i \in \mathcal{A} \cup \mathcal{S}$ and BS $b \in \mathcal{B}$ to be inversely proportional to the maximum data rate of the link given by its Shannon capacity,

$$c_{i,b} = \frac{n}{\log_2\left(1 + \frac{p_0 d_{i,b}^{-h}}{N_0}\right)},$$

where n is a positive cost coefficient, p_0 is the fixed transmit power of the transmitter, $d_{i,b}$ is the distance between i and b, h is the path loss factor, and N_0 is the Additive White Gaussian Noise (AWGN). This cost model is reasonable since a radio link with a lower unit capacity requires more radio resource (e.g., radio spectrum) for the same bitrate.

In terms of failure scenarios, we include the single node failure of each MEC node and the scenario that all the MEC nodes are suitable. We set the occurrence probability π_l of the scenarios by first choosing a failure rate $u_l \sim \text{Unif}(1, 10), l \in \mathcal{L}$ for each scenario, and used the normalized failure rate as probability, $\pi_l = u_l / \sum_l u_l$. The failure rate is widely used to assess hardware components in operating systems, network design, and in the manufacturing industry [33]–[35]. As it only appears in the objective function, the choice of π_l has limited impact on the running time of the algorithms. The results shown are the averages of 100 simulations, the confidence intervals are at the 95% confidence level. The system parameters are summarized in Table II.

A. Cost Performance

As baseline of comparison for the RVP and the SEVP algorithms we use a greedy algorithm, which only executes Lines 1-4 of Algorithm 1 to minimize the sub-problem cost.

We start with comparing the cost performance of the RVP, the SEVP, and the greedy algorithm with respect to the number of VPFs. Figure 4 shows the total cost and the sub-problem cost of the RVP and the SEVP algorithm normalized by that

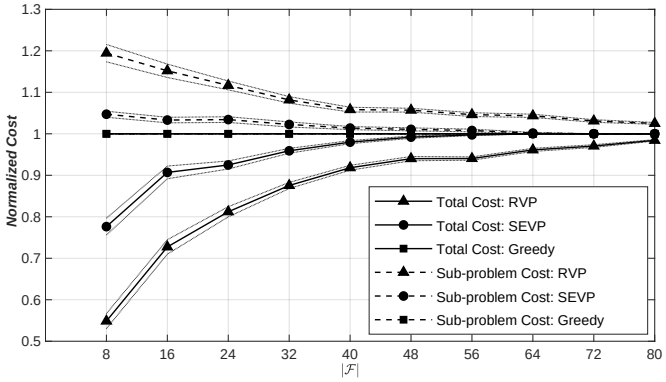


Figure 4. Normalized total cost and sub-problem cost vs. the number of VPFs for a system of 30 MEC nodes.

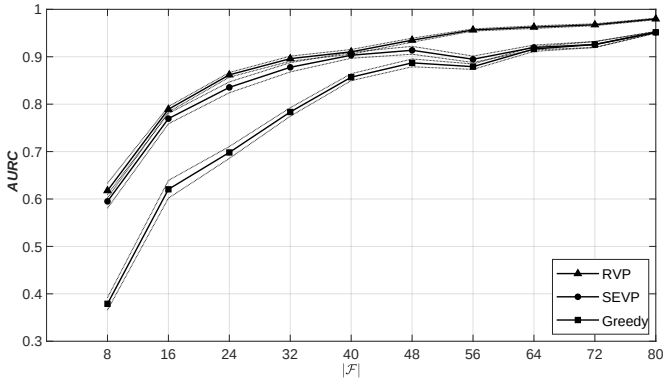


Figure 5. AURC vs. the number of VPFs for a system of 30 MEC nodes.

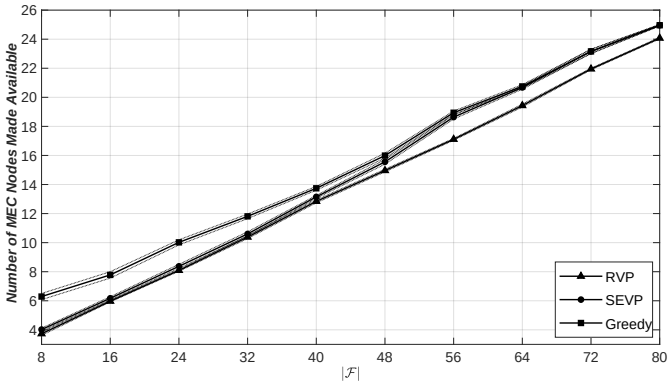


Figure 6. Average number of MEC nodes that are made available vs. the number of VPFs for a system of 30 MEC nodes.

of the greedy algorithm, for 30 MEC nodes as a function of the number of VPFs. In terms of the total cost, the RVP algorithm outperforms the SEVP algorithm, which outperforms the baseline greedy algorithm. It is interesting to observe that in terms of the sub-problem cost the greedy algorithm performs better than the RVP and the SEVP algorithm. This is because the greedy algorithm optimizes the sub-problem cost at the price of increasing the availability cost, while the RVP algorithm performs joint optimization over the availability cost and the sub-problem cost to minimize total cost, and the SEVP performs local search to lower the total cost. Furthermore, the SEVP algorithm tends to perform close to the greedy algorithm when the number of VPFs is high. This is because as $|\mathcal{F}|$ increases more MEC nodes have to be made available to accommodate the VPFs, and thus there are less feasible solutions that the SEVP algorithm can find. Clearly, in practice the performance gain of the RVP and the SEVP algorithm compared to the greedy algorithm depends on the system parameters, mainly F_m and $p_{m,f}$.

Besides the total cost, we consider the average utilization ratio of capacities (AURC) as a second performance metric,

$$AURC = \frac{|\mathcal{F}|}{\sum_{l\in\mathcal{L}} \pi_l \sum_{m\in\mathcal{M}} r_{l,m}\omega_m v_m}. \quad (44)$$

The AURC shows how well the algorithms utilize the available MEC nodes. Figure 5 shows the AURC of the available MEC nodes for the scenario shown in Figure 4. The results show that both the RVP and the SEVP algorithm outperform the greedy algorithm by up to 63% and 57%, respectively.

Figure 6 shows the average number of MEC nodes that are made available for the scenarios shown in Figure 4. We observe that on average the RVP and the SEVP algorithms make less MEC nodes available than the greedy algorithm. Note that in general making less MEC nodes available is not equivalent to a high AURC. For example, an algorithm that makes a few MEC nodes with high capacity available to host a few VPFs would achieve a low AURC.

B. Convergence and Efficiency of the RVP algorithm

Figure 7 shows the convergence of the RVP algorithm for 40 VPFs and 30 MEC nodes. The solid line and the dotted line show the lowest upper bound and the lower bound in each iteration, respectively. During each of the iterations that the UB is absence, the algorithm encountered an infeasible set of MEC nodes made available by the master problem, and thus an infeasibility cut has been added. In the example convergence happens after 204 iterations, i.e., after considering 204 subsets of $\mathcal{M}$. As a comparison, using an exhaustive search over all subsets of MECs nodes would have to consider $2^{30} \approx 1.07e9$ subsets of $\mathcal{M}$.

Figure 8 shows the average running time of the three algorithms as a function of the number of VPFs for 15 and 30 MEC nodes. First, we focus on the scenarios with $|\mathcal{M}| = 30$. While the running time of the greedy algorithm is in general lowest, as expected, it is determined by the dimension of the sub-problems, and thus it increases as $|\mathcal{F}|$ increases. Compared to the greedy algorithm, the running time of the SEVP algorithm is heavily influenced by local search, especially when the number of VPF is low, as in this case local search can generate many feasible solutions. Compared to these two algorithms, the RVP algorithm needs to solve an IP problem and multiple LP problems in each iteration, thus its running time is in general higher. It is worth noting that the running time of RVP decreases when $|\mathcal{F}|$ is increased beyond 40. This is because increasing $|\mathcal{F}|$ increases the number of MEC nodes that are needed to be made available, and therefore the feasible region of variable v_m decreases. As an effect, the RVP algorithm needs fewer iterations to converge, and therefore its running time decreases.

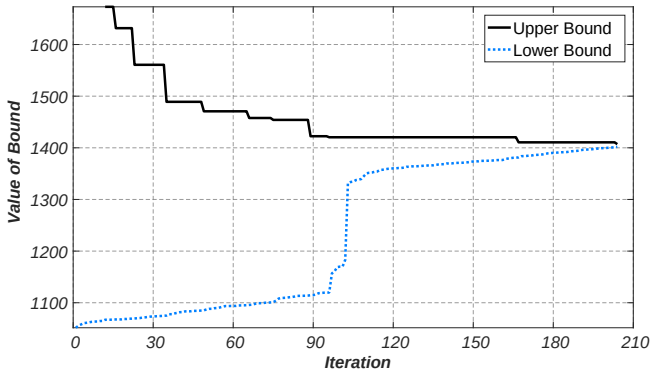


Figure 7. The convergence performance of the RVP algorithm for solving a sample VPF placement problem with 40 VPFs and 30 MEC nodes. The RVP algorithm converges at iteration 204.

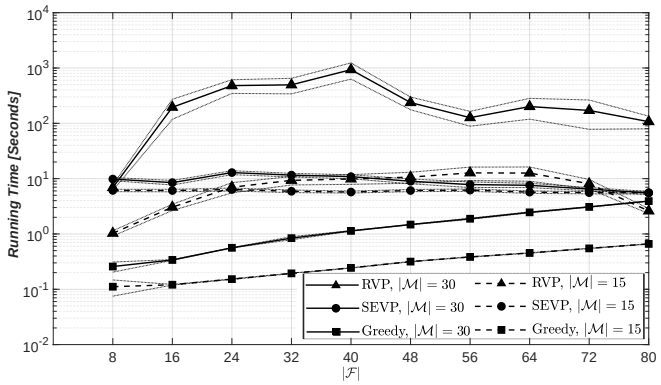


Figure 8. Average running time for 15 and 30 MEC nodes vs. number of VPFs.

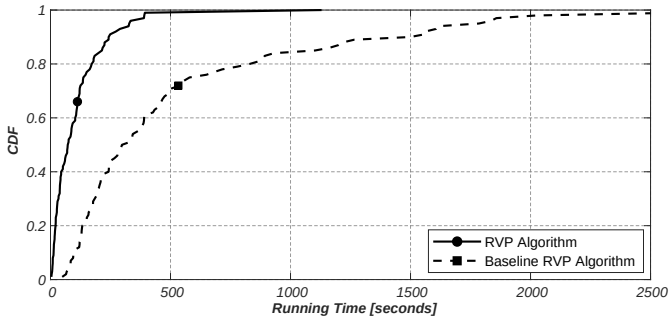


Figure 9. Distribution of the running time by the RVP algorithm and the baseline RVP algorithm for a system with 30 MEC nodes and 16 VPFs.

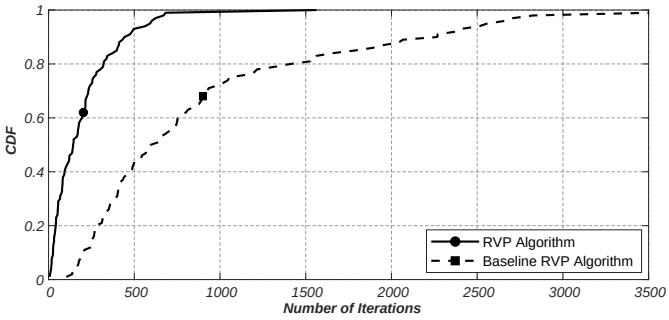


Figure 10. Distribution of the number of iterations by the RVP algorithm and the baseline RVP algorithm for a system with 30 MEC nodes and 16 VPFs.

The results for $|\mathcal{M}| = 15$ in Figure 8 show similar behavior as those for $|\mathcal{M}| = 30$. Comparing the running time of RVP for $|\mathcal{M}| = 15$ and $|\mathcal{M}| = 30$ indicates that the running time of RVP scales exponentially with $|\mathcal{M}|$. This is because increasing $|\mathcal{M}|$ increases the dimension of the master problem, of the subproblems and their dual, and of the feasibility problems and their dual. Also, the number of sets of MEC nodes that make the VPF placement feasible increases as $|\mathcal{M}|$ increases. Similarily, the SEVP and the greedy algorithm take more time for 30 MEC nodes than for 15 MEC nodes due to the increased dimension of the problem, but the difference in terms of running time is significantly smaller than for RVP. It is interesting to see that when $|\mathcal{F}| = 80$, the SEVP algorithm takes about the same time for 30 MEC nodes and for 15 MEC nodes. This is because the SEVP algorithm has to make most of the MEC nodes available to host 80 VPFs, and thus the number of solutions that SEVP can explore is limited.

Finally, comparing Figures 4 and 8 we can obtain interesting insight into the cost vs. running time trade-off. It is interesting to see that for $|\mathcal{F}|$=8 the running time of the RVP algorithm is lower than that of the SEVP algorithm, and at the same time this is when RVP provides the highest cost performance improvement over SEVP. Thus, when the system is lightly loaded, RVP is clearly superior to the SEVP algorithm. At the same time, for moderate to high number of VPFs the SEVP algorithm has significantly lower running time than RVP, at the price of a somewhat higher total cost, and may provide a good trade-off between cost and computational complexity.

In what follows we evaluate the benefit of including constraints (12) and (13) in the master problem (Section IV-A) and of the proposed initialization (Section IV-D). For the evaluation, we implemented RVP without constraints (12) and (13) and with initialization $\bar{\mathbf{v}}^{(0)} = \mathbf{e}$. We refer to this as baseline RVP. Recall that by Lemma 2 the difference of cost obtained by the two versions of RVP is negligible. Figure 9 shows the distribution of the running time for a system with 30 MEC nodes and 16 VPFs, while Figure 10 shows the distribution of the number of iterations for the same scenario. The markers on the curves in Figure 9 and 10 show the average values of the corresponding quantities. Figure 9 shows that the percentiles of the running time for the RVP algorithm are about one fourth of those for the baseline RVP. This is because constraints (12) and (13), and the alternative initialization effectively tighten the feasible region of the variables. Therefore, in general RVP requires less iterations to converge, as Figure 10 shows that the percentiles of the number of iterations for RVP are also about one fourth of those for the baseline RVP. Overall, constraints (12) and (13), and the alternative initialization can speed up the convergence for the RVP by up to 400% for the considered problem instances.

Besides showing the importance of constraints (12) and (13) and the proposed initialization, Figures 9 and 10 show that the average running time and the average number of iterations for the RVP algorithm are higher than the corresponding median values. This is caused by a number of problem instances that take a very long time to converge. In practice, a larger convergence threshold ϵ at the price of relatively higher cost can shorten the running time.

Overall, our results show that joint optimization of the available MEC nodes and the placement of VPFs can provide a significant cost reduction compared to a greedy allocation and a local search heuristic, and the proposed VPF algorithm is a computationally efficient solution for moderate instances of the resilient VPF placement problem.

VII. Conclusion

We have proposed a framework and an algorithm for solving problem of the resilient placement of virtual process control functions with the objective to minimize the computing and communication cost subject to capacity and resilience constraints. The iterative algorithm we proposed makes use of the special structure of the optimization problem, and together with a linear relaxation, the Benders decomposition based approach effectively reduces the search space. Extensive numerical results show that the proposed algorithm reduces the total cost significantly compared to a greedy baseline algorithm and a local search heuristic. Furthermore, the numerical results show that the proposed algorithm can scale to larger problem instances, independent of the number of resilience scenarios, and we note that its running time can be further reduced if the feasibility and infeasibility cuts are computed in parallel on a per scenario basis.

Our work is a first step towards the study of resilient process control function placement in 5G mobile networks. Interesting extensions of our work include the investigation of column generation methods for solving the virtual process control function placement problem, and on-line algorithms for dynamic virtual process control function migration to support mobility, e.g., for unmanned aerial vehicles.

VIII. Acknowledgment

This work was partly funded by the EU H2020 SUCCESS project, grant agreement No. 700416, and by the MSB CERCES project.

References

[1] S. A. Boyer, *SCADA: supervisory control and data acquisition*. International Society of Automation, 2009.

[2] G. Di Orio, J. Barata, C. Sousa, and L. Flores, "Control system software design methodology for automotive industry," in *Proc. of IEEE International Conference on Systems, Man, and Cybernetics*, 2013, pp. 3848–3853.

[3] I. S. Acharyya and A. Al-Anbuky, "Towards wireless sensor network softwarization," in *Proc. of IEEE Conference on Network Softwarization*, 2016, pp. 378–383.

[4] C. Alippi, R. Fantacci, D. Marabissi, and M. Roveri, "A cloud to the ground: The new frontier of intelligent and autonomous networks of things," *IEEE Comm. Mag.*, vol. 54, no. 12, pp. 14–20, 2016.

[5] Q. C. Li, H. Niu, A. T. Papathanassiou, and G. Wu, "5G network capacity: key elements and technologies," *IEEE Vehicular Technology Magazine*, vol. 9, no. 1, pp. 71–78, 2014.

[6] Peiyue Zhao and György Dán, "Time constrained service-aware migration of virtualized services for mobile edge computing," in *Proc. of International Teletraffic Congress*, September, 2018.

[7] ——, "Resilient placement of virtual process control functions in mobile edge clouds," in *Proc. of IFIP TC6 Networking*, June, 2017.

[8] M. Yigit, V. C. Gungor, and S. Baktir, "Cloud computing for smart grid applications," *Computer Networks*, vol. 70, pp. 312–329, 2014.

[9] G. Dán, R. B. Bobba, G. Gross, and R. H. Campbell, "Cloud computing for the power grid: from service composition to assured clouds," in *Proc. of Usenix Workshop on Hot Topics in Cloud Computing (HotCloud)*, 2013.

[10] E. Emil, G. Dán, and V. Fodor, "Radio and computational resource management for fog computing enabled wireless camera netoworks," in *Proc. of IEEE GlobeCom Workshop on Internet of Everything*, 2016, pp. 1–6.

[11] S. Das, S. Misra, M. Khatua, and J. J. Rodrigues, "Mapping of sensor nodes with servers in a mobile health-cloud environment," in *Proc. of IEEE International Conference on e-Health Networking, Applications & Services (Healthcom)*, 2013, pp. 481–485.

[12] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Near optimal placement of virtual network functions," in *Proc. of IEEE INFOCOM*, 2015, pp. 1346–1354.

[13] M. Ghaznavi, A. Khan, N. Shahriar, K. Alsubhi, R. Ahmed, and R. Boutaba, "Elastic virtual network function placement," in *Proc. of IEEE CloudNet*, 2015, pp. 255–260.

[14] H. Moens and F. De Turck, "VNF-P: A model for efficient placement of virtualized network functions," in *Proc. of IEEE International Conference on Network and Service Management (CNSM)*, 2014, pp. 418–423.

[15] M. Xia, M. Shirazipour, Y. Zhang, H. Green, and A. Takacs, "Network function placement for NFV chaining in packet/optical datacenters," *Journal of Lightwave Technology*, vol. 33, no. 8, pp. 1565–1570, 2015.

[16] T.-W. Kuo, B.-H. Liou, K. C.-J. Lin, and M.-J. Tsai, "Deploying chains of virtual network functions: on the relation between link and server usage," in *Proc. of IEEE INFOCOM*, 2016, pp. 1–9.

[17] A. Hmaity, M. Savi, F. Musumeci, M. Tornatore, and A. Pattavina, "Virtual network function placement for resilient service chain provisioning," in *Proc. of IEEE International Workshop on Resilient Networks Design and Modeling (RNDM)*, 2016, pp. 245–252.

[18] J. Lischka and H. Karl, "A virtual network mapping algorithm based on subgraph isomorphism detection," in *Proc. of ACM workshop on virtualized infrastructure systems and architectures*, 2009, pp. 81–88.

[19] W. Liu, Y. Xiang, S. Ma, and X. Tang, "Completing virtual network embedding all in one mathematical programming," in *Proc. of IEEE International Conference on Electronics, Communications and Control*, 2011, pp. 183–185.

[20] M. R. Rahman and R. Boutaba, "Svne: Survivable virtual network embedding algorithms for network virtualization," *IEEE Transactions on Network and Service Management*, vol. 10, no. 2, pp. 105–118, 2013.

[21] F. A. Chudak and D. P. Williamson, "Improved approximation algorithms for capacitated facility location problems," in *Proc. of International Conference on Integer Programming and Combinatorial Optimization*, 1999, pp. 99–113.

[22] K. Jain and V. V. Vazirani, "Primal-dual approximation algorithms for metric facility location and k-median problems," in *Proc. IEEE Annual Symposium on Foundations of Computer Science*, 1999, pp. 2–13.

[23] R. K. Ahuja, J. B. Orlin, S. Pallottino, M. P. Scaparra, and M. G. Scutellà, "A multi-exchange heuristic for the single-source capacitated facility location problem," *Management Science*, vol. 50, no. 6, pp. 749–760, 2004.

[24] P. Garcia-Herreros, J. M. Wassick, and I. E. Grossmann, "Design of resilient supply chains with risk of facility disruptions," *Industrial & Engineering Chemistry Research*, vol. 53, no. 44, pp. 17 240–17 251, 2014.

[25] M. T. Beck, M. Werner, S. Feld, and S. Schimper, "Mobile edge computing: A taxonomy," in *Proc. of International Conference on Advances in Future Internet*, 2014, pp. 48–54.

[26] R. Uluski, "VVC in the smart grid era," in *IEEE PES General Meeting*, 2010, pp. 1–7.

[27] H. Li, G. Shou, Y. Hu, and Z. Guo, "Mobile edge computing: progress and challenges," in *Proc. of IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*, 2016, pp. 83–84.

[28] Q. Zhang, M. F. Zhani, M. Jabri, and R. Boutaba, "Venice: Reliable virtual data center embedding in clouds," in *Proc. of IEEE INFOCOM*, 2014, pp. 289–297.

[29] A. M. Geoffrion, "Generalized benders decomposition," *Journal of optimization theory and applications*, vol. 10, no. 4, pp. 237–260, 1972.

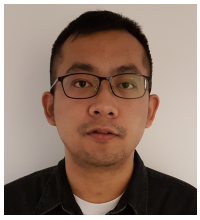
[30] C. A. Floudas, *Nonlinear and mixed-integer optimization: fundamentals and applications*. Oxford University Press, 1995.

[31] R. A. Brualdi and H. J. Ryser, *Combinatorial matrix theory*. Cambridge University Press, 1991, vol. 39.

[32] A. Schrijver, *Theory of linear and integer programming*. John Wiley & Sons, 1998.

[33] U. D. Perera, "Reliability index-a method to predict failure rate and monitor maturity of mobile phones," in *Proc. of IEEE Annual Reliability and Maintainability Symposium*, 2006, pp. 234–238.

[34] I. M. Soi and K. K. Aggarwal, "Reliability indices for topological design of computer communication networks," *IEEE Transactions on reliability*, vol. 30, no. 5, pp. 438–443, 1981.
[35] D. Pogue, "Windows 10: The missing manual." O'Reilly Media, 2015, ch. 16.



Peiyue Zhao is a Ph.D. student at the Department of Network and Systems Engineering at KTH Royal Institute of Technology. He received his B.Sc degree in Automation from Beijing Institute of Technology, China in 2013, and his M.Sc. degree in Information and Communication Technology from KTH Royal Institute of Technology, Sweden in 2015. His research interests include resource management for mobile edge computing and radio resource management.

György Dán (M'07, SM'17) is Professor of Teletraffic Systems at KTH Royal Institute of Technology, Stockholm, Sweden. He received the M.Sc. in computer engineering from the Budapest University of Technology and Economics, Hungary in 1999, the M.Sc. in business administration from the Corvinus University of Budapest, Hungary in 2003, and the Ph.D. in Telecommunications from KTH in 2006. He worked as a consultant in the field of access networks, streaming media and videoconferencing 1999-2001. He was a visiting researcher at the Swedish Institute of Computer Science in 2008, a Fulbright research scholar at University of Illinois at Urbana-Champaign in 2012-2013, and an invited professor at EPFL in 2014-2015. He has been an area editor of Computer Communications since 2014. His research interests include the design and analysis of content management and computing systems, game theoretical models of networked systems, and cyber-physical system security and resilience.